Fast, Scalable, and Accurate Algorithms for Time-Series Analysis

Ioannis Paparrizos

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2018

ABSTRACT

Fast, Scalable, and Accurate Algorithms for Time-Series Analysis

Ioannis Paparrizos

Time is a critical element for the understanding of natural processes (e.g., earthquakes and weather) or human-made artifacts (e.g., stock market and speech signals). The analysis of time series, the result of sequentially collecting observations of such processes and artifacts, is becoming increasingly prevalent across scientific and industrial applications. The extraction of non-trivial features (e.g., patterns, correlations, and trends) in time series is a critical step for devising effective time-series mining methods for real-world problems and the subject of active research for decades. In this dissertation, we address this fundamental problem by studying and presenting computational methods for efficient unsupervised learning of robust feature representations from time series. Our objective is to (i) simplify and unify the design of scalable and accurate time-series mining algorithms; and (ii) provide a set of readily available tools for effective time-series analysis. We focus on applications operating solely over time-series collections and on applications where the analysis of time series complements the analysis of other types of data, such as text and graphs.

For applications operating solely over time-series collections, we propose a generic computational framework, GRAIL, to learn low-dimensional representations that natively preserve the invariances offered by a given time-series comparison method. GRAIL represents a departure from classic approaches in the time-series literature where representation methods are agnostic to the similarity function used in subsequent learning processes. GRAIL relies on the attractive idea that once we construct the data-to-data similarity matrix most time-series mining tasks can be trivially solved. To overcome scalability issues associated with approaches relying on such matrices, GRAIL exploits time-series clustering to construct a small set of landmark time series and learns representations to reduce the data-to-data matrix to a data-to-landmark points matrix. To demonstrate the effectiveness of GRAIL, we first present domain-independent, highly accurate, and scalable time-series clustering methods to facilitate exploration and summarization of time-series collections. Then, we show that GRAIL representations, when combined with suitable methods, significantly outperform, in terms of efficiency and accuracy, state-of-the-art methods in major time-series mining tasks, such as querying, clustering, classification, sampling, and visualization. Overall, GRAIL rises as a new primitive for highly accurate, yet scalable, time-series analysis.

For applications where the analysis of time series complements the analysis of other types of data, such as text and graphs, we propose generic, simple, and lightweight methodologies to learn features from time-varying measurements. Such applications often organize operations over different types of data in a pipeline such that one operation provides input—in the form of feature vectors—to subsequent operations. To reason about the temporal patterns and trends in the underlying features, we need to (i) track the evolution of features over different time periods; and (ii) transform these time-varying features into actionable knowledge (e.g., forecasting an outcome). To address this challenging problem, we propose principled approaches to model time-varying features and study two large-scale, real-world, applications. Specifically, we first study the problem of predicting the impact of scientific concepts through temporal analysis of characteristics extracted from the metadata and full text of scientific articles. Then, we explore the promise of harnessing temporal patterns in behavioral signals extracted from web search engine logs for early detection of devastating diseases. In both applications, combinations of features with time-series relevant features yielded the greatest impact than any other indicator considered in our analysis. We believe that our simple methodology, along with the interesting domain-specific findings that our work revealed, will motivate new studies across different scientific and industrial settings.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This dissertation is the culmination of a long and arduous research effort, which would not have been feasible without the exceptional guidance, support, and inspiration I received by others. At this point, I would like to express my deep gratitude to those who contributed to the completion of this journey.

Foremost, I am forever indebted to Luis Gravano. Luis has been a truly great advisor: always available for discussion, supportive during the agitations of a Ph.D., and insightful in his guidance. Luis allowed me the freedom to pursue research outside of his research topics, yet he always had thoughtful advice and crisp ideas to push the technical depth of our work to greater levels. Luis taught me the importance of humility, the tenacity for detail, and the pursuit of principle in research and everyday life. His unceasing feedback on my ideas and his careful commentary on my drafts had a profound effect on my growth as a researcher and bolstered my ability to think clearly, communicate effectively, and write professionally. I will always be grateful for the aforementioned skills Luis has patiently taught me.

Next, I would like to express my sincere appreciation to Kathy McKeown. Kathy has been an extremely supportive and understanding mentor throughout my time at Columbia. From the very beginning of our multi-year collaboration in the FUSE project, Kathy welcomed me in her research group, showed trust in my skills, and prepared me for an independent research career by involving me in PI meetings and student mentoring. I was fortunate to have a front-row seat to witness Kathy leading effectively large teams of researchers across multiple universities during FUSE, supervising patiently a dozen of her own students, and successfully launching the Data Science Institute at Columbia. Kathy is a brilliant academic role model—an exemplar of modesty (despite her contributions and sharp intellect) and industriousness—who has tremendously influenced my research career.

I would also like to thank the other members of my committee—Dragomir Radev, Ken Ross, and Eugene Wu—who provided valuable feedback and insightful comments to improve the clarity of my dissertation. In particular, Drago has offered invaluable advice throughout my Ph.D. studies and he made sure I was kept up to speed with NLP concepts during FUSE. I am grateful to Ken for the thought-provoking and inspiring discussions during our database group meetings that significantly improved my critical thinking and my presentation skills. Finally, Eugene has been an exceptional source of advice and encouragement; I will miss his contagious energy and enthusiasm for any type of research and everyday topic.

During my Ph.D. studies I had the pleasure to collaborate with wonderful researchers who introduced me to fascinating research problems. Specifically, I am very appreciative of all participants of the FUSE project whose ideas and efforts contributed significantly to Chapter 5 of this dissertation. Many thanks go to Fernando Diaz, my mentor during my first internship at Microsoft Research, who introduced me to the world of online media mining and initiated a fruitful collaboration with Jacob Eisenstein and Hanna Wallach. In my second internship at Microsoft Research, I had the great fortune to collaborate with Eric Horvitz and Ryen White, an exceptional research duo who allowed me to contribute to their ongoing efforts towards understanding health-related web search activities. In particular, the research in Chapter 6 is joint work with Eric and Ryen, who I deeply thank for their insightful guidance.

I would also like to thank past and present students of the Database Research Group for the enjoyable and intellectually stimulating discussions. In particular, I want to thank Pablo Barrio, Bingyi Cao, Thomas Effland, Lampros Flokas, Giannis Karamanolakis, Orestis Polychroniou, Fotis Psallidas, Thibault Sellam, Goncalo Simoes, Eva Sitaridi, and Wangda Zhang, who were always available for brainstorming, conversation, and dry runs. Special thanks to Orestis, my officemate, who made sure I had not missed any of the latest (Greek) music hits, to Fotis, who was always available for a coffee break, and to Pablo, who was always ready to share stories (and his wisdom) about virtually any topic.

Throughout my years at Columbia, I was fortunate to overlap with many talented colleagues and to form new friendships. In particular, I want to thank Or Biran and Kapil Thadani for pleasant hours of chatting and brainstorming over coffee, as well as for their tireless feedback on my drafts. Special thanks go to the Greek "gang" at Columbia who was tremendously supportive on various occasions. I also greatly enjoyed my interactions with faculty and students of the Systems, NLP, and Machine Learning groups. Specifically, I am grateful to Daniel Hsu, who was always patiently answering my ML-related questions.

The completion of my Ph.D. would have never been possible without the support of my close friends. I am grateful to my friends from my hometown, Thessaloniki, who have been around from the very beginning and who were always making sure I had the most amazing time when I was returning home. I am thankful to my friends from Lausanne, who have stuck with me despite my brief stay. Finally, New York gave me many exceptional friends. I want to thank them for the great moments, the funny discussions, the trips, the gatherings in every American or Greek holiday, and for the memorable tours to explore New York.

Finally, my heartfelt gratitude to my family and to the most amazing sister and mother, without whom I would have never completed my graduate studies. I want to deeply thank my sister, Natasa, for being so caring and supportive as well as for the constant reminders to take care of myself. The biggest thank you goes to my mother, Olga, for her unconditional support, endless love, and selfless sacrifice. Thank you for always being ready to hear my stories and news, to celebrate with my successes, and cheer me up in my failures.

Dad, you are not here today with us but I know how proud you have always been for me. Thank you for influencing me to get into research and encouraging me to pursue studies in the USA. My dissertation is dedicated to you.

This work was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via the Department of Interior National Business Center (DoI/NBC) contract number D11PC20153 and by generous gifts from Microsoft Research. I am also grateful to the Alexander S. Onassis Public Benefit Foundation for awarding me a fellowship. Finally, I acknowledge computing resources from Columbia University's Shared Research

Computing Facility project, which is supported by NIH Research Facility Improvement Grant 1G20RR030893-01, and associated funds from the New York State Empire State Development, Division of Science Technology and Innovation (NYSTAR) Contract C090171, both awarded April 15, 2010.

In loving memory of my father…

Chapter 1

Introduction

This thesis studies and develops computational methods for efficient unsupervised learning of robust feature representations from time series with the purpose of (i) simplifying and unifying the design of scalable and accurate time-series mining algorithms; and (ii) providing a set of readily available tools for effective time-series analysis.

Time is a critical element for the understanding and the explanation of behaviors associated with the evolution of natural processes (e.g., earthquakes and weather) or human-made artifacts (e.g., stock market, audio, and speech signals). *Time series* are often the outcome of sequentially recording time-varying measurements of such processes and artifacts. Time series appear in almost every discipline, including astronomy (e.g., light curves of stars [Wachman et al., 2009]), biology (e.g., gene expression [Bar-Joseph et al., 2002]), chemistry (e.g., chlorine levels in drinking water [Papadimitriou et al., 2005]), meteorology (e.g., weather forecasting [Honda et al., 2002]), earth and environmental sciences (e.g., drought monitoring [Goddard et al., 2003] and automobile traffic monitoring [Papadimitriou et al., 2003]), medicine and healthcare (e.g., electrocardiogram signals [Ge and Smyth, 2000]), finance and economics (e.g., stock market signals [Mantegna, 1999; Gavrilov et al., 2000; Ruiz et al., 2012]), engineering (e.g., human motion signals [Uehara and Shimada, 2002; Keogh et al., 2004]), and others. Recent advances in data processing, storage, networking, and sensing technologies permit the collection of enormous amounts of time series across scientific and industrial applications [Palpanas, 2015; Mahdavinejad et al., 2017]. This ubiquity and proliferation of time-varying measurements has generated substantial interest in querying (i.e., detection of similar time series to a given time-series query) [Agrawal et al., 1993; Ding et al., 2008; Kin-pong and Ada, 1999; Korn et al., 1997; Lian et al., 2007; Papapetrou et al., 2011; Shou et al., 2005; Wang et al., 2014a], indexing (i.e., efficient retrieval of time series given a time-series query) [Cai and Ng, 2004; Chen et al., 2007a; Keogh, 2006; Keogh et al., 2001b; Keogh and Ratanamahatana, 2005; Vlachos et al., 2006], classification (i.e., determination of the category of a previously unseen time series) [Hu et al., 2013; Mueen et al., 2011; Ratanamahatana and Keogh, 2004; Ye and Keogh, 2009; Bagnall et al., 2017], and clustering (i.e., partitioning of time series into groups based on some notion of similarity) [Kalpakis et al., 2001; Xiong and Yeung, 2002; Alon et al., 2003; Warren Liao, 2005; Keogh and Lin, 2005; Megalooikonomou et al., 2005; Petitjean et al., 2011; Yang and Leskovec, 2011; Zakaria et al., 2012] of time series.

Prior to the application of any task-specific operations, all time-series mining algorithms first discover and exploit hidden patterns in time series [Han et al., 2011]. Despite being the subject of active research for decades across scientific fields, such as in statistics [Hamilton, 1994], signal processing [Orfanidis, 1985], pattern recognition [Lecun and Bengio, 1995; Müller et al., 1997], machine learning [Dietterich, 2002], data mining [Keogh and Kasetty, 2003], and databases [Agrawal et al., 1993; Faloutsos et al., 1994], there is currently no agreement on a set of readily available methods and tools for effective exploitation of patterns in time series. The extraction of non-trivial patterns (e.g., relations, correlations, trends, and anomalies) in time series, which are often simply referred to as features, is a critical step for devising effective methods to analyze time series and solve real-world problems. In this dissertation, we address this problem by presenting principled methods to efficiently learn feature representations from time series with the goal to solve temporal mining tasks for two types of applications: (i) applications operating solely over time-series collections; and (ii) applications where the analysis of time series complements the analysis of other types of data, such as text and graphs.

The most prominent difficulties in time-series analysis arise from three factors: (i) the temporal ordering of observations; (ii) the high dimensionality of data; and (iii) the large volumes of data. This combination of factors introduces severe complications in time-series analysis, which is considered as one of the top 10 challenging problems in data mining [Yang and Wu, 2006]. Specifically, the temporal ordering may introduce distortions in time series, which complicates the definition of models to describe time series and similarity functions to compare time series that agree with the human perception of similarity. Additionally, the high dimensionality increases the computation and storage requirements of methods operating directly over time series. Finally, the large volumes of time series may dramatically affect the performance of time-series mining algorithms, which can make effective methods over small time-series collections appear impractical in large-scale settings. Therefore, if we could automatically learn feature representations from time series while at the same time addressing the challenges associated with the temporal ordering, the high dimensionality, and the large volume of time series, we would be able to unify the design of scalable and accurate time-series mining algorithms.

The bottleneck in automating the process of learning feature representations from time series revolves around the design choices that time-series mining algorithms make to address the aforementioned challenges [Esling and Agon, 2012]. Specifically, the design of time-series mining algorithms involves taking decisions for three major components: (i) the method of choice to represent time series; (ii) the method of choice to compare time series; (iii) the method of choice to index time series. Each one of these components attempts to separately address challenges associated with time-series analysis. To address challenges associated with the high dimensionality of data, time-series representation methods construct low-dimensional representations that preserve fundamental characteristics of each individual time series. To address challenges associated with the temporal ordering of observations, time-series comparison methods define similarity functions between pairs of time series that offer invariances to inherent distortions in time series. Finally, to address challenges associated with the large volumes of data, time-series indexing methods organize time series to permit efficient retrieval from massive time-series collections. The time-series analysis community has devoted significant effort to propose new approaches and optimize existing approaches for each one of these core components [Esling and Agon, 2012].

A consequence of the separate optimization of each one of the core components is that currently there is a mismatch between the characteristics preserved by time-series representations methods, the invariances offered by effective, yet computationally expensive, similarity functions, and the properties that similarity functions require to obey so that indexing mechanisms enable efficient retrieval of time series. Therefore, existing approaches for major time-series mining tasks suffer from a number of drawbacks. Specifically, to avoid any loss of information due to the exploitation of representation methods that reduce the dimensionality of the time series, existing methods often operate directly over the original high-dimensional time series and, therefore, such methods become prohibitively expensive. For example, this is the case for the majority of time-series classification methods [Bagnall et al., 2017]. In contrast, to scale to large volumes of data, existing methods often sacrifice accuracy for efficiency in their attempt to directly exploit representations that are not suitable to capture invariances offered by highly accurate, yet expensive, similarity measures. For example, this is the case for online time-series clustering methods that currently only support less-effective ℓ_p-norms as their similarity measures in order to directly exploit existing representation methods [Ding et al., 2015]. Finally, to scale to large volumes of data without sacrificing accuracy, existing methods often follow a complicated, two-step approach to indirectly exploit representations to prune part of the expensive pairwise comparisons. For example, this is the case for querying methods that required multiple attempts to properly exploit representations that do not preserve invariances offered by the chosen similarity function [Yi et al., 1998; Kim et al., 2001; Sakurai et al., 2005b; Keogh and Ratanamahatana, 2005; Rakthanmanon et al., 2012].

The two-step approach is the most prominent paradigm to accelerate time-series mining tasks while addressing all aforementioned challenges. The requirement to perform two steps is due to the dependence on the seminal GEMINI framework [**?**; Faloutsos et al., 1994], which laid the foundations for efficient time-series retrieval under Euclidean distance (ED). Specifically, the core idea behind GEMINI is to define a distance measure over the low-dimensional representations of time series to *lower bound* (i.e., prune part of) the ED comparisons. To support distance measures other than ED, two-step approaches [Yi et al., 1998; Kim et al., 2001; Sakurai et al., 2005b; Keogh and Ratanamahatana, 2005] additionally have to show that ED lower bounds the new distance measure. Considering the plethora of representation methods and distance measures [Esling and Agon, 2012; Wang et al., 2013; Bagnall et al., 2017], as well as the difficulty in developing effective lower bounding measures, we believe that such two-step approach is not sustainable anymore. The need for time-series mining methods to exploit similarity functions other than ED is also responsible for the slow adaptation of techniques developed to accelerate computational methods in metric space, such as using sparse coding methods to learn sparse representations [Lee et al., 2007; Mairal et al., 2010] or locality sensitive hashing to answer queries approximately [Gionis et al., 1999; Kulis and Grauman, 2012].

Inspired by the insatiable demand for fast, scalable, and accurate solutions to analyze large time-series collections, in this thesis we investigate the design of computational methods to automatically learn effective, compact, and low-dimensional feature representations from time series. By constructing such time-series representations, our goal is to (i) simplify and unify the design of scalable and accurate time-series mining algorithms; and (ii) provide tools that require minimal input from scientists or practitioners and ease the analysis of massive time-series collections. Specifically, we advocate that given only a method to compare or model time series (i.e., if we address challenges associated with the temporal ordering of observations), the extraction of compact feature representations in time series can be fully automated. Importantly, such feature representations can seamlessly integrate with and, therefore, accelerate existing methods to perform major time-series mining tasks. We focus on two types of applications: (i) applications that operate solely over time-series collections; and (ii) applications where the analysis of time series complements the analysis of other types of data.

For applications operating solely over time-series collections, we propose a generic computational framework, GRAIL, to learn time-series representations that natively preserve the invariances offered by a given time-series comparison method. This is fundamentally different from current approaches in the time-series literature where representation methods are agnostic to the similarity function used in subsequent learning processes. Specifically, given a similarity function, such computational framework learns low-dimensional representations that satisfy a number of important principles: (i) preserve the pairwise similarities by the chosen similarity function on time series in the original high-dimensional space, with the goal to leverage methods that require as input feature vectors or methods developed for data in metric space; (ii) lower bound the similarity function, with the goal to facilitate time-series querying and indexing; (iii) permit reusing part of its coordinates without having to reconstruct the representation, with the goal to enable operations in online settings with limited computational resources; and (iv) support efficient and memory-tractable operations, such as similarity computation over available and new time series and eigendecomposition of the full similarity matrix, with the goal to exploit highly effective methods relying on these operations. To learn such robust representations, GRAIL relies on the attractive idea that once we construct the full data-to-data similarity matrix most time-series mining tasks can be solved using existing off-the-shelf methods. However, the scalability of approaches relying on full data-to-data matrices remains challenging. For example, the construction of such data-to-data matrices requires time that is quadratic in the number of time series. To overcome this scalability issue, our computational framework constructs a small set of landmark time series that effectively summarize time-series collections and reduces the data-to-data matrix to a data-to-landmark points matrix, which requires linear time to construct and linear space to store. Once such data-to-landmark points matrix is constructed, GRAIL learns representations to satisfy the aforementioned principles. GRAIL representations seamlessly integrate with and, therefore, accelerate existing methods to perform tasks of critical importance for time-series analysis, such as querying and indexing, clustering, classification, sampling, and visualization of time series. In Chapter 3, we present a set of novel scalable and accurate time-series clustering methods to group time series that share similar characteristics and to effectively summarize such sets of time series. Later, in Chapter 4, we exploit the cluster centroids of such clustering methods as landmark time series and we show how GRAIL represents time series as a linear combination of those landmark time series.

For applications where the analysis of time series complements the analysis of other types of data, we propose generic, simple, and lightweight methodologies to learn feature representations from time-varying measurements that seamlessly integrate with techniques developed for other types of data. Specifically, in many complex, real-world applications, we need to measure different facets of an underlying process, which requires operations over different types of data (e.g., text, graphs, and time series). These applications often organize operations in pipeline, where each operation often captures user-defined characteristics in the form of feature vectors and provides input to subsequent operations. To reason about the temporal relationships, trends, and patterns in such simple features, we need to (i) track the evolution of features over different time periods; and (ii) transform these time-varying features into actionable knowledge (e.g., predicting or forecasting an outcome). Unfortunately, because these operations often produce thousands of (simple) features, the exhaustive combination of such features and their computation over different time periods might result in an explosion of new features that we cannot afford to store or preprocess. Importantly, defining time-series comparison methods for each one of the new features becomes prohibitively time-consuming and requires expertise over the data and the underlying features. Therefore, approaches such as GRAIL, which was developed to learn feature representation for a single type of time series, are not easily applicable in such settings. To address this challenging problem that often occurs in scientific and industrial applications, as we will see, we rely on traditional, lightweight, and principled approaches to model the time-varying features produced by other operators. To demonstrate the impact of exploiting methods from time-series analysis to reason about the temporal relationships of features produced over different types of data, we study two large-scale, real-world, applications. Specifically, in Chapter 5 we study the problem of predicting the impact of scientific concepts through temporal analysis of characteristics extracted from the metadata and full text of scientific articles. Later, in Chapter 6, we explore the promise of harnessing temporal patterns in behavioral signals extracted from web search engine logs for early detection of devastating diseases.

Specifically, the key contributions of this dissertation are as follows:

- **Efficient and Accurate Time-Series Clustering:** In Chapter 3, we address the problem of domain-independent, accurate, and scalable clustering of time series. Clustering of time series has received significant attention, not only as a powerful standalone exploratory method, which identifies and summarizes interesting patterns in the underlying data, but also as a preprocessing step or subroutine for other tasks. Unfortunately, existing highly accurate methods cannot scale to large volumes of data, they are domain-specific, or they rely on expensive distance measures to compare time series. To address these drawbacks, we proceed as follows: (i) we show how to derive a distance measure from cross-correlation that offers important invariances to time-series distortions in a principled manner; we also show how to efficiently compute this measure; (ii) we present two novel methods to compute a single centroid or multiple centroids per cluster when that distance measure is used; (iii) we develop two novel algorithms for time-series clustering that rely on the aforementioned distance measure: one algorithm computes a single centroid per cluster based on the entire set of time series in each cluster, whereas the other (outlier-aware) algorithm computes multiple centroids per cluster in order to consider the proximity and spatial distribution of time series in each cluster; (iv) we present a one-nearest-neighbor classification algorithm that relies on time-series clustering as a subroutine to reduce the search space and accelerate one-nearest-neighbor algorithms; and (v) we perform an extensive experimental evaluation of our methods against state-of-the-art time-series distance measures, clustering algorithms, and classification algorithms. Our findings show that our efficient and parameter-free distance measure leads to similar results to the most accurate but computationally expensive distance measures that require parameter tuning. Our clustering methods outperform all state-of-the-art scalable and non-scalable partitional, hierarchical, spectral, density-based, and shapelet-based clustering approaches, with only one method achieving similar performance, but this method requires tuning of parameters and cannot scale to large volumes of data. Finally, our evaluation suggests that we can rely on time-series clustering methods as subroutines to effectively reduce the search space for one-nearest-neighbor time-series classification algorithms. Consequently, we believe that our methods emerge as domain-independent, accurate, and scalable approaches for time-series comparison, clustering, and classification.

- **Efficient Time-Series Representation Learning:** In Chapter 4, we address the problem of efficiently learning data-aware, low-dimensional representations of time series that preserve the invariances offered by a given similarity function. This is fundamentally different from the time-series literature where representation methods are agnostic to the similarity function used in subsequent learning processes. The current mismatch between the properties preserved by low-dimensional representations and the properties required by effective, yet computationally expensive, similarity measures, has led existing approaches for time-series mining tasks to sacrifice effectiveness, and many times accuracy, for efficiency. A promising direction to unify the properties preserved by similarity measures and representation methods arises with the use of kernel methods. Specifically, kernel methods interact with data through pairwise comparisons using a kernel function and, therefore, require constructing a data-to-data similarity matrix prior to solving any task. To alleviate the burdens associated with the high memory and runtime costs of (i) computing the data-to-data matrix and (ii) performing dimensionality reduction using such matrix, we propose GRAIL, a generic representation learning framework. GRAIL uses the Nyström method [Nyström, 1930; Williams and Seeger, 2001] to reduce the data-to-data matrix to a data-to-landmark points matrix and relies on matrix sketching [Liberty, 2013] to approximate the eigendecomposition of the data-to-data matrix and, therefore, learn low-dimensional representations that preserve invariances offered by a user-specified kernel function. The effectiveness of GRAIL critically relies on the (i) choice of kernel function; (ii) choice of landmark time series; (iii) unsupervised estimation of parameters that affects the compactness and the reconstruction quality of the learned representations. To address these issues, we proceed as follows: (i) we develop an efficient kernel function to compare time series under important invariances for time-series distortions; (ii) we construct landmark time series using the effective time-series clustering methods we developed in Chapter 3; (iii) we present a method to estimate kernel function parameters and compactness of representation; (iv) we efficiently learn GRAIL representations by approximating the eigendecomposition of the data-to-data matrix using matrix sketching; (v) we show how GRAIL representations accelerate existing methods for major time-series mining tasks, such as querying and indexing, clustering, classification, sampling, and visualization; and (vi) we evaluate our ideas by conducting an extensive experimental evaluation. Our findings show that kernel classifiers using our kernel function significantly outperform one-nearest-neighbor classifiers with state-of-the-art distance measures, which debunks a long-standing perception in the time-series literature that such classifiers are difficult to beat. GRAIL representations are more compact and have significantly better pruning power than current time-series representations. Finally, GRAIL representations, combined with kernel methods, significantly outperform, in terms of efficiency and accuracy, state-of-the-art methods that operate over the full length of time series in all aforementioned tasks. We believe that GRAIL rises as a new primitive for highly accurate, yet scalable, time-series analysis that significantly simplifies the design of new algorithms while requiring minimal user input.

- **An End-to-End System for Predicting the Impact of Scientific Concepts:** In Chapter 5, we study the problem of predicting the future impact of scientific concepts and describe an end-to-end system developed for this task by a team of researchers at Columbia and several other universities. In many complex, real-world applications, such as the one we consider in this chapter, we need to measure different facets of an underlying process, which requires answers in real time and the combination of knowledge from different types of data (e.g., graphs, text, and time series). Considering these requirements, our objective is to (i) develop generic, simple, and lightweight methodologies to effectively extract features from time-varying measurements that seamlessly integrate with techniques developed for other types of data; and (ii) demonstrate the impact of exploiting methods from time-series analysis to significantly improve the accuracy of prediction models used in such large-scale, real-world applications. To achieve this goal, we proceed as follows: (i) we review the challenges of this ambitious, yet important, problem of identifying research concepts that hold the most promise as early as possible; (ii) we describe the architecture of the system we built to facilitate the extraction of characteristics from articles in the scientific literature and real-time prediction of the future impact of scientific concepts; (iii) we outline the large number of characteristics that our system extracts from the metadata and the full text of scientific articles; (iv) we describe a variety of principled statistical measures capable of capturing patterns in short, sparse, and noisy time-varying characteristics extracted from scientific articles; and (v) we perform a large-scale experimental evaluation. Our results show the clear benefit of full text features over metadata features and that temporal features contribute the most in predictions. Specifically, the benefit of the analysis of the full text of scientific articles is well worth the increased performance cost of the natural language analysis. Importantly, across all our experiments, combinations of full text features and metadata features with time series relevant features yielded the greatest impact than any other indicator considered in our analysis. This is of particular importance, considering the lightweight methodology used to extract feature representations in comparison to approaches used in Chapters 3 and 4. We believe such simple methodology is well-suited for applications where underlying operations produce thousands of user-defined characteristics that we need to track over time and transform into actionable knowledge.

- **Detecting Devastating Diseases in Search Logs:** In Chapter 6, we explore the promise of harnessing temporal patterns in behavioral signals extracted from web search engine logs for early detection of devastating diseases. Together with collaborators from Microsoft Research, we leveraged billions of queries from the Bing search engine and studied the feasibility of doing early detection of the presence of pancreatic cancer in experiential user queries. We cast this as a binary classification task, where the model is trained on features extracted from search log query timelines of experiential pancreatic cancer searchers and symptom-only searchers. We rely on a similar methodology to the one we developed in Chapter 5 to extract features from short, sparse, and noisy time series that often appear in the analysis of web search engine logs. We proceed as follows: (i) we introduce the early detection of diseases as a promising new application of search log mining and machine learning that scales to millions of searchers; (ii) we describe the process of generating a large-scale real-world dataset to study the feasibility of early detecting experiential pancreatic cancer cases from longitudinal individual search activity; (iii) we review the large set of characteristics that we extract from query timelines; (iv) we present a simple methodology to track the temporal relationships and patterns in the content of queries over time; (v) we summarize our prediction model to forecast with significant lead times that users will later input experiential queries for pancreatic cancer; (vi) we explore the influence of different factors, such as the lead time or the presence of specific symptoms in the search activity, on the predictive performance of our learned models, including true positive rates when false positive rates are strictly controlled. Our study provides insights on the feasibility of learning from search engine logs to predict future issuance of experiential queries about pancreatic cancer at a low error rate. The success of these methods has implications for online methods that would provide passive screening of searchers, to offer early warning about potential signs of pancreatic cancer and other devastating diseases. Our evaluation suggests that the combination of static features extracted from query timelines with our temporal features yielded the greatest impact than any other feature category considered in our analysis. Additionally, our analysis revealed a number of important domain-specific findings. For example, we discovered that conditioning on different symptoms and risk factors can enhance predictive power. We found capture-cost tradeoffs associated with different symptoms and risk factors in terms of the total number of truly positive cases identified versus the number of searchers who would be mistakenly alerted. We characterized model performance as we increase lead time and we found that we can attain a true positive rate of 5-30% while controlling the false positive rate to 0.00001-0.01 many months before a diagnostic query is observed. We believe that our simple methodology to reason about the evolution of time-varying characteristics in such important, large-scale, and real-world application, along with the interesting domain-specific findings that our work revealed, will motivate new studies across different scientific disciplines and industrial settings.

We start with a review of the relevant and necessary background for time-series analysis in Chapter 2. Chapters 3, 4, 5, and 6 present our work on clustering time series, learning time-series representations, predicting the future impact of scientific concepts, and detecting devastating diseases in search logs, respectively, as we discussed above. Then, we describe related work in Chapter 7. Finally, we present our conclusions in Chapter 8 and discuss directions for future work in Chapter 9.

Chapter 2

Background and Preliminaries

In this chapter, we provide the necessary definitions and notation for time series (Section 2.1). We summarize distortions that are common in time series (Section 2.2) and the most popular distance measures for such data (Section 2.3). Then, we review common time-series representations and dimensionality reduction methods (Section 2.4) and provide a brief discussion of common tasks for time-series analysis (Section 2.5).

2.1 Definitions and Notation

A *time series* is a real-valued, high-dimensional vector that consists of observations with a natural temporal ordering. A time series is often the outcome of recording time-varying measurements of an underlying process. Specifically, a *univariate time series* contains sequentially collected observations of a single time-varying measurement whereas a *multivariate time series* contains sequentially and simultaneously collected observations of two or more time-varying measurements. Depending on the spacing between observation times, a time series is *equally spaced* if the spacing of observation times is constant and *unequally or irregularly spaced* if the spacing of observation times is not constant. The analysis of univariate, equally spaced time series is the most popular in practice because (i) methods to analyze multivariate time series are often extensions of methods for univariate time series and (ii) it is common to transform irregularly spaced time series to equally spaced time series using some form of interpolation (i.e., the process of constructing new observations within a range of known observations). In this thesis, we focus on univariate, equally spaced time series that we refer simply as time series.

We denote a time series of m ordered real values as a vector $\vec{x} = (x_1, \ldots, x_m)$, where $x_i \in \mathbb{R}$, and we omit any explicit reference to the time stamps associated with each x_i. We organize an unordered set of n time series into a *time-series dataset or database* denoted as $X = \{\vec{x}_1, \ldots, \vec{x}_n\}$, where $\vec{x}_i \in \mathbb{R}^m$, or, equivalently, in a row-wise matrix $X = [\vec{x}_1, \ldots, \vec{x}_n]^T \in \mathbb{R}^{nxm}$. Considering that the observation of an underlying process might span a large period of time or that new observations might continuously append into existing time series (e.g., as in the case of a streaming application), it becomes common to omit values of a long time series and only operate over time-series *subsequences*. Specifically, given a time series $\vec{x}$, a subsequence $\vec{s}(k,l) = (x_k, \ldots, x_{k+l-1})$, with $1 \leq k \leq m-l+1$, is a time series consisting of $l \leq m$ contiguous values starting at offset k of $\vec{x}$. For a fixed l and an offset of *step*, with $1 \leq step \leq m-l$, we can extract $\frac{m-l}{step}+1$ subsequences using a sliding window that shifts and extracts $\vec{s}(k,l)$ from each offset as follows: $S_x^l(step) = \bigcup_{i=0}^{\frac{m-l}{step}} \vec{s}(i \cdot step + 1, l)$.

The temporal ordering and the high-dimensionality of observations raise diverse facets of complications that make the analysis of time series distinct from other types of data. Specifically, the requirement to sequentially collect observations may distort time series in some way (as we see in Section 2.2) and increase the complexity of defining time-series comparison methods that agree with the human perception. Additionally, the high-dimensionality of observations increases the computation and storage requirements of methods operating directly over raw time series. Therefore, it is well understood [Esling and Agon, 2012], that core aspects for effective and scalable time-series analysis are the methods for time-series representation and time-series comparison. The goal in time-series representation is to construct X', a representation of X of reduced dimensionality $d < m$ such that X' closely approximates X. The goals In time-series comparison is to define a distance function $Dist(\vec{x}, \vec{y}) \rightarrow \mathbb{R}$ between two time series $\vec{x}$ and $\vec{y}$, such that small values of $Dist$ to indicate very similar time series whereas large values to indicate dissimilar time series. The distance measure has to be nonnegative (i.e., $D(\vec{x},\vec{y}) \geq 0$). If the distance measure is symmetric (i.e., $D(\vec{x},\vec{y}) = D(\vec{y},\vec{x})$) and satisfies the triangle inequality (i.e., given a third time series $\vec{z}$, $D(\vec{x},\vec{y}) \leq D(\vec{x},\vec{z}) + D(\vec{z},\vec{y})$), the distance measure is a metric. Distance metrics are particularly important for efficient time-series analysis as metrics are suitable to perform time-series indexing (as we will see in Section 2.4).

Depending on whether the full time series or subsequences of the time series are considered during the computation of the distance value, we observe three scenarios to compute a distance: (i) full comparison: the distance measure considers the entire time series; (ii) sequence to subsequence comparison: the distance measure slides the shorter time series against the larger time series and determines the point where the distance value between the two time series is minimized; and (iii) subsequence to subsequence comparison: the distance measure considers subsequences of the time series. In each of the three scenarios, distortions in the time series can affect the computation of the distance value and, therefore, make similar time series appear as dissimilar. Next, we review the most common distortions in time series.

2.2 Time-Series Invariances

Based on the domain, time series sequences are often distorted in some way, and distance measures need to satisfy a number of invariances in order to compare sequences meaningfully. In this section, we review common time-series distortions and their invariances.

Scaling and translation invariances: In many cases, it is useful to recognize the similarity of sequences despite differences in amplitude (scaling) and offset (translation). In other words, transforming a sequence $\vec{x}$ as $\vec{x}' = a\vec{x} + b$, where a and b are constants, should not change $\vec{x}$'s similarity to other sequences.. For example, these invariances are useful to analyze variations in currency values on exchange markets without being biased by inflation.

Shift invariance: When two sequences are similar but differ in phase (global alignment) or when there are regions of the sequences that are aligned and others are not (local alignment), we might still need to consider them similar. For example, heartbeats can be out of phase depending on when we start taking the measurements (global alignment) and handwritings of a phrase from different people will need alignment depending on the size of the letters and on the spaces between words (local alignment).

Uniform scaling invariance: Sequences that differ in length require either stretching of the shorter sequence or shrinking of the longer sequence so that we can compare them effectively. For example, this is useful for heartbeats recorded in periods of different duration.

Occlusion invariance: When subsequences are missing, we can still compare the sequences by ignoring the subsequences that do not match well. This invariance is useful in handwritings if there is a typo or a letter is missing.

Complexity invariance: When sequences have similar shape but different complexities, we might want to make them have low or high similarity based on the application. For example, audio signals that were recorded indoors and outdoors might be considered similar, despite the fact that outdoor signals will be more noisy than indoor signals.

For many tasks, some or all of the above invariances are required when we compare time-series sequences. To satisfy the appropriate invariances, we could preprocess the data to eliminate the corresponding distortions. For example, by z-normalizing [Goldin and Kanellakis, 1995] the data we can achieve the scaling and translation invariances. However, for invariances that cannot be trivially achieved with a preprocessing step, we can define sophisticated distance measures that offer distortion invariances. Next, we review the most common such distance measures.

2.3 Time-Series Distance Measures

The two state-of-the-art approaches for time-series comparison first z-normalize the sequences and then use a distance measure to determine their similarity, and possibly capture more invariances. The most widely used distance metric is the simple Euclidean distance (ED) [Faloutsos et al., 1994]. ED compares two time series $\vec{x} = (x_1, \ldots, x_m)$ and $\vec{y} = (y_1, \ldots, y_m)$ of length m as follows:

$$ED(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{2.1}$$

Another popular distance measure is Dynamic Time Warping (DTW) [Sakoe and Chiba, 1978]. DTW can be seen as an extension of ED that offers a local (non-linear) alignment. To achieve that, an m-by-m matrix M is constructed, with the ED between any two points of $\vec{x}$ and $\vec{y}$. A *warping path* $W = \{w_1, w_2, \ldots, w_k\}$, with $k \geq m$, is a contiguous set of matrix elements that defines a mapping between $\vec{x}$ and $\vec{y}$ under several constraints [Keogh and Ratanamahatana, 2005]:

$$DTW(\vec{x},\vec{y}) = \min\sqrt{\sum_{i=1}^{k} w_i} \tag{2.2}$$

This path can be computed on matrix M with dynamic programming for the evaluation of the following recurrence:

$$\gamma(i,j) = ED(i,j) + \min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\}.$$

It is common practice to constrain the warping path to visit only a subset of cells on matrix M. The shape of the subset matrix is called *band* and the width of the band is called *warping window*. The most frequently used band for constrained Dynamic Time Warping (cDTW) is the Sakoe-Chiba (SC) band [Sakoe and Chiba, 1978]. Figure 2.1a shows the difference in alignments of two sequences offered by ED and DTW, whereas Figure 2.1b presents the warping path (dark cells) for cDTW constrained by the SC band with width 5 cells (light cells).

Recently, Wang et al. [Wang et al., 2013] extensively evaluated 9 distance measures and several variants thereof. They found that ED is the most efficient measure with a reasonably high accuracy, and that DTW and cDTW perform exceptionally well in comparison to other measures. cDTW is slightly better than DTW and significantly reduces the computation time. Several optimizations have been proposed to further speed up cDTW [Rakthanmanon

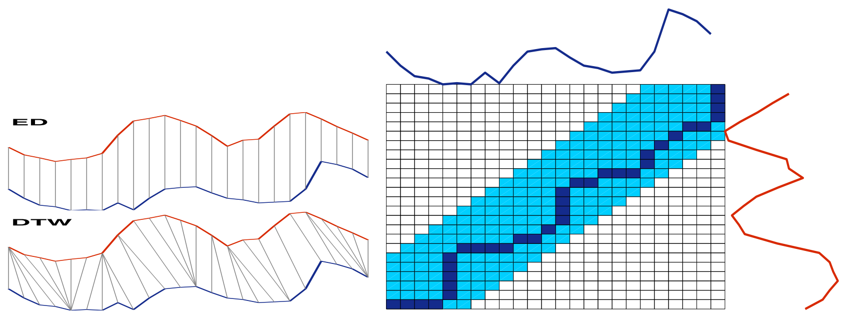


Figure 2.1: Similarity computation: (a) alignment under ED (top) and DTW (bottom), (b) Sakoe-Chiba band with a warping window of 5 cells (light cells in band) and the warping path computed under cDTW (dark cells in band).

et al., 2012]. In Chapter 3, we propose the use of an alternative distance measure that is as accurate as cDTW, but significantly faster.

Considering that dimensionality reduction techniques can extract features that preserve the characteristics of the time series and, subsequently, distance measures can operate over the time series of reduced dimensionality, we review such time-series representations in the next section.

2.4 Time-Series Representations

Apart from the choice of distance measure, another important aspect for effective comparison of time series is the choice of *representation method*. Time series are high-dimensional data and manipulation of time series in their raw format can lead to inefficient approaches due to computation and storage requirements. Instead of working directly with time series in their raw format, dimensionality reduction methods can extract features that preserve the characteristics of the time series while significantly reducing their dimensionality. Apart from the direct benefits of manipulating time series in reduced dimensionality, such as computation speedup and storage reduction, time-series representation methods can also remove noise from time series, increase the accuracy of certain tasks, and allow more effective indexing.

Since the debut of the GEMINI framework [Agrawal et al., 1993; Faloutsos et al., 1994], research on representation methods has focused on exploring trade-offs between low-dimensional representations, such as reconstructing quality, sensitivity to noise, compactness, and computational cost. Depending on the transformation applied to time series and on the output format, we divide representation methods into *data-agnostic* and *data-aware* methods and into *numeric* and *symbolic* methods [Esling and Agon, 2012].

Data-agnostic methods: The most popular representations rely on spectral decompositions. Specifically, the GEMINI framework represented time series as a set of sinusoidal coefficients using the DFT [Agrawal et al., 1993; Faloutsos et al., 1994]. Subsequently, dozens of methods were proposed to replace the sinusoidal functions, including the Discrete Cosine Transform (DCT) [Korn et al., 1997], the Discrete Wavelet Transform (DWT) [Chan and Fu, 1999], Daubechies wavelets [Popivanov and Miller, 2002], Haar wavelets [Chan et al., 2003], Coiflets [Shasha and Zhu, 2004], and Chebychev polynomials [Cai and Ng, 2004]. More specific to time series, the Piecewise Aggregate Approximation (PAA) [Yi and Faloutsos, 2000; Keogh et al., 2001a] represents time series as mean values of time-series segments.

Data-aware methods: In contrast to data-agnostic methods, data-aware methods tune transformation parameters on available data to improve their effectiveness. For example, data-aware methods relying on spectral decompositions select a subset of DFT [Vlachos et al., 2004] or DWT [Struzik and Siebes, 1999] coefficients. A data-aware version of PAA uses vector quantization to construct a codebook of segments [Megalooikonomou et al., 2004; Megalooikonomou et al., 2005], whereas other approaches, namely, Piecewise Linear Approximation (PLA) [Shatkay and Zdonik, 1996] and Adaptive Piecewise Constant Approximation (APCA) [**?**], fit a polynomial model or use a constant approximation for each segment, respectively. Linear dimensionality reduction methods, such as SVD and PCA, are also inherently data-aware methods proposed to represent time series [Korn et al., 1997; Ravi Kanth et al., 1998].

Symbolic methods: The output of all previous methods is numeric. Symbolic methods additionally quantize the numeric output. For example, the Symbolic Aggregate approXimation (SAX) [Lin et al., 2003] and the Symbolic Fourier Approximation (SFA) [Schäfer and

Högqvist, 2012] rely on alphabets to transform PAA and DFT representations, respectively, into short words.

Despite the abundance of time-series representation methods, existing approaches cannot preserve similarities between time series under important invariances (e.g., shifting and warping) and, therefore, sacrifice effectiveness for efficiency. In Chapter 4, we present a numeric representation method to address the problem of representation learning that preserves the properties of any given distance measure.

Time-series representation methods, along with distance measures discussed earlier, are critical components of time-series analysis techniques that we discuss next.

2.5 Time-Series Tasks

In time-series analysis, the objective is to apply data mining algorithms to discover and extract previously unknown information in time series. Clustering and classification of time series are two of the most common tasks in time-series analysis. The aim of clustering [Warren Liao, 2005] is to partition time series into groups based on some notion of similarity in order to extract patterns and correlations. Usually, the correct number of partitions is unknown in advance and labels are not available to guide the partition process (i.e., this is an unsupervised learning scenario). In Chapter 3, we address the problem of time-series clustering and we present two efficient and scalable algorithms that significantly outperform prior state-of-the-art methods. The aim of classification [Bagnall et al., 2016] is to predict the categories of previously unseen time series based on a model learnt for each category. In contrast to clustering, the number of categories are known and examples of time series exist for each category (i.e., this is a supervised learning scenario). In Chapter 4, we address the problem of time-series classification and we present a unified framework to learn representations and efficiently classify time series. Other tasks include motif discovery (i.e., extraction of frequently occurring patterns in time series) [Chiu et al., 2003], discord discovery (i.e., extraction of abnormal patterns) [Keogh et al., 2005], and prediction (i.e., forecasting future values after observation of existing values in time series) [Chatfield, 2000] in time series.

Chapter 3

Efficient and Accurate Time-Series Clustering

In Chapter 2, we described the necessary background for time-series analysis. In this chapter, we exploit that background and focus on the problem of time-series clustering. Clustering, the partitioning of data into groups based on some notion of similarity, is one of the first and still one of the most popular data mining tasks. Despite the plethora of clustering methods, a particular method, namely the k-means algorithm [MacQueen, 1967], has been recognized as one of the most influential data mining algorithms of all times [Wu et al., 2008] due to its simplicity, efficiency, and wide applicability. Unfortunately, k-means is not suitable for time-series clustering, as we will see. Our goal is to develop novel clustering methods that build on the same successful principles of k-means but are suitable for time-series clustering. First, we provide an overview and motivation for this task (Section 3.1). Second, we describe relevant theoretical background and review existing approaches for time-series clustering and time-series averaging (Section 3.2). Third, we present our time-series clustering algorithms (Section 3.3). Then, we exploit time-series clustering to accelerate one-nearest-neighbor classification algorithms (Section 3.4) and summarize our extensive experimental evaluation (Sections 3.5 and 3.6). Finally, we conclude with the contributions of this chapter (Section 3.7).

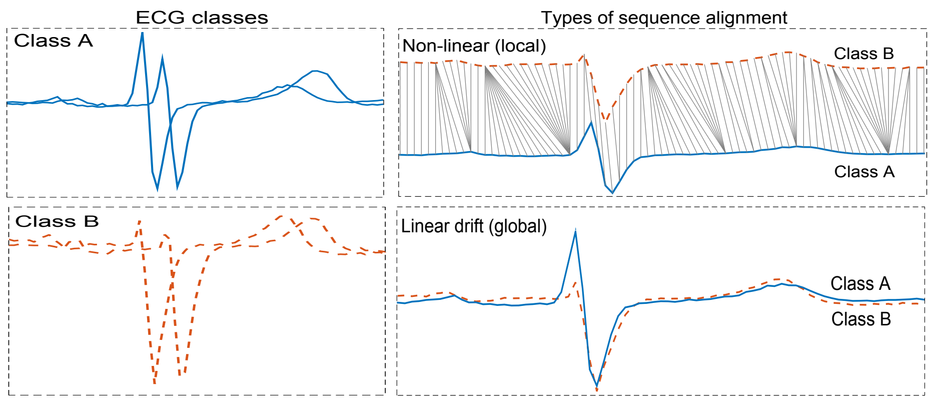


Figure 3.1: ECG sequence examples and types of alignments for the two classes of the ECGFiveDays dataset [Keogh et al., 2015].

3.1 Overview and Motivation

Among all techniques applied to time-series sequences, clustering is one of the most widely used as it does not rely on costly human supervision or time-consuming annotation of data. With clustering, we can identify and summarize interesting patterns and correlations in the underlying data [Halkidi et al., 2001]. In the last few decades, clustering of time-series sequences has received significant attention [Bagnall and Janacek, 2004; Das et al., 1998; Gavrilov et al., 2000; Li et al., 1998; Oates, 1999; Petitjean et al., 2011; Rakthanmanon et al., 2011; Yang and Leskovec, 2011; Zakaria et al., 2012], not only as a powerful standalone exploratory method, but also as a preprocessing step or subroutine for other tasks.

Most time-series analysis techniques, including clustering, critically depend on the choice of distance measure. A key issue when comparing two time-series sequences is how to handle the variety of distortions, as we discussed (see Section 2.2), that are characteristic of the sequences. To illustrate this point, consider the well-known ECGFiveDays dataset [Keogh et al., 2015], with ECG sequences recorded for the same patient on two different days. While the sequences seem similar overall, they exhibit patterns that belong in one of two distinct classes (see Figure 3.1): Class A is characterized by a sharp rise, a drop, and another gradual increase while Class B is characterized by a gradual increase, a drop, and another gradual increase. Ideally, a *shape-based* clustering method should generate a partition similar to the classes shown in Figure 3.1, where sequences exhibiting similar patterns are placed into the same cluster based on their *shape* similarity, regardless of differences in amplitude and phase. As the notion of shape cannot be precisely defined, dozens of distance measures have been proposed [Chen and Ng, 2004; Chen et al., 2005; Chen et al., 2007b; Ding et al., 2008; Faloutsos et al., 1994; Frentzos et al., 2007; Morse and Patel, 2007; Stefan et al., 2013; Vlachos et al., 2002; Wang et al., 2013] to offer invariances to multiple inherent distortions in the data. However, it has been shown that distance measures offering invariances to amplitude and phase perform exceptionally well [Ding et al., 2008; Wang et al., 2013] and, hence, such distance measures are used for shape-based clustering [Meesrikamolkul et al., 2012; Niennattrakul and Ratanamahatana, 2009; Petitjean et al., 2011; Yang and Leskovec, 2011].

Due to these difficulties and the different needs for invariances from one domain to another, more attention has been given to the creation of new distance measures rather than to the creation of new clustering algorithms. It is generally believed that the choice of distance measure is more important than the clustering algorithm itself [Batista et al., 2014]. As a consequence, time-series clustering relies mostly on classic clustering methods, either by replacing the default distance measure with one that is more appropriate for time series, or by transforming time series into “flat” data so that existing clustering algorithms can be directly used [Warren Liao, 2005]. However, the choice of clustering method can affect: (i) accuracy, as every method expresses homogeneity and separation of clusters differently; and (ii) efficiency, as the computational cost differs from one method to another. For example, spectral clustering [Filippone et al., 2008] or certain variants of hierarchical clustering [Kaufman and Rousseeuw, 2009] are more appropriate to identify density-based clusters (i.e., areas of higher density than the remainder of the data) than partitional methods such as k-means [MacQueen, 1967] or k-medoids [Kaufman and Rousseeuw, 2009]. On the other hand, k-means is more efficient than hierarchical, spectral, or k-medoids methods.

Unfortunately, state-of-the-art approaches for shape-based clustering, which use partitional methods with distance measures that are scale- and shift-invariant, suffer from two main drawbacks: (i) these approaches cannot scale to large volumes of data as they depend on computationally expensive methods or distance measures [Meesrikamolkul et al., 2012; Niennattrakul and Ratanamahatana, 2009; Petitjean et al., 2011; Yang and Leskovec, 2011]; (ii) these approaches have been developed for particular domains [Yang and Leskovec, 2011] or their effectiveness has only been shown for a limited number of datasets [Meesrikamolkul et al., 2012; Niennattrakul and Ratanamahatana, 2009]; and (iii) these approaches are sensitive to outliers and noise as they do not take into consideration the proximity and spatial distribution of time series. Moreover, the most successful shape-based clustering methods handle phase invariance through a local, non-linear alignment of the sequence coordinates, even though a global, linear alignment is often adequate and in certain cases more accurate. For example, for the ECG dataset in Figure 3.1, an efficient linear drift can reveal the underlying differences in patterns of sequences of two classes, whereas an expensive non-linear alignment attempts to match every corresponding increase or drop of each sequence, making it difficult to distinguish the two classes (see Figure 3.1). Importantly, in contrast to linear alignment, the effectiveness of non-linear alignments might degrade for time series with large variability in their prefix and suffix coordinates [Silva et al., 2016].

Despite the advantages of the simple linear alignment, starting with the application of DTW decades ago [Berndt and Clifford, 1994], research on the problem of time-series comparison focused on elastic distance measures that offer non-linear alignment [Chen and Ng, 2004; Chen et al., 2005; Keogh and Ratanamahatana, 2005; Morse and Patel, 2007; Vlachos et al., 2002]. Interestingly, independent experimental evaluations of state-of-the-art distance measures for time-series comparison — 9 measures and their variants in [Ding et al., 2008; Wang et al., 2013] and 48 measures in [Giusti and Batista, 2013] — did not consider distance measures offering global linear alignment. Our goal is to develop shape-based clustering methods that handle the phase invariance through a global linear alignment and, subsequently, perform an extensive evaluation against other shape-based clustering methods, against other partitional methods, and against different clustering approaches, such as hierarchical or spectral methods.

In this chapter, we propose k-Shape and k-MS, two novel algorithms for shape-based time-series clustering that are efficient and domain independent. k-Shape and k-MS are based on a scalable iterative refinement procedure similar to the one used by the k-means algorithm, but with significant differences. Specifically, k-Shape and k-MS use both a different distance measure and a different method for centroid computation from those of k-means. As argued above, k-Shape and k-MS attempt to preserve the shapes of time series while comparing them. Furthermore, k-MS also considers the proximity and spatial distribution of time series. To do so, k-Shape and k-MS require a distance measure that is invariant to scaling and shifting. Unlike other clustering methods [Meesrikamolkul et al., 2012; Petitjean et al., 2011; Yang and Leskovec, 2011], we adapt the cross-correlation measure and we show: (i) how we can derive in a principled manner a time-series distance measure that is scale- and shift-invariant; and (ii) how this distance measure can be computed efficiently by exploiting intrinsic characteristics of Fourier transform algorithms. Based on the properties of this normalized version of cross-correlation, namely, SBD, we develop ShapeExtraction (SE) and MultiShapesExtraction (MSE), two novel methods to compute cluster centroids, which are used in every iteration to update the assignment of time series to clusters. k-Shape relies on SE to compute a single centroid per cluster based on the entire set of time series in each cluster. In contrast, k-MS relies on MSE to compute multiple centroids per cluster in order to consider the proximity and spatial distribution of time series in each cluster.

To demonstrate the effectiveness of SBD, k-Shape, and k-MS we have conducted an extensive experimental evaluation over 85 datasets and compared the state-of-the-art distance measures and clustering approaches for time series using rigorous statistical analysis. We took steps to ensure the reproducibility of our results, including making available our source code as well as using public datasets. Specifically, our results show that SBD is competitive, outperforming ED [Faloutsos et al., 1994], and achieving similar accuracy as cDTW [Sakoe and Chiba, 1978], one of the best performing distance measures [Wang et al., 2013], without requiring any tuning and performing significantly faster. Interestingly, our results further suggest that only the supervised tuning of cDTW leads to significant improvements in classification accuracy of cDTW over the unconstrained Dynamic Time Warping (DTW). In contrast, DTW outperforms cDTW in clustering accuracy, which contradicts the belief that cDTW generally improves the accuracy of DTW.

For time-series clustering, we show that the k-means algorithm with ED, in contrast to what has been reported in the literature, is a robust approach and that inadequate modifications of the distance measure and the centroid computation can reduce its performance. Our results suggest that partitional methods outperform hierarchical, spectral, density-based, and shapelet-based methods with the most competitive distance measures. These results indicate that the choice of algorithm, which is sometimes believed to be less important than that of distance measure, is as critical as the choice of distance measure. Similarly, we show that key characteristics of the clustering methods, such as the linkage criterion in hierarchical clustering, can significantly affect the accuracy of the clustering results, whereas the choice of distance measure is many times less important.

Our extensive experimental evaluation shows that k-Shape outperforms all scalable and non-scalable partitional, hierarchical, spectral, density-based, and shapelet-based methods in terms of accuracy, with the only exception of one existing approach that achieves similar accuracy results, namely, k-medoids with cDTW. However, there are problems with this approach that can be avoided with k-Shape: (i) the requirement of k-medoids to compute the dissimilarity matrix makes it unable to scale and particularly slow, two orders of magnitude slower than k-Shape; (ii) its distance measure requires tuning, either through automated methods that rely on labeling of instances or through the help of a domain expert; this requirement is problematic for clustering, which is an unsupervised task. In contrast, k-Shape uses a parameter-free and efficient distance measure. Overall, k-Shape is a scalable — yet accurate — choice for time-series clustering that achieve state-of-the-art performance across different domains and is particularly effective for applications involving similar but out-of-phase sequences.

k-MS behaves similarly to k-Shape in comparison to scalable and non-scalable rival methods but, importantly, k-MS is significantly more accurate than k-Shape on datasets with large variance in the proximity and spatial distribution of time series. Therefore, k-MS is suitable to cluster time series in the presence of outliers and noise.

Overall, k-Shape and k-MS are scalable — yet accurate — choices for time-series clustering that achieve state-of-the-art performance across different domains and are particularly effective for applications involving similar but out-of-phase sequences. In addition to our clustering results, and inspired by the work by [Petitjean et al., 2014; Petitjean et al., 2016], we show how k-Shape can also lead to efficient techniques for other related time-series tasks. Specifically, in this chapter we show that we can rely on k-Shape as a subroutine to effectively reduce the search space for one-nearest-neighbor time-series classification algorithms. Later, in Chapter 4, we exploit the cluster centroids of k-Shape as landmark time series to efficiently learn representations by expressing time series as a linear combination of landmark time series.

We start with a review of the state of the art for clustering time series, as well as with a precise definition of our problem of focus (Section 3.2). We continue as follows:

- We show how to derive SBD, a scale-, translate-, and shift-invariant distance measure, in a principled manner from cross-correlation and how to efficiently compute this measure by exploiting intrinsic characteristics of Fourier transform algorithms (Section 3.3.1).

- We present two novel methods to compute a single centroid or multiple centroids per cluster when the SBD distance measure is used (Section 3.3.2).

- We develop k-Shape and k-MS, two algorithms for time-series clustering (Section 3.3.3).

- We present NSC, a one-nearest-neighbor classification algorithm that relies on k-Shape as subroutine to reduce the search space of one-nearest-neighbor algorithms (Section 3.4).

- We perform an extensive experimental evaluation of our ideas (Sections 3.5 and 3.6).

Finally, we conclude and discuss the implications of our work (Section 3.7). The material described in this chapter appears in [Paparrizos and Gravano, 2015; Paparrizos and Gravano, 2016; Paparrizos and Gravano, 2017].

We now review necessary background and define our problem of focus in this chapter.

3.2 Preliminaries

In this section, we first review the relevant theoretical background (Section 3.2.1). Later, we summarize existing approaches for clustering time-series data (Section 3.2.2) and for centroid computation (Section 3.2.3). Then, we formally present our problem of focus (Section 3.2.4).

3.2.1 Theoretical Background

Clustering is the general problem of partitioning n observations into k clusters, where a *cluster* is characterized with the notions of homogeneity — the similarity of observations within a cluster — and separation — the dissimilarity of observations from different clusters. Even though many clustering criteria to capture homogeneity and separation have been proposed [Hansen and Jaumard, 1997], the minimum within-cluster sum of squared distances is most commonly used as it expresses both of them. Given a set of n observations $X = \{\vec{x}_1, \ldots, \vec{x}_n\}$, where $\vec{x}_i \in \mathbb{R}^m$, and the number of clusters $k < n$, the objective is to partition X into k pairwise-disjoint clusters $P = \{p_1, \ldots, p_k\}$, such that the within-cluster sum of squared distances is minimized:

$$P^* = \arg\min_{P} \sum_{j=1}^{k} \sum_{\vec{x}_i \in p_j} dist(\vec{x}_i, \vec{c}_j)^2 \tag{3.1}$$

where $\vec{c}_j$ is the centroid of partition $p_j \in P$. In Euclidean space this is an NP-hard optimization problem for $k \geq 2$ [Aloise et al., 2009], even for number of dimensions $m = 2$ [Mahajan et al., 2009]. Because finding a global optimum is difficult, heuristics such as the k-means method [MacQueen, 1967] are often used to find a local optimum. Specifically, k-means randomly assigns the data points into k clusters and then uses an iterative procedure that performs two steps in every iteration: (i) in the assignment step, every data point is assigned to the cluster of its nearest centroid, which is determined with the use of a distance function; (ii) in the refinement step, the centroids of the clusters are updated to reflect the changes in cluster memberships. The algorithm converges either when there is no change in cluster memberships or when the maximum number of iterations is reached.

In the refinement step, k-means computes new centroids to serve as representatives of the clusters. The centroid is defined as the data point that minimizes the sum of squared distances to all other data points and, hence, it depends on the distance measure used. Finding such a centroid is known as the Steiner's sequence problem [Petitjean and Gançarski, 2012]: given a partition $p_j \in P$, the corresponding centroid $\vec{c}_j$ needs to fulfill:

$$\vec{c}_j = \arg\min_{\vec{w}} \sum_{\vec{x}_i \in p_j} dist(\vec{w}, \vec{x}_i)^2, \quad \vec{w} \in \mathbb{R}^m \tag{3.2}$$

When ED is used, the centroid can be computed with the arithmetic mean property [Dimitriadou et al., 2002]. In many cases where alignment of observations is required, this problem is referred to as the multiple sequence alignment problem, which is known to be NP-complete [Wang and Jiang, 1994]. In the context of time series, Dynamic Time Warping (DTW) (see Section 2.3) is the most widely used measure to compare time-series sequences with alignment, and many heuristics have been proposed to find the average sequence under DTW (see Section 3.2.3).

3.2.2 Time-Series Clustering Algorithms

Several methods have been proposed to cluster time series. All approaches generally modify existing algorithms, either by replacing the default distance measures with a version that is more suitable for comparing time series (raw-based methods), or by transforming the sequences into "flat" data so that they can be directly used in classic algorithms (feature- and model-based methods) [Warren Liao, 2005]. Raw-based approaches can easily leverage the vast literature on distance measures (see Section 2.3), which has shown that invariances offered by certain measures, such as DTW, are general and, hence, suitable for almost every domain [Ding et al., 2008]. In contrast, feature- and model-based approaches are usually domain-dependent and applications on different domains require that we modify the features or models. Because of these drawbacks of feature- and model-based methods, in this chapter we follow a raw-based approach.

The four most popular raw-based methods are agglomerative hierarchical, spectral, density-based, and partitional clustering [Batista et al., 2014; Warren Liao, 2005]. For hierarchical clustering, the most widely used "linkage" criteria are the single, average, and complete linkage variants [Kaufman and Rousseeuw, 2009]. Spectral clustering [Ng et al., 2002] has recently started receiving attention [Batista et al., 2014] due to its success over other types of data [Filippone et al., 2008]. Similarly, density-based clustering [Ester et al., 1996] has gained popularity due to its ability to handle outliers in datasets [Begum et al., 2015]. Among partitional methods, k-means [MacQueen, 1967] and k-medoids [Kaufman and Rousseeuw, 2009] are the most representative examples. When partitional methods use distances that offer invariances to scaling, translation, and shifting, we consider them as shape-based approaches. From these methods, k-medoids is usually preferred [Warren Liao, 2005]: unlike k-means, k-medoids computes the dissimilarity matrix of all data sequences and uses actual sequences as cluster centroids; in contrast, k-means requires the computation of artificial sequences as centroids, which hinders the easy adaptation of distance measures other than ED. However, from all these methods, only the k-means class of algorithms can scale linearly with the size of the datasets. Recently, k-means was modified to work with: (i) DTW [Petitjean et al., 2011] and (ii) a distance that offers pairwise scaling and shifting of sequences [Yang and Leskovec, 2011]. Both of these modifications rely on new approaches to compute cluster centroids that we will review next.

3.2.3 Time-Series Averaging Techniques

The computation of an average sequence or, in the context of clustering, a centroid, is a difficult task and it critically depends on the distance measure used to compare time series. We now review the state-of-the-art methods for the computation of an average sequence.

With ED, the property of arithmetic mean is used to compute an average sequence (e.g., as is the case in the centroid computation of k-means). However, as DTW is more appropriate for many time-series tasks [Keogh and Ratanamahatana, 2005; Rakthanmanon et al., 2012], several methods have been proposed to average sequences under DTW. Nonlinear alignment and averaging filters (NLAAF) [Gupta et al., 1996] uses a simple pairwise method where each coordinate of the average sequence is calculated as the center of the mapping produced by DTW. This method is applied sequentially to pairs of sequences until only one pair is left. Prioritized shape averaging (PSA) [Niennattrakul and Ratanamahatana, 2009] uses a hierarchical method to average sequences. The coordinates of an average sequence are computed as the weighted center of the coordinates of two time-series sequences that were coupled by DTW. Initially, all sequences have weight one, and each average sequence produced in the nodes of the tree has a weight that corresponds to the number of sequences it averages. To avoid the high computation cost of previous approaches, Ranking Shape-based Template Matching Framework (RSTMF) [Meesrikamolkul et al., 2012] approximates an ordering of the time-series sequences by looking at the distances of sequences to all other cluster centroids, instead of computing the distances of all pairs of sequences.

Several drawbacks of these methods have led to the creation of a more robust technique called Dynamic Time Warping Barycenter Averaging (DBA) [Petitjean et al., 2011], which iteratively refines the coordinates of a sequence initially picked from the data. Each coordinate of the average sequence is updated with the use of barycenter of one or more coordinates of the other sequences that were associated with the use of DTW. Among all these methods, DBA seems to be the most efficient and accurate averaging approach when DTW is used [Petitjean et al., 2011]. Another averaging technique that is based on matrix decomposition was proposed as part of K-Spectral Centroid Clustering (KSC) [Yang and

Leskovec, 2011], to compute the centroid of a cluster when a distance measure for pairwise scaling and shifting is used. In our approach, which we will present in Section 3.3, we also rely on matrix decomposition to compute centroids.

3.2.4 Problem Definition

We address the problem of domain-independent, accurate, and scalable clustering of time series into k clusters, for a given value of the target number of clusters k.[1] Even though different domains might require different invariances to data distortions (see Section 2.2), we focus on distance measures that offer invariances to scaling and shifting, which are generally sufficient (see Section 2.3) [Ding et al., 2008]. Furthermore, to easily adopt such distance measures, we focus our analysis on raw-based clustering approaches, as we argued in Section 3.2.2. Next, we introduce k-Shape and k-MS, our novel time-series clustering algorithms.

3.3 Shape-based Clustering of Time Series

Our objective is to develop a domain-independent, accurate, and scalable algorithm for time-series clustering with a distance measure that is invariant to scaling and shifting. In this section, we first discuss our distance measure, namely, SBD, which is based on cross-correlation (Section 3.3.1). Based on SBD, we then present two methods to compute centroids of time-series clusters (Section 3.3.2). Finally, we propose k-Shape and k-MS, two novel clustering algorithms that rely on an iterative refinement procedure that scales linearly in the number of sequences and generates homogeneous and well-separated clusters (Section 3.3.3).

3.3.1 Time-Series Shape Similarity

Capturing shape-based similarity requires measures that can handle distortions in amplitude and phase. Unfortunately, the best performing measures offering invariances to these

[1]Although the estimation of k is difficult without a gold standard, we can do so by varying k and evaluating clustering quality with criteria to capture information intrinsic to the data [Kaufman and Rousseeuw, 2009].

distortions, such as DTW, are computationally expensive (see Section 2.3). To circumvent this efficiency limitation, we adapt a normalized version of cross-correlation.

Cross-correlation is a measure of similarity that compares points of time-lagged signals one-to-one. Cross-correlation is widely used in signal processing to compare sequences that differ in phase (global alignment). In contrast, DTW is a measure that is able to compare regions of sequences (local alignment). After DTW [Berndt and Clifford, 1994], research on the problem of time-series comparison has mostly focused on elastic measures that compare one-to-many or one-to-none points [Chen and Ng, 2004; Chen et al., 2005; Keogh and Ratanamahatana, 2005; Morse and Patel, 2007; Vlachos et al., 2002; Wang et al., 2013]. Therefore, the relative performance of elastic measures against cross-correlation remains largely unexplored. Different needs from one domain or application to another hinder the process of finding appropriate normalizations for the data and the cross-correlation measure. Moreover, inefficient implementations of cross-correlation can make it particularly slow. In the rest of this section, we show how to address these drawbacks. Specifically, we show how to choose normalizations that are domain-independent and efficient, and lead to a shape-based distance for comparing time series efficiently and effectively.

Cross-correlation: Cross-correlation is a measure with which we can determine the similarity of two sequences $\vec{x} = (x_1, \ldots, x_m)$ and $\vec{y} = (y_1, \ldots, y_m)$, even if they are not properly aligned.[2] To achieve shift-invariance, cross-correlation keeps $\vec{y}$ static and slides $\vec{x}$ over $\vec{y}$ to compute their inner product for each *shift* s of $\vec{x}$. We denote a shift of a sequence as follows:

$$\vec{x}_{(s)} = \begin{cases} (\overbrace{0, \ldots, 0}^{|s|}, x_1, x_2, \ldots, x_{m-s}), & s \geq 0 \\ (x_{1-s}, \ldots, x_{m-1}, x_m, \underbrace{0, \ldots, 0}_{|s|}), & s < 0 \end{cases} \tag{3.3}$$

When all possible shifts $\vec{x}_{(s)}$ are considered, with $s \in [-m, m]$, we produce $CC_w(\vec{x}, \vec{y}) =$

[2]For simplicity, we consider sequences of equal length even though cross-correlation can be computed on sequences of different length.

$(c_1, \ldots, c_w)$, the cross-correlation sequence with length $2m-1$, defined as follows:

$$CC_w(\vec{x},\vec{y}) = R_{w-m}(\vec{x},\vec{y}), \quad w \in \{1,2,\ldots,2m-1\} \tag{3.4}$$

where $R_{w-m}(\vec{x},\vec{y})$ is computed, in turn, as:

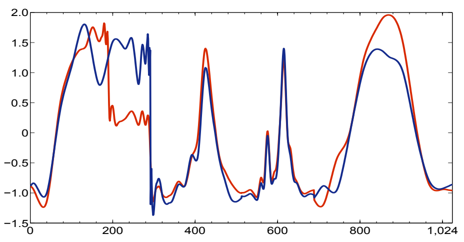
$$R_k(\vec{x},\vec{y}) = \begin{cases} \sum\limits_{l=1}^{m-k} x_{l+k} \cdot y_l, & k \geq 0 \\ R_{-k}(\vec{y},\vec{x}), & k < 0 \end{cases} \tag{3.5}$$

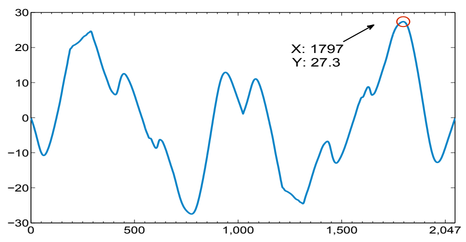
Our goal is to compute the position w at which $CC_w(\vec{x},\vec{y})$ is maximized. Based on this value of w, the optimal shift of $\vec{x}$ with respect to $\vec{y}$ is then $\vec{x}_{(s)}$, where $s = m - w$.

Depending on the domain or the application, different normalizations for $CC_w(\vec{x},\vec{y})$ might be required. The most common normalizations are the biased estimator, NCC_b, the unbiased estimator, NCC_u, and the coefficient normalization, NCC_c, defined as follows:

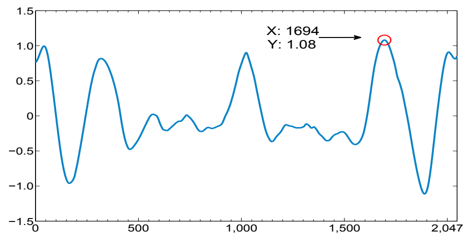
$$NCC_q(\vec{x},\vec{y}) = \begin{cases} \frac{CC_w(\vec{x},\vec{y})}{m}, & q = \text{“}b\text{”} \ (NCC_b) \\ \frac{CC_w(\vec{x},\vec{y})}{m-|w-m|}, & q = \text{“}u\text{”} \ (NCC_u) \\ \frac{CC_w(\vec{x},\vec{y})}{\sqrt{R_0(\vec{x},\vec{x}) \cdot R_0(\vec{y},\vec{y})}}, & q = \text{“}c\text{”} \ (NCC_c) \end{cases} \tag{3.6}$$

Beyond the cross-correlation normalizations, time series might also require normalization to remove inherent distortions. Figure 3.2 illustrates how the cross-correlation normalizations for two sequences $\vec{x}$ and $\vec{y}$ of length $m = 1024$ are affected by time-series normalizations. (Appendix A elaborates on the classification accuracy of cross-correlation variants under other time-series normalizations.) Independently of the normalization applied to $CC_w(\vec{x},\vec{y})$, the produced sequence will have length 2047. Initially, in Figure 3.2a, we remove differences in amplitude by z-normalizing $\vec{x}$ and $\vec{y}$ in order to show that they are aligned and, hence, no shifting is required. If $CC_w(\vec{x},\vec{y})$ is maximized for $w \in [1025, 2047]$ (or $w \in [1, 1023]$), the $\vec{x}$ sequence should be shifted by $w - 1024$ to the left (or $1024 - w$ to the right). Otherwise,

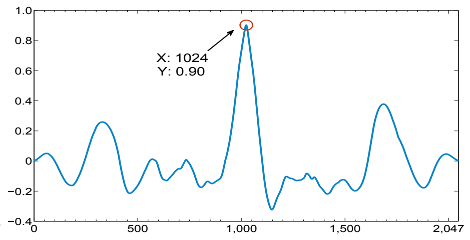

(a) z-normalized time series



(b) NCC_b (no z-normalization)



(c) NCC_u with z-normalization



(d) NCC_c with z-normalization

Figure 3.2: Time-series and cross-correlation normalizations.

if $w = 1024$, $\vec{x}$ and $\vec{y}$ are properly aligned, which is what we expect in our example. Figure 3.2b shows that if we do not z-normalize $\vec{x}$ and $\vec{y}$, and we use the biased estimator, then NCC_b is maximized at $w = 1797$, which indicates a shifting of the $\vec{x}$ sequence to the left $1797 - 1024 = 773$ times. If we z-normalize $\vec{x}$ and $\vec{y}$, and use the unbiased estimator, then NCC_u is maximized at $w = 1694$, which indicates a shifting of the $\vec{x}$ sequence to the left $1694 - 1024 = 670$ times (Figure 3.2c). Finally, if we z-normalize $\vec{x}$ and $\vec{y}$, and use the coefficient normalization, then NCC_c is maximized at $w = 1024$, which indicates that no shifting is required (Figure 3.2d).

As illustrated by the example above, normalizations of the data and the cross-correlation measure can have a significant impact on the cross-correlation sequence produced, which makes the creation of a distance measure a non-trivial task. Furthermore, as we have seen in Figure 3.2, cross-correlation sequences produced by pairwise comparisons of time series will differ in amplitude based on the normalizations. Thus, a normalization that produces values within a specified range should be used in order to meaningfully compare such sequences.

Shape-based distance (SBD): To devise a shape-based distance measure, and based on the previous discussion, we use the coefficient normalization that gives values between -1 and 1, regardless of the data normalization. Coefficient normalization divides the cross-correlation sequence by the geometric mean of autocorrelations of the individual sequences. After normalization of the sequence, we detect the position w where $NCC_c(\vec{x},\vec{y})$ is maximized and we derive the following distance measure:

$$SBD(\vec{x},\vec{y}) = 1 - \max_{w}\left(\frac{CC_w(\vec{x},\vec{y})}{\sqrt{R_0(\vec{x},\vec{x}) \cdot R_0(\vec{y},\vec{y})}}\right) \tag{3.7}$$

which takes values between 0 to 2, with 0 indicating perfect similarity for time series.

Up to now we have addressed shift invariance. For scaling invariance, we transform each sequence $\vec{x}$ into $\vec{x}' = \frac{\vec{x}-\mu}{\sigma}$, so that its mean μ is zero and its standard deviation σ is one.

Efficient computation of SBD: From Equation 3.4, the computation of $CC_w(\vec{x},\vec{y})$ for all values of w requires $\mathcal{O}(m^2)$ time, where m is the time-series length. The convolution theorem [Katznelson, 2004] states that the convolution of two time series can be computed as the Inverse Discrete Fourier Transform (IDFT) of the product of the individual Discrete Fourier Transforms (DFT) of the time series, where DFT is:

$$\mathcal{F}(x_k) = \sum_{r=0}^{|\vec{x}|-1} x_r e^{\frac{-2jrk\pi}{|\vec{x}|}}, \quad k = 0,\dots,|\vec{x}|-1 \tag{3.8}$$

and IDFT is:

$$\mathcal{F}^{-1}(x_r) = \frac{1}{|\vec{x}|}\sum_{k=0}^{|\vec{x}|-1} \mathcal{F}(x_k) e^{\frac{2jrk\pi}{|\vec{x}|}}, \quad r = 0,\dots,|\vec{x}|-1 \tag{3.9}$$

where $j = \sqrt{-1}$. Cross-correlation is then computed as the convolution of two time series if one sequence is first reversed in time, $\vec{x}^{(t)} = \vec{x}^{(-t)}$ [Katznelson, 2004], which equals taking the complex conjugate (represented by $*$) in the frequency domain. Thus, Equation 3.4 can be computed for every m as:

$$CC(\vec{x},\vec{y}) = \mathcal{F}^{-1}\{\mathcal{F}(\vec{x}) * \mathcal{F}(\vec{y})\} \tag{3.10}$$

However, DFT and IDFT still require $\mathcal{O}(m^2)$ time. By using a Fast Fourier Transform (FFT) algorithm [Cooley and Tukey, 1965], the time reduces to $\mathcal{O}(m \cdot \log(m))$. Data and

Algorithm 1: $[dist, y'] = SBD(x, y)$

```
Input: Two $z$-normalized sequences $x$ and $y$
Output: Dissimilarity $dist$ of $x$ and $y$
        Aligned sequence $y'$ of $y$ towards $x$
1  $length = 2^{nextpower2(2*length(x)-1)}$
2  $CC = IFFT\{FFT(x, length) * FFT(y, length)\}$        // Equation 3.10
3  $NCC_c = \frac{CC}{||x||||y||}$                         // Equation 3.6
4  $[value, index] = max(NCC_c)$
5  $dist = 1 - value$                                      // Equation 3.7
6  $shift = index - length(x)$
7  if $shift \geq 0$ then
8      $y' = [zeros(1, shift), y(1 : end - shift)]$        // Equation 3.3
9  else
10     $y' = [y(1 - shift : end), zeros(1, -shift)]$       // Equation 3.3
```

cross-correlation normalizations can also be efficiently computed; thus the overall time complexity of SBD remains $\mathcal{O}(m \cdot \log(m))$. Importantly, by exploiting intrinsic characteristics of FFT algorithms, we can further improve the performance of SBD. Specifically, recursive algorithms compute an FFT by dividing it into pieces of power-of-two size [Frigo and Johnson, 2005]. Therefore, when $CC(\vec{x},\vec{y})$ is not an exact power of two we pad $\vec{x}$ and $\vec{y}$ with zeros to reach the next power-of-two length after $2m-1$. This important property of FFT algorithms, which is not often exploited in the literature (e.g., [Mueen et al., 2014; Zakaria et al., 2012]), leads to a significant improvement in the performance of SBD, as we show in Section 3.6.1. Algorithm 1 outlines how this efficient and parameter-free measure is computed in a few lines of code using modern mathematical software.

In this section, we showed effective cross-correlation and data normalizations to derive a shape-based distance measure. Importantly, we also discussed how cross-correlation can be efficiently computed. In our experimental evaluation (Sections 3.5 and 3.6), we will show that SBD is highly competitive, achieving similar results to cDTW while being significantly faster. We now turn to the critical problem of extracting cluster centroids, to represent the cluster data consistently with the shape-based distance measure described above.

3.3.2 Time-Series Shape Extraction

Many tasks in time-series analysis rely on methods that effectively summarize a set of time series by a small number of sequences and, in some cases, by only one sequence. These summary sequences are often referred to as *average sequences* or, in the context of clustering, *centroids*. The extraction of meaningful centroids is a challenging task that critically depends on the choice of distance measure (see Section 3.2.1). We now show how to determine such centroids for time-series clustering for SBD, to capture shared characteristics of the underlying data. Specifically, we present ShapeExtraction, a method that extracts a single centroid to summarize an entire set of time series, and MultiShapesExtraction, a method that extracts multiple centroids to summarize an entire set of time series.

ShapeExtraction (SE): The easiest way to extract an average sequence from a set of sequences is to compute each coordinate of the average sequence as the arithmetic mean of the corresponding coordinates of all sequences. This approach is used by k-means, the most popular clustering method. In Figure 3.3, the solid line shows the centroid for "Class A" in the ECGFiveDays dataset of Figure 3.1: these centroids do not offer invariances to scaling and shifting and, therefore, such centroids do not capture effectively the class characteristics.

To avoid such problems, we cast the centroid computation as an optimization problem where the objective is to find the minimizer of the sum of squared distances to all other time series sequences (Equation 3.2). However, as cross-correlation intuitively captures the similarity — rather than the dissimilarity — of time series, we can express the computed sequence as the maximizer $\mu_k^\star$ of the squared similarities to all other time-series sequences. By rewriting Equation 3.2 as a maximization problem and from Equation 3.6, we obtain:

$$\begin{aligned}\vec{\mu_k}^\star &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \sum_{\vec{x_i} \in P_k} NCC_c(\vec{x_i}, \vec{\mu_k})^2 \\ &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \sum_{\vec{x_i} \in P_k} \left(\max_w \frac{CC_w(\vec{x_i}, \vec{\mu_k})}{\sqrt{R_0(\vec{x_i}, \vec{x_i}) \cdot R_0(\vec{\mu_k}, \vec{\mu_k})}} \right)^2 \end{aligned} \tag{3.11}$$

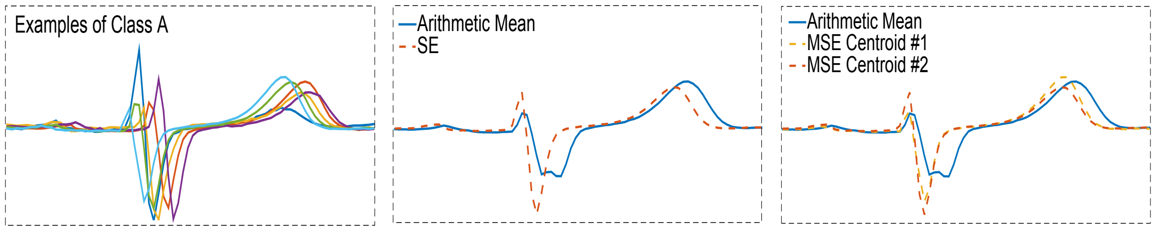


Figure 3.3: Examples of "Class A" sequences of the ECGFiveDays dataset and centroids based on the arithmetic mean property (solid lines) and our SE and MSE methods (dashed lines).

Equation 3.11 requires the computation of an optimal shift for every $\vec{x_i} \in P_k$. As this approach is used in the context of iterative clustering, we use the previously computed centroid as reference and align all sequences towards this reference sequence.[3]

This is a reasonable choice because the previous centroid will be very close to the new centroid. For this alignment, we use SBD, which identifies an optimal shift for every $\vec{x_i} \in P_k$. Subsequently, as sequences are already aligned towards a reference sequence before the centroid computation, we can also omit the denominator of Equation 3.11. Then, by combining Equations 3.4 and 3.5, we obtain:

$$\vec{\mu_k}^\star = \underset{\vec{\mu_k}}{\operatorname{argmax}} \sum_{\vec{x_i} \in P_k} \left(\sum_{l \in [1,m]} x_{il} \cdot \mu_{kl} \right)^2$$

For simplicity, we express this equation with vectors and assume that the $\vec{x_i}$ sequences have already been z-normalized to handle the differences in amplitude:

$$\begin{aligned}
\vec{\mu_k^\star} &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \sum_{\vec{x_i} \in P_k} (\vec{x_i}^T \cdot \vec{\mu_k})^2 \\
&= \underset{\vec{\mu_k}}{\operatorname{argmax}} \sum_{\vec{x_i} \in P_k} \vec{x_i}^T \cdot \vec{\mu_k} \cdot \vec{x_i}^T \cdot \vec{\mu_k} \\
&= \underset{\vec{\mu_k}}{\operatorname{argmax}} \, \vec{\mu_k}^T \cdot \sum_{\vec{x_i} \in P_k} (\vec{x_i} \cdot \vec{x_i}^T) \cdot \vec{\mu_k} \qquad (3.12)
\end{aligned}$$

In the previous equation, only $\vec{\mu_k}$ is not z-normalized. To handle the centering (i.e., the

[3]For simplicity, we consider sequences of equal length. In cases where sequences are of different length, we first pad with zeros all sequences to reach the length of the longest sequence.

```
Algorithm 2: $C = SE(X,R)$
  Input: $X$ is an $n$-by-$m$ matrix with $z$-normalized time series.
         $R$ is a 1-by-$m$ vector with the reference sequence against which time series of $X$
         are aligned.
  Output: $C$ is a 1-by-$m$ vector with the centroid.
1 $X' \leftarrow [\ ]$
2 for $i \leftarrow 1$ to $n$ do
3     $[dist, x'] \leftarrow SBD(R, X(i))$                    // Algorithm 1
4     $X' \leftarrow [X'; x']$
5 $S \leftarrow X'^T \cdot X'$                                // S of Equation 3.13
6 $Q \leftarrow I - \frac{1}{m} \cdot O$                       // Q of Equation 3.13
7 $M \leftarrow Q^T \cdot S \cdot Q$                           // M of Equation 3.13
8 $C \leftarrow Eig(M, 1)$                          // Extract first eigenvector
```

subtraction of mean) of $\vec{\mu_k}$ we set $\vec{\mu_k} = \vec{\mu_k} \cdot Q$, where $Q = I - \frac{1}{m}O$, I is the identity matrix, and O is a matrix with all ones. Furthermore, to make $\vec{\mu_k}$ have unit norm, we divide Equation 3.12 by $\vec{\mu_k}^T \cdot \vec{\mu_k}$. (We omit the division with the standard deviation as such step will not alter the results.) Finally, by substituting S for $\sum_{\vec{x_i} \in P_k} (\vec{x_i} \cdot \vec{x_i}^T)$, we obtain:

$$\begin{aligned} \vec{\mu_k}^{\star} &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \frac{\vec{\mu_k}^T \cdot Q^T \cdot S \cdot Q \cdot \vec{\mu_k}}{\vec{\mu_k}^T \cdot \vec{\mu_k}} \\ &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \frac{\vec{\mu_k}^T \cdot M \cdot \vec{\mu_k}}{\vec{\mu_k}^T \cdot \vec{\mu_k}} \end{aligned} \tag{3.13}$$

where $M = Q^T \cdot S \cdot Q$. Through the above transformations, we have reduced the optimization of Equation 3.11 to the optimization of Equation 3.13, which is a well-known problem called maximization of the Rayleigh Quotient [Golub and Van Loan, 2012]. We can find the maximizer $\vec{\mu_k}^{\star}$ as the eigenvector that corresponds to the largest eigenvalue of matrix M.

Algorithm 2 shows how we can extract the most representative shape from the underlying data in a few lines of code. As a first step, SE aligns all sequences towards a reference sequence C. Then, SE computes matrix M (as shown in Equation 3.13) and, subsequently, extracts the primary eigenvector of M, which corresponds to the sequence with the maximum similarity to all others sequences. In Figure 3.3, we show the centroid of "Class A" in the ECGFiveDays dataset, extracted with SE and using a randomly selected sequence as reference sequence. SE method can more effectively capture the characteristics of "Class

A" (Figure 3.1) than by using the arithmetic mean property (solid lines in Figure 3.3).

MultiShapesExtraction (MSE): Similarly to all averaging techniques described in Section 3.2.3, SE is based on two important assumptions: (i) a single centroid can effectively represent the characteristics of the underlying time series and (ii) all sequences contribute equally to the computation of the most representative sequence. However, in practice, time series are not uniformly distributed in space. Instead, some time series might appear in areas of higher density than the remainder of the time series. Therefore, a single centroid might not be sufficient to effectively capture characteristics of time series in clusters where time series are not uniformly distributed. Additionally, noisy sequences and outliers complicate the extraction of a representative sequence. Importantly, such noisy sequences and outliers do not always appear more distant to a reference sequence than the remainder of the sequences. Therefore, a weighted contribution of sequences in the computation of the most representative sequence is not sufficient.

To avoid the above limitations, we present MSE, a method to extract multiple centroids to summarize an entire set of time series. MSE relies on SE to extract each individual centroid but with two important differences. First, during the process of alignment of sequences towards a reference sequence, MSE computes the proximity of sequences to the reference sequence. Second, MSE divides the time series into evenly distributed "segments" by analyzing the proximity of sequences to a reference sequence. This division of time series into segments leads to the computation of multiple centroids, with each centroid capturing characteristics of segments with different proximity to the reference sequence. With this process, MSE solves the limitations of SE discussed above: (i) dense areas that might contain the majority of time series are now represented by multiple centroids and (ii) noisy sequences and outliers only influence the centroids for the segments where the sequences belong.

Algorithm 3 shows how we can extract multiple representative shapes from the underlying data in a few lines of code. As a first step, MSE aligns all sequences towards a reference sequence and computes the proximity of those sequences to the reference sequence

```
Algorithm 3: $C = MSE(X, R, L)$
Input: $X$ is an $n$-by-$m$ matrix with $z$-normalized time series.
       $R$ is a 1-by-$m$ vector with the reference sequence against which time series of
       $X$ are aligned.
       $L$ is the desired number of centroids per cluster.
Output: $C$ is an $L$-by-$m$ matrix with the $L$ centroids.
1  $Dists \leftarrow [\ ]$
2  $X' \leftarrow [\ ]$
3  for $i \leftarrow 1$ to $n$ do
4      $[dist, x'] \leftarrow SBD(R, X(i))$                    // Algorithm 1
5      $X' \leftarrow [X'; x']$
6      $Dists \leftarrow [Dists; dist]$
7  $V \leftarrow quantile(Dists, L)$
8  for $LIndex \leftarrow 1$ to $L$ do
9      $NewX' \leftarrow []$
10     for $i \leftarrow 1$ to $n$ do
11         if $mod(LIndex, L) = 0$ then
12             $NewX' \leftarrow [NewX'; X(i)']$
13         else
14             if $Dists(i) \leq V(mod(LIndex, L))$ then
15                 $NewX' \leftarrow [NewX'; X(i)']$
16     $S \leftarrow NewX'^T \cdot NewX'$                       // S of Equation 3.13
17     $Q \leftarrow I - \frac{1}{m} \cdot O$                    // Q of Equation 3.13
18     $M \leftarrow Q^T \cdot S \cdot Q$                       // M of Equation 3.13
19     $C(LIndex) \leftarrow Eig(M, 1)$                         // Extract first eigenvector
```

(lines 3-6). MSE analyzes the proximity information and computes L quantiles that divide the data set into $L+1$ evenly distributed segments. Vector V contains distance values that correspond to L evenly spaced cumulative probabilities $\frac{1}{L+1}, \frac{2}{L+1}, \ldots, \frac{L}{L+1}$. Then, MSE computes centroids as follows. Each centroid c_j, with $j = 1, \ldots, L-1$, covers sequences with distances less than $V(\text{mod}(j,\ L))$, where mod denotes the modulo operation; and the last centroid, namely, centroid c_L, covers all sequences. Finally, similarly to SE, MSE computes matrix M (as shown in Equation 3.13) for each subset of sequences and, subsequently, extracts the primary eigenvector of M, which corresponds to the sequence with the maximum similarity to all other sequences in the subset of sequences. In Figure 3.3, we show two centroids of "Class A" in ECGFiveDays, extracted with MSE and using a randomly selected sequence as reference sequence. MSE can more effectively capture the characteristics of "Class A" (Figure 3.1) than by using either SE or the arithmetic mean property (solid lines in Figure 3.3).

We now show how SE and MSE are used in time-series clustering algorithms.

3.3.3 Shape-based Time-Series Clustering Methods

In this section, we present k-Shape and k-MS, our novel algorithms for time-series clustering. We first describe k-Shape, which relies on the SBD distance measure (Section 3.3.1) and SE for centroid computation (Section 3.3.2) to efficiently produce clusters of time series.

k-Shape Clustering Algorithm: k-Shape is a partitional clustering method that is based on an iterative refinement procedure similar to the one used in k-means. Through this iterative procedure, k-Shape minimizes the sum of squared distances (Equation 3.1) and manages to: (i) produce homogeneous and well-separated clusters, and (ii) scale linearly with the number of time series. k-Shape is a nontrivial instantiation of k-means that compares sequences efficiently and computes centroids effectively under the scaling, translation, and shift invariances. As we will see, the choice of distance measure and centroid computation method make k-Shape the only scalable method that significantly outperforms k-means.

In every iteration, k-Shape performs two steps: (i) in the assignment step, k-Shape updates the cluster memberships by comparing each time series with all computed centroids and by assigning each time series to the cluster of the closest centroid; (ii) in the refinement step, the cluster centroids are updated using the SE method to reflect the changes in cluster memberships in the previous step. k-Shape repeats these two steps until either no change in cluster membership occurs or the maximum number of iterations allowed is reached.

k-Shape (see Algorithm 4) expects as input the time series set X and the number of clusters k that we want to produce. Initially, we randomly assign the time series in X to clusters. Then, we compute each cluster centroid using Algorithm 2 (lines 5-10). Once the centroids are computed, we refine the memberships of the clusters by using SBD (Algorithm 1) in lines 11-17. We repeat this procedure until the algorithm converges or

```
Algorithm 4: $[IDX, C] = k\text{-}Shape(X, k)$
Input: $X$ is an $n$-by-$m$ matrix containing $n$ time series of length $m$ that are initially
       $z$-normalized.
       $k$ is the number of clusters to produce.
Output: $IDX$ is an $n$-by-1 vector with the assignment of $n$ time series to $k$ clusters
        (initialized randomly).
        $C$ is a $k$-by-$m$ matrix with $k$ centroids of length $m$ (initialized as vectors
with all zeros).
1  $iter \leftarrow 0$
2  $IDX' \leftarrow [\ ]$
3  while $IDX! = IDX'$ and $iter < 100$ do
4      $IDX' \leftarrow IDX$
       // Refinement step
5      for $j \leftarrow 1$ to $k$ do
6          $X' \leftarrow [\ ]$
7          for $i \leftarrow 1$ to $n$ do
8              if $IDX(i) = j$ then
9                  $X' \leftarrow [X'; X(i)]$
10         $C(j) \leftarrow SE(X', C(j))$                          // Algorithm 2
       // Assignment step
11     for $i \leftarrow 1$ to $n$ do
12         $mindist \leftarrow \infty$
13         for $j \leftarrow 1$ to $k$ do
14             $[dist, x'] \leftarrow SBD(C(j), X(i))$             // Algorithm 1
15             if $dist < mindist$ then
16                 $mindist \leftarrow dist$
17                 $IDX(i) \leftarrow j$
18     $iter \leftarrow iter + 1$
```

reaches the maximum number of iterations (usually a small number, such as 100). The output of the algorithm is the assignment of sequences to clusters and the centroids for each cluster.

Having described k-Shape, we now present k-MS, which relies on the SBD distance measure (Section 3.3.1) and MSE for centroid computation (Section 3.3.2).

k-MS Clustering Algorithm: k-MS relies on a similar iterative refinement procedure to the one used for k-Shape. In every iteration, k-MS performs two steps: (i) in the assignment step, k-MS compares each time series to all computed centroids per cluster, assigns each time series to the closest centroid, and updates cluster memberships based on the memberships of centroids that belong to the same cluster; (ii) in the refinement step, the cluster centroids are updated using the MSE method to reflect the changes in cluster memberships in the previous step. k-MS repeats these two steps until either no change in cluster membership occurs or the maximum number of iterations allowed is reached.

k-MS (see Algorithm 5) expects three input parameters: (i) the time series set X; (ii) the number of clusters k; and (iii) the number of centroids L per cluster that we want to produce. In contrast to k-Shape, k-MS needs to handle multiple centroids in each cluster after the refinement and the assignment steps. In particular, in the refinement step, k-MS extracts multiple centroids for each cluster and, subsequently, k-MS appends these centroids in appropriate positions in the global list of centroids (lines 15-16). This step is critical for k-MS and allows the computation of centroids in each cluster with the same reference sequence. In the assignment step, k-MS performs an additional step not present in k-Shape: after the assignment of each time series to a centroid, k-MS merges the memberships of centroids that belong to the same cluster (lines 24-25).

Complexity of k-Shape and k-MS: As we claimed earlier, k-Shape and k-MS scale linearly with the number of time series. To see why, we will analyze the computational complexity of Algorithms 4 and 5, where n is the number of time series, k is the number of clusters, L is the number of centroids per cluster, and m is the length of the time series. In the assignment step, k-Shape computes the dissimilarity of n time series to k centroids by using SBD, which requires $\mathcal{O}(m \cdot \log(m))$ time. Thus, the time complexity of this step is $\mathcal{O}(n \cdot k \cdot m \cdot \log(m))$. k-MS computes the dissimilarity of n time series to $k \cdot L$ centroids and, therefore, the time complexity of this step is $\mathcal{O}(n \cdot k \cdot L \cdot m \cdot \log(m))$. In the refinement step, for every cluster, k-Shape computes matrix M, which requires $\mathcal{O}(m^2)$ time, and performs an eigenvalue decomposition on M, which requires $\mathcal{O}(m^3)$ time. Thus, the complexity of this step is $\mathcal{O}(\max\{n \cdot m^2, k \cdot m^3\})$. k-MS performs the same procedure L times per cluster and, therefore, the complexity of this step for k-MS is $\mathcal{O}(\max\{n \cdot m^2, k \cdot L \cdot m^3\})$. Overall, k-Shape requires $\mathcal{O}(\max\{n \cdot k \cdot m \cdot \log(m), n \cdot m^2, k \cdot m^3\})$ time per

Algorithm 5: $[IDX, C] = k\text{-}MS(X, k, L)$

```
Input: $X$ is an $n$-by-$m$ matrix containing $n$ time series of length $m$ that are initially
        $z$-normalized.
        $k$ is the number of clusters to produce.
        $L$ is the number of centroids per cluster.
Output: $IDX$ is an $n$-by-1 vector with the assignment of $n$ time series to $k$ clusters
        (initialized randomly).
        $C$ is a $k \cdot L$-by-$m$ matrix containing $k \cdot L$ centroids of length $m$ (initialized as
vectors with all zeros).
1  $iter \leftarrow 0$
2  $IDX' \leftarrow [\ ]$
3  $[IDXtemp, Ctmp] \leftarrow k - Shape(X, k)$                          // Algorithm 4
4  for $j \leftarrow 1$ to $k$ do
5      for $w \leftarrow 1$ to $L$ do
6          $C(w + (j-1) \cdot L) \leftarrow Ctmp(j)$
7  while $IDX! = IDX'$ and $iter < 100$ do
8      $IDX' \leftarrow IDX$
       // Refinement step
9      for $j \leftarrow 1$ to $k$ do
10         $X' \leftarrow [\ ]$
11         for $i \leftarrow 1$ to $n$ do
12             if $IDX(i) = j$ then
13                 $X' \leftarrow [X'; X(i)]$
14         $Ctmp \leftarrow MSE(X', C(j \cdot L)), L)$                  // Algorithm 3
15         for $w \leftarrow 1$ to $L$ do
16             $C(w + (j-1) \cdot L) \leftarrow Ctmp(w)$
       // Assignment step
17     for $i \leftarrow 1$ to $n$ do
18         $mindist \leftarrow \infty$
19         for $j \leftarrow 1$ to $k \cdot L$ do
20             $[dist, x'] \leftarrow SBD(C(j), X(i))$                  // Algorithm 1
21             if $dist < mindist$ then
22                 $mindist \leftarrow dist$
23                 $IDX(i) \leftarrow j$
24     for $i \leftarrow 1$ to $n$ do
25         $IDX(i) \leftarrow ceil(IDX(i)/L)$
26     $iter \leftarrow iter + 1$
```

iteration and k-MS requires $\mathcal{O}(\max\{n \cdot k \cdot L \cdot m \cdot \log(m), n \cdot m^2, k \cdot L \cdot m^3\})$. We see that both algorithms have a linear dependence in the number of time series, and the majority of the computation cost depends on the length of the time series. However, this length is usually much smaller than the number of time series (i.e., $m \ll n$) and, hence, the dependence on m is not a bottleneck. (Appendix B further examines the scalability of k-Shape.) In rare cases where m is very large (i.e., $m \gg n$), segmentation or dimensionality reduction approaches can be used to sufficiently reduce the length of the sequences [Chan et al., 2003; Lin et al., 2004].

3.4 Exploiting Clustering for Faster One-Nearest-Neighbor Classification

So far, we have focused on how to perform shape-based clustering of time series effectively. However, as noted earlier, clustering is not only useful as a powerful standalone method, but also as a preprocessing or subroutine for other related tasks over time series. In this section, we show how k-Shape, our simplest shape-based clustering method, can effectively reduce the search space of one-nearest-neighbor algorithms and lead to fast and accurate techniques for the important problem of classification of time series.

One-nearest-neighbor, or 1-NN, classifiers address the time-series classification problem by assigning each time series to the category or class of the "nearest" time series according to a distance measure. 1-NN classifiers are popular because they do not rely on tunable parameters, and also because they can be used in conjunction with an appropriate choice of distance measure, out of the vast array of options (see Section 2.3 for examples of distance measures). Furthermore, 1-NN classifiers perform exceptionally well for time-series classification when used in conjunction with competitive distance measures for time series, as shown in [Bagnall and Lines, 2014; Lines and Bagnall, 2014; Lines and Bagnall, 2015; Bagnall et al., 2016].

Reduction of dimensionality, auxiliary index structures, and lower bounding of distance measures have been previously exploited to reduce the computational time of 1-NN classifiers for time series [Agrawal et al., 1993; Faloutsos et al., 1994; Rakthanmanon et al., 2012]. Alternative methods for 1-NN classification, known as k-Nearest Centroid Classifiers (k-NCC), extract for each "query" the k nearest neighbors from each class in the search space, and then summarize each of the sets of k neighbors with a centroid that is computed as some

Algorithm 6: $Accuracy = NNC(Training, Test, TrainingLabels, TestLabels, LeaveOneOut)$

```
Input: $Training$ is a $k$-by-$m$ matrix containing $k$ time series of length $m$.
       $Test$ is an $n$-by-$m$ matrix containing $n$ time series of length $m$.
       $TrainingLabels$ is a 1-by-$k$ vector with the class labels of $k$ time series in $Training$.
       $TestLabels$ is a 1-by-$n$ vector with the class labels of $n$ time series in $Test$.
       $LeaveOneOut$ is a Boolean indicating if this is a leave-one-out classification (true) or
     one-nearest-neighbor classification (false).
Output: $Accuracy$ is the classification accuracy.
1  $Accuracy \leftarrow 0$
2  for $i \leftarrow 1$ to $n$ do
3      $best\_dist \leftarrow Inf$
4      for $j \leftarrow 1$ to $k$ do
5          if $LeaveOneOut = true$ then
6              if $i! = j$ then
7                  $[dist, \sim] \leftarrow SBD(Training(j), Test(i))$          // Algorithm 1
8          else
9              $[dist, \sim] \leftarrow SBD(Training(j), Test(i))$              // Algorithm 1
10         if $dist < best\_dist$ then
11             $class \leftarrow TrainingLabels(j)$
12             $best\_dist \leftarrow dist$
13     if $TestLabels(i) = class$ then
14         $Accuracy \leftarrow Accuracy + 1$
15 $Accuracy \leftarrow Accuracy/n$
```

variant of a mean vector. k-NCC methods determine the class of the query using factors such as the proximity and the spatial distribution of the computed centroids relative to the query [Mitani and Hamamoto, 2000; Mitani and Hamamoto, 2006; Chaudhuri, 1996; Sánchez et al., 1997; Gou et al., 2012]. However, k-NCC methods are usually more expensive than 1-NN classifiers because of the additional processing that is required beyond nearest-neighbor search. Interestingly, these approaches reduce to 1-NN when $k = 1$. Recently, [Petitjean et al., 2014; Petitjean et al., 2016] demonstrated the potential of representing each class in the search space with a small number of centroids computed via clustering. However, the clustering methods that they use are particularly inefficient, as we show in Section 3.6, and they significantly increase the cost to estimate an appropriate number of centroids to represent each class in the search space.

To address this issue, and inspired by [Petitjean et al., 2014; Petitjean et al., 2016], we now introduce the Nearest Shape Classifier (NSC), a 1-NN classifier that relies on k-Shape

Algorithm 7: *Accuracy* = *NSC*(*Train*, *Test*, *TrainLabels*, *TestLabels*, *InstancesLabels*)

```
Input: $Train$ is a $k$-by-$m$ matrix containing $k$ time series of length $m$.
       $Test$ is an $n$-by-$m$ matrix containing $n$ time series of length $m$.
       $TrainLabels$ is a 1-by-$k$ vector with the class labels of the $k$ time series in $Train$.
       $TestLabels$ is a 1-by-$n$ vector with the class labels of the $n$ time series in $Test$.
       $InstancesLabels$ is a 1-by-$w$ vector with each cell pointing to a list with IDs of
       instances in $Train$ belonging to each of the $w$ classes.
Output: $Accuracy$ is a scalar value containing the classification accuracy on $Test$.
1  $Best\_Accuracy \leftarrow NNC(Train, Train, TrainLabels, TrainLabels, true)$   // Algorithm 6
2  $Best\_NumberCentroids \leftarrow k$
3  $Best\_Centroids \leftarrow Train$
4  $Best\_CentroidsLabels \leftarrow TrainLabels$
5  $Centroids \leftarrow []$
6  $CentroidsLabels \leftarrow []$
7  for $i \leftarrow 1$ to $max(length(InstancesLabels([1:w]))) - 1$ do
8      for $class \leftarrow 1$ to $w$ do
9          $IDs \leftarrow InstancesLabels(class)$
10         $[nrows, ncolumns] \leftarrow size(Train(IDs))$
11         if $i = 1$ and $nrows = 1$ then
12             $tmp\_centroids \leftarrow Train(IDs)$
13             $Centroids \leftarrow [Centroids; tmp\_centroids]$
14             $CentroidsLabels \leftarrow [CentroidsLabels; class]$
15         else if $i = 1$ and $nrows > 1$ then
16             $index \leftarrow rand(IDs)$
17             $ref\_centroid \leftarrow Train(index)$
18             $tmp\_centroids \leftarrow SE(Train(IDs), ref\_centroid)$   // Algorithm 2
19             $Centroids \leftarrow [Centroids; tmp\_centroids]$
20             $CentroidsLabels \leftarrow [CentroidsLabels; class]$
21         else if $i > 1$ and $nrows > i$ then
22             $[\sim, tmp\_centroids] \leftarrow k-Shape(Train(IDs), i)$   // Algorithm 4
23             $Centroids \leftarrow [Centroids; tmp\_centroids]$
24             for $p \leftarrow 1$ to $nrows$ do
25                 $CentroidsLabels \leftarrow [CentroidsLabels; class]$
26         else if $i > 1$ and $nrows \leq i$ then
27             $Centroids \leftarrow [Centroids; Train(IDs)]$
28             for $p \leftarrow 1$ to $nrows$ do
29                 $CentroidsLabels \leftarrow [CentroidsLabels; class]$
30     $Acc \leftarrow NNC(Centroids, Train, CentroidsLabels, TrainLabels, true)$   // Algorithm 6
31     if $Acc > Best\_Accuracy$ then
32         $Best\_Accuracy \leftarrow AccuracyOnCentroids$
33         $Best\_NumberCentroids \leftarrow size(Centroids, 1)$
34         $Best\_Centroids \leftarrow Centroids$
35         $Best\_CentroidsLabels \leftarrow CentroidsLabels$
36     $Centroids \leftarrow []$
37     $CentroidsLabels \leftarrow []$
38 $Accuracy \leftarrow NNC(Best\_Centroids, Test, Best\_CentroidsLabels, TestLabels, false)$
       // Algorithm 6
```

to effectively summarize time series in its search space (see Algorithm 7). In contrast to k-NCC methods, NSC uses cluster centroids to summarize the entire search space and does not recompute the centroids for every query. The estimation of the number of centroids relies solely on analysis over the search space of the 1-NN classifier. Specifically, NSC first determines how accurately it can estimate the class for queries taken from the search space (line 1). This accuracy serves as a baseline estimation of the best achieved accuracy when the entire search space is used. Then, NSC varies the number of centroids required to summarize each class in the search space (line 7). NSC computes centroids for each class in the search space by considering the following four cases: (i) when one centroid is requested and the class contains one time series, NSC uses this time series as centroid (lines 11-14); (ii) when one centroid is requested and the class contains more than one time series, NSC uses Algorithm 2 in Section 3.3.2 to summarize the time series (lines 15-20); (iii) when two or more centroids are requested and the class contains more time series than the requested centroids, NSC uses Algorithm 4 of Section 3.3.3 to cluster and summarize the time series (lines 21-25); and (iv) when two or more centroids are requested and the class contains fewer time series than the requested centroids, NSC uses the time series as centroids and does not perform clustering, to avoid producing singleton clusters with just one time series in them (lines 26-29). For each number of centroids, NSC evaluates how accurately it can estimate the class of the queries in the search space when the search space is summarized by that number of centroids. The process terminates if this accuracy outperforms the baseline accuracy computed earlier (in line 1). In the end, NSC determines the class of a query as the class of the nearest centroid.

The above procedure permits the estimation of the number of centroids and the computation of centroids that capture internal characteristics of every class of time series accurately, as we will see in Section 3.6.7. NSC requires as input the instances and the labels of the training and test sets along with a breakdown of training instances per class. As output, NSC computes the 1-NN classification accuracy (Algorithm 6) on the test set but using as search space the estimated number of centroids per class over the training set.

We now turn to the description of the settings of our in-depth experimental evaluation of SBD, k-Shape, k-MS, and NSC against the state-of-the-art time-series distance measures, clustering algorithms, and classification algorithms.

3.5 Experimental Settings

In this section, we review in detail the settings for the evaluation of SBD, k-Shape, k-MS, and NSC.

Datasets: We use the largest public collection of class-labeled time-series datasets, namely, the UCR collection [Keogh et al., 2015]. It consists of 85 datasets, both synthetic and real, which span several different domains. (Appendix C provides the exact names of the 85 datasets used in our analysis along with some relevant characteristics.) Each dataset contains from 40 to $16,637$ sequences. The sequences in each dataset have equal length, but from one dataset to another the sequence length varies from 24 to $2,709$. These datasets are annotated and every sequence can belong to only one class. In the context of clustering, and for the sake of convenience, the class label for a sequence is often interpreted as identifying the cluster where the sequence belongs. Furthermore, the datasets are already z-normalized and split into training and test sets. As we will see, we use this split of the datasets for the distance measure evaluation; we also use the training sets for tuning some of the baselines.

Platform: We ran our experiments on a cluster of 61 servers with identical configuration: Dual Intel Xeon E5-2650 (8-core with 2-way SMT) processor with clock speed at 2.6 GHz and up to 256 GB RAM. We utilized on average 20 servers continuously for a period of two months in order to perform all experiments included in this chapter. Each server runs Red Hat Enterprise Linux 6.6 (64-bit) and Matlab R2014a (64-bit).

Implementation: We implemented all approaches under the same framework, in Matlab, for a consistent evaluation in terms of both accuracy and efficiency. For repeatability purposes, we make all datasets and source code available.[4]

[4]`http://www.cs.columbia.edu/~jopa/kshape.html`

Baselines: We evaluate SBD, our distance measure; k-Shape and k-MS, our clustering approaches; and NSC, our one-nearest neighbor method. Specifically, we compare SBD against the strongest state-of-the-art distance measures for time series (see Section 2.3):

- **ED:** a simple, efficient — yet accurate — distance measure [Faloutsos et al., 1994]
- **DTW:** one of the best performing — but expensive — distance measure for time series [Sakoe and Chiba, 1978]
- **cDTW:** the constrained version of DTW, with improved accuracy and efficiency [Sakoe and Chiba, 1978]

We compare k-Shape and k-MS against the three strongest types of scalable and non-scalable clustering methods, namely, partitional, hierarchical, and spectral methods (see Section 3.2.2 for a detailed discussion), combined with the most competitive distance measures. As scalable methods, we consider the classic k-means algorithm with ED (k-AVG+ED) [MacQueen, 1967] and the following variants of k-means:

- **k-AVG+Dist:** k-means with DTW and SBD as distance measures and the arithmetic mean of time series coordinates for centroid computation
- **k-DBA:** k-means with DTW as distance measure and the DBA method for centroid computation [Petitjean et al., 2011]
- **KSC:** k-means with a distance measure offering pairwise scaling and shifting of time series and computation of the spectral norm of a matrix (i.e., matrix decomposition) for centroid computation [Yang and Leskovec, 2011]

As non-scalable methods, among partitional methods we consider the Partitioning Around Medoids (PAM) implementation of the k-medoids algorithm [Kaufman and Rousseeuw, 2009]. Among hierarchical methods, we use agglomerative hierarchical clustering with single, average, and complete linkage criteria [Kaufman and Rousseeuw, 2009]. Among spectral methods, we consider the popular normalized spectral clustering method [Ng et al., 2002]. These non-scalable approaches require a large number of distance calculations to compute the full dissimilarity matrix and, hence, they become unacceptably inefficient with high-cost distance measures. For this reason, we distribute the computation of the full dissimilarity matrices and, therefore, we do not report runtime results for these methods.

Beyond these scalable and non-scalable methods, we further evaluate k-Shape and k-MS against the strongest types of density-based methods, namely, DBSCAN [Ester et al., 1996] and TADPole [Begum et al., 2015]. Density-based methods have started to receive attention for time-series clustering to identify outliers in the data. DBSCAN can be combined with any distance measure, while TADPole, a variant of the recently proposed Density Peaks (DP) clustering method [Rodriguez and Laio, 2014], uses the cDTW distance. DBSCAN and TADPole are also non-scalable methods. Specifically, DBSCAN requires the computation of the full dissimilarity matrix to operate.[5] In contrast, TADPole significantly prunes the dissimilarity matrix for cDTW, but requires the computation of two additional dissimilarity matrices based on ED. TADPole improves in runtime performance relative to DP [Rodriguez and Laio, 2014], but remains non-scalable and particularly inefficient relative to k-means variants. Importantly, as we discuss later, both algorithms require the computation of the dissimilarity matrix for cDTW in order to accurately estimate input parameters, a step that is often omitted from runtime experiments. Therefore, both density-based methods are inefficient and, hence, we do not report runtime results for them.

We also compare k-Shape and k-MS against U-Shapelets [Zakaria et al., 2012]. Unlike previous approaches, U-Shapelets exploits patterns in subsequences of time series to isolate outliers. Unfortunately, U-Shapelets cannot handle large datasets, so we use a variant that has been shown to be faster than the original technique without a significant loss in accuracy [Begum et al., 2015]. Table 3.1 summarizes the non-scalable clustering combinations used in our evaluation. Overall, we compared k-Shape against 30 clustering approaches.

We compare NSC against data editing and condensing algorithms that are applied in the full dissimilarity matrix of the training instances of a 1-NN classifier and determine the order in which instances can be removed to guarantee the performance of a 1-NN classifier. Specifically, we evaluate NSC against three popular variants of such *reduction*

[5]DBSCAN can use auxiliary index structures to accelerate neighborhood queries only for metric distance functions, such as ED.

Name	Clustering Algorithm	Distance Measure
PAM+ED	Partitioning Around Medoids	ED
PAM+cDTW	Partitioning Around Medoids	$cDTW^5$
PAM+SBD	Partitioning Around Medoids	SBD
H-S+ED	Hierarchical with *single* linkage	ED
H-A+ED	Hierarchical with *average* linkage	ED
H-C+ED	Hierarchical with *complete* linkage	ED
H-S+cDTW	Hierarchical with *single* linkage	$cDTW^5$
H-A+cDTW	Hierarchical with *average* linkage	$cDTW^5$
H-C+cDTW	Hierarchical with *complete* linkage	$cDTW^5$
H-S+SBD	Hierarchical with *single* linkage	SBD
H-A+SBD	Hierarchical with *average* linkage	SBD
H-C+SBD	Hierarchical with *complete* linkage	SBD
S+ED	Normalized Spectral Clustering	ED
S+cDTW	Normalized Spectral Clustering	$cDTW^5$
S+SBD	Normalized Spectral Clustering	SBD
$DBSCAN^{0.3}$+ED	Density-based Spatial Clustering	ED
$DBSCAN^{Best}$+ED	Density-based Spatial Clustering	ED
$DBSCAN^{0.3}$+SBD	Density-based Spatial Clustering	SBD
$DBSCAN^{Best}$+SBD	Density-based Spatial Clustering	SBD
$DBSCAN^{0.3}$+cDTW	Density-based Spatial Clustering	$cDTW^5$
$DBSCAN^{Best}$+cDTW	Density-based Spatial Clustering	$cDTW^5$
$TADPole^{0.3}$	Time-series Anytime Density Peaks Clustering	$cDTW^5$
$TADPole^{Best}$	Time-series Anytime Density Peaks Clustering	$cDTW^5$
$U\text{-}Shapelets^{0.5}$	Unsupervised-Shapelets Clustering	-
$U\text{-}Shapelets^{Best}$	Unsupervised-Shapelets Clustering	-

Table 3.1: Combinations of PAM, hierarchical, spectral, density-based, shapelet-based methods with ED, cDTW, and SBD for our evaluation.

techniques, namely, RT1, RT2, and RT3 [Wilson and Martinez, 1997; Wilson and Martinez, 2000]. Additionally, we also compare NSC against a simple baseline, namely, Random, that randomly chooses instances for removal from the training instances. Finally, we compare the performance of NSC with three competitive distance measures (Sections 2.3 and 3.3.1), namely, ED (NSC+ED), SBD (NSC+SBD), and $cDTW^5$ (NSC+cDTW), against 1-NN classifiers.

Parameter settings: Among the distance measures discussed above, only cDTW requires setting a parameter, to constrain its warping window. We consider two cases from the literature: (i) $cDTW^{opt}$: we compute the optimal window by performing a leave-one-out classification step over the training set of each dataset; (ii) $cDTW^{w}$: we use as window 5%, for $cDTW^5$, and 10%, for $cDTW^{10}$, of the length of the time series of each dataset; this second case is more realistic for an unsupervised setting such as clustering.[6] For the 1-NN classification computation, we also consider the state-of-the-art lower bounding approach LB_{Keogh} [Keogh and Ratanamahatana, 2005], which prunes time series that could not be matched when DTW and cDTW are used. We denote lower bounding with the LB subscript (e.g., cDTW_{LB}^{10}). For clustering, all partitional and spectral algorithms that we compare receive the target number of clusters k as input, which is equal to the number of classes for each dataset. The only exception is the partitional k-MS method, which requires setting an additional parameter, namely, the number of centroids per cluster. For k-MS, we set the number of centroids per cluster, L, to $L = 5$ across all datasets.

For hierarchical clustering, we compute a threshold that cuts the produced dendrogram at the minimum height such that k clusters are formed. We set the maximum number of iterations for k-Shape, all variants of k-means, PAM, and spectral methods to 100. Finally, in every iteration of k-Shape, k-DBA, and KSC, we use the centroids of the previous run as reference sequences to refine the centroids of the current run once.

In contrast to partitional, hierarchical, and spectral methods, density-based methods require as input two parameters. Specifically, DBSCAN requires as input a cutoff threshold, e, and the minimum number of points to form a density region, *minPts*. To estimate e for each distance measure, we fix the value of *minPts* (we consider values 3, 4, 5, 10, and 20), compute the full dissimilarity matrix, and keep, for each time series, the largest distance value such that exactly *minPts* time series are within that distance from it. We sort these distance values in descending order and compute the knee point of the curve of distance values. The process of determining this knee point is similar to the process of determining the number of clusters in clustering methods [Salvador and Chan, 2004]. We consider all bisections of the curve and, then, by walking along the curve one bisection point at a time, we fit two straight lines, one to all the points to the left of the bisection point and one to all the points to the right of the bisection point. The knee point of the curve corresponds to

[6]For non-scalable methods, we use the cDTW^{5} variant of cDTW for its efficiency, as noted in Table 3.1, even though cDTW^{10}, cDTW^{opt}, and DTW perform similarly. We explicitly note when DTW is used.

the bisection point with the minimum sum of errors for the two fitted lines. For all distance measures, $minPts = 3$ performed the best on average across all datasets and we denote this variant as DBSCAN^3. However, the best $minPts$ value for a specific dataset might be different, so we also determine the best $minPts$ parameter for each dataset after performing a post-hoc analysis on the Rand Index results (see below). We denote the resulting version of DBSCAN with dataset-specific values of $minPts$ as DBSCAN^{Best}.

The second density-based method that we consider, TADPole, also requires setting the value of two parameters: (i) the number k of clusters and (ii) a cutoff threshold e. Considering that DBSCAN and TADPole operate similarly, and the fact that the authors of TADPole [Begum et al., 2015] do not provide details on the estimation of these parameters, we use the same procedure as with DBSCAN. TADPole also performs the best on average across all datasets when $minPts = 3$. We refer to this version of the algorithm as TADPole^3. To determine the best $minPts$ parameter for each dataset, we perform a post-hoc analysis on the Rand Index results. We refer to this version of the algorithm as TADPole^{Best}.

U-Shapelets requires setting the value of just one parameter, namely, the length of subsequences of the entire time series. We consider values of 10%, 20%, 30%, 40%, and 50% of the time series length of each dataset. The best performing length for U-Shapelets across all datasets is 50% and we denote the resulting technique as U-Shapelets$^{0.5}$. Similarly to DBSCAN and TADPole, we also consider the best length parameter for each dataset after performing a post-hoc analysis. We denote this approach as U-Shapelets^{Best}. We note that U-Shapelets^{Best}, DBSCAN^{Best}, and TADPole^{Best} are not truly options for fully unsupervised clustering, given that their parameters are the result of post-hoc analysis with labeled data. However, we consider these options for completeness.

Finally, we use cDTW^5 to compute the dissimilarity matrices for RT1, RT2, and RT3, as well as to measure the accuracy of 1-NN classifiers for Random, RT1, RT2, and RT3.

Metrics: We compare the approaches on both runtime and accuracy. For runtime, we compute CPU time utilization and measure the time ratios for our comparisons for each dataset. We summarize the runtime performance by reporting, for each method, the number of datasets with time ratios of up to one order of magnitude, between one but no higher than two orders of magnitude, and at least two orders of magnitude in comparison to another method. Following [Ding et al., 2008], we use the 1-NN classifier, which is a simple and parameter-free classifier, to evaluate distance measures. We report the classification accuracy (i.e., number of correctly classified instances over all instances) by performing 1-NN classification over the training and test sets of each dataset. Because the 1-NN classifier is deterministic, we make this computation once. Following [Batista et al., 2014], we also visualize the *actual accuracy gain* and the *expected accuracy gain* for pairs of distance measures to evaluate if we can predict in advance which distance measure is more useful for which datasets. The actual accuracy gain for each dataset is defined as the ratio of the 1-NN classification accuracies of the two distance measures over the test portion of the dataset. The expected accuracy gain for each dataset is defined as the ratio of the leave-one-out classification accuracies of the two distance measures over the training portion of the dataset. In both ratios, the denominator is the accuracy of the baseline distance measure whereas the numerator is the accuracy of the competing distance measure. The resulting plot illustrates the actual accuracy gains against the expected accuracy gains over all datasets and consists of four regions. Each region corresponds to the true-positive (TP), true-negative (TN), false-positive (FP), and false-negative (FN) cases of the well-known contingency table for comparing pairs of classifiers. For datasets in the TP region, we can predict (i.e., by evaluation on the training portions) that the competing distance measure will outperform the baseline distance measure. For datasets in the TN region, we can predict that the baseline distance measure will outperform the competing distance measure. For datasets in the FN region, we inaccurately predict that the baseline distance measure will outperform the competing distance measure. Similarly, for datasets in the FP region, we inaccurately predict that the competing distance measure will outperform the baseline distance measure.

We use the Rand Index [Rand, 1971] to evaluate clustering accuracy over the fused training and test sets of each dataset. This metric is related to the classification accuracy and is defined as $R = \frac{TP+TN}{TP+TN+FP+FN}$, where TP is the number of time series pairs that belong to the same class and are assigned to the same cluster, TN is the number of time series pairs that belong to different classes and are assigned to different clusters, FP is the number of time series pairs that belong to different classes but are assigned to the same cluster, and FN is the number of time series pairs that belong to the same class but are assigned to different clusters. As hierarchical algorithms are deterministic, we report the Rand Index over one run. However, for partitional methods, we report the average Rand Index over 10 runs and for spectral methods the average Rand Index over 100 runs; in every run we use a different random initialization.

Statistical analysis: Following [Batista et al., 2014; Giusti and Batista, 2013], we analyze the results of every pairwise comparison of algorithms over multiple datasets using the Wilcoxon test [Wilcoxon, 1945] with a 99% confidence level. According to [Demšar, 2006], the Wilcoxon test is less affected by outliers than is the t-test [Rice, 2006], as Wilcoxon does not consider absolute commensurability of differences. Moreover, using pairwise tests to reason about multiple algorithms is not fully satisfactory because sometimes the null hypotheses are rejected due to random chance. Therefore, we also use the Friedman test [Friedman, 1937] followed by the post-hoc Nemenyi test [Nemenyi, 1963] for comparison of multiple algorithms over multiple datasets, as suggested in [Demšar, 2006]. The Friedman and Nemenyi tests require more evidence to detect statistical significance than the Wilcoxon test [Giusti and Batista, 2013] (i.e., the larger the number of methods, the larger the number of datasets required) and, hence, as we already use all 85 datasets for the Wilcoxon test, we report statistical significant results with a 95% confidence level.

3.6 Experimental Results

In this section, we report our experimental results. Firstly, we evaluate SBD against the state-of-the-art distance measures (Section 3.6.1) and corroborate our performance results over optimized implementations for the state-of-the-art distance measures (Section 3.6.2).

Secondly, we compare k-Shape and k-MS against scalable (Section 3.6.3) and non-scalable (Section 3.6.4) clustering approaches. Thirdly, we evaluate k-Shape and k-MS against approaches that handle outliers in datasets (Section 3.6.5) and approaches that consider only subsequences of the entire time series in an attempt to handle abnormal behaviors in time series (Section 3.6.6). Then, we evaluate NSC against data editing algorithms that reduce the search space of 1-NN classifiers (Section 3.6.7). Finally, we summarize our findings (Section 3.6.8).

3.6.1 Evaluation of SBD Against Other Distance Measures

We evaluate SBD, the time-series distance measure discussed in Section 3.3.1, in terms of runtime and accuracy performance in comparison to other state-of-the-art time-series distance measures.

Comparison against ED: To understand if SBD (Section 3.3.1) is an effective measure for time-series comparison, we evaluate it against the state-of-the-art distance measures, using their 1-NN classification accuracy across 85 datasets (Section 3.5). Table 3.2 reports the performance of the state-of-the-art measures against the baseline ED. All distance measures, including SBD, outperform ED with statistical significance. In particular, SBD performs at least as well as ED in 77 out of 85 datasets in contrast to DTW, cDTW^5, and cDTW^{10}, which perform at least as well as ED in 56, 63, and 60 datasets, respectively. Only cDTW^{Opt}, with its optimal warping window constraint, reaches the performance of SBD (i.e., cDTW^{Opt} performs at least as well as ED in 92% of the datasets).

However, to better understand the usefulness of each distance measure, we further analyze if we can predict in advance the accuracy gain by using one distance measure over another distance measure (i.e., we want to predict the ratio of the classification accuracy of one distance measure over that of another distance measure). Figure 3.4a shows that for the large majority of the datasets we can accurately predict in advance if SBD will outperform ED or not (please refer to the TP and TN regions of the figure). For only 6 datasets we incorrectly predict that ED will outperform SBD (please refer to the FN region

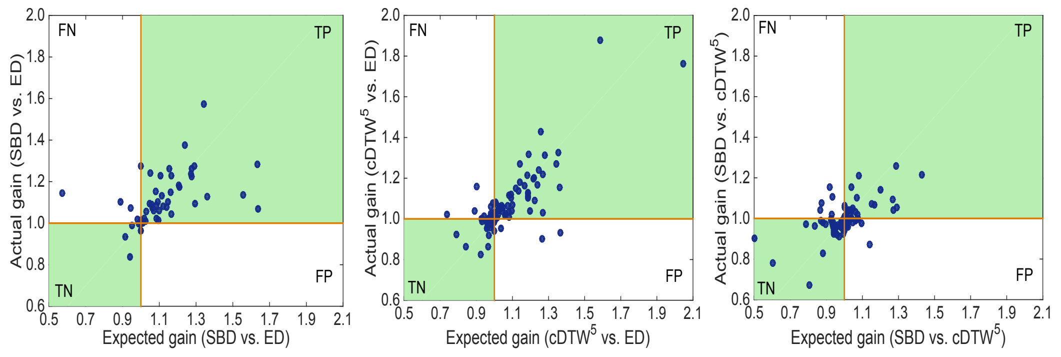
Distance Measure	$>$	$=$	$<$	**Better**	**[0,10x)**	**[10x,100x)**	**[100x,+∞)**
DTW	53	3	29	✓	0	9	76
DTW$_{LB}$					0	14	71
cDTW^{opt}	54	24	7	✓	22	28	35
cDTW$^{opt}_{LB}$					40	29	16
cDTW^{5}	58	5	22	✓	5	39	41
cDTW$^{5}_{LB}$					23	41	21
cDTW^{10}	56	4	25	✓	2	32	51
cDTW$^{10}_{LB}$					8	46	31
SBD$_{NoFFT}$	54	23	8	✓	14	50	21
SBD$_{NoPow2}$					57	28	0
SBD					79	6	0

Table 3.2: Comparison of distance measures. Columns "$>$," "$=$," and "$<$" denote the number of datasets over which a distance measure is better, equal, or worse, respectively, in comparison to ED. "Better" indicates that a distance measure outperforms ED with statistical significance. Columns "[0, 10x)," "[10x, 100x)," and "[100x, $+\infty$)" denote the number of datasets over which a distance measure is slower up to 10x, between 10x but no higher than 100x, and at least 100x, respectively, in comparison to ED.

of the figure). Importantly, 3 out of the 6 datasets are very close to the (1, 1) region, which highlights the difficulties for identifying differences over those datasets. We note that none of the datasets belongs to the worst-case FP region, an important characteristic that also appears in the evaluation of the supervised cDTW^{Opt} distance. In contrast, for cDTW^5 (Figure 3.4b), 3 datasets appear in the FP region, 10 in the FN region, and the rest of the datasets are scattered in the TP and TN regions. As a result, SBD, a parameter-free and unsupervised distance measure, shows similar merit to the supervised cDTW^{Opt} distance.

To further elaborate on the importance of optimally constraining the warping window of DTW, we note that only cDTW^{Opt} is significantly better than DTW. In particular, cDTW^{opt} performs at least as well as DTW in 61 datasets, cDTW^5 performs at least as well as DTW in 57 datasets, and cDTW^{10} performs at least as well as DTW in 67 datasets. For cDTW^5, the Wilcoxon test suggests no statistical significance over DTW whereas for cDTW^{10} the improvement is statistically significant but with a p-value close to the confidence level, which indicates that deeper statistical analysis is required.

Figure 3.5 shows the average rank across datasets of cDTW^{opt}, cDTW^5, cDTW^{10}, and DTW. cDTW^{opt} is the top measure, with an average rank of 2.1, meaning that cDTW^{opt} performed best in the majority of the datasets. The Friedman test rejects the null hypothesis



(a) SBD vs. ED (b) cDTW^5 vs. ED (c) SBD vs. cDTW^5

Figure 3.4: Comparison of accuracy gain for ED, SBD, and cDTW^5 over 85 datasets.

that all measures behave similarly, and, hence, we proceed with a post-hoc Nemenyi test, to evaluate the significance of the differences in the ranks. The wiggly line in the figure connects all measures that do not perform statistically differently according to the Nemenyi test. We observe two clusters: one where the ranks of cDTW^{opt}, cDTW^5, and cDTW^{10} do not present a significant difference, and one where the ranks of cDTW^5, cDTW^{10}, and DTW do not present a significant difference. As a consequence, only cDTW^{opt} appears to perform significantly better than DTW. This conclusion contradicts the general belief that constraining DTW's warping window significantly improves accuracy, which was also confirmed by our previous analysis on a subset of 48 datasets [Paparrizos and Gravano, 2015].

Comparison against DTW and cDTW: SBD performs at least as well as DTW in 50 datasets, but the statistical test reveals no evidence that either measure is better than the other. Considering the cDTW versions, we observe that SBD performs similarly to or better than cDTW^{opt}, cDTW^5, and cDTW^{10} in 41, 37, and 40 datasets, respectively. Figure 3.4c shows that for the large majority of datasets we can accurately predict in advance when either SBD or cDTW^5 should be used (we omit figures for cDTW^{10} and cDTW^{Opt} as they exhibit similar trends). Out of 10 datasets in the FP region, only 2 cases are problematic, while the remaining 8 datasets are very close to the (1, 1) region, and hence it is challenging to predict which distance measure is best. Interestingly, there is no significant difference among SBD, cDTW^5, and cDTW^{10}, but cDTW^{opt} is significantly

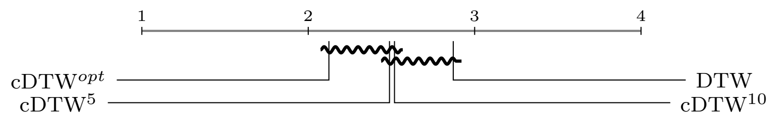


Figure 3.5: Ranking of distance measures based on the average of their ranks across datasets. The wiggly line connects all measures that do not perform statistically differently according to the Nemenyi test.

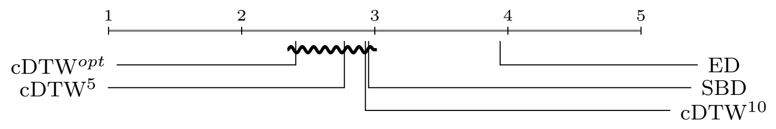


Figure 3.6: Ranking of distance measures based on the average of their ranks across datasets. The wiggly line connects all measures that do not perform statistically differently according to the Nemenyi test.

better than SBD. Importantly, the p-values for Wilcoxon between cDTW^{opt} and SBD are close to the confidence level, which indicates that deeper statistical analysis is required, as we discuss next.

Statistical analysis: To better understand the performance of SBD in comparison with cDTW^{opt}, cDTW^5, and cDTW^{10}, we evaluate the significance of their differences in accuracy when considered all together. Figure 3.6 shows the average rank across datasets of each distance measure. cDTW^{opt} is the top measure, with an average rank of 2.4, meaning that cDTW^{opt} performed best in the majority of the datasets. The Friedman test rejects the null hypothesis that all measures behave similarly, and, hence, we proceed with a post-hoc Nemenyi test, to evaluate the significance of the differences in the ranks. We observe that the ranks of cDTW^{opt}, cDTW^5, cDTW^{10}, and SBD do not present a significant difference, and ED, which is ranked last, is significantly worse than the others. In conclusion, SBD is a very competitive distance measure that significantly outperforms ED and achieves similar results to both cDTW and DTW. Moreover, SBD is the most robust variant of the cross-correlation measure (see Appendix A).

Efficiency: We now show that SBD is not only competitive in terms of accuracy, but also highly efficient. We also demonstrate that implementation choices for SBD can significantly impact its speed. The last row of Table 3.2 shows the number of datasets over which each SBD variation is slower than ED. The optimized version of SBD, denoted simply as SBD, is the fastest version, performing up to 10x slower than ED in 93% of the datasets and

Distance Measure	**Language**	**[0,10x)**	**[10x,100x)**	**[100x,+∞)**	**Reference**
cDTW$^{opt}_{LB}$	Java	25	45	9	[**Schäfer, 2016**]
	Java	19	52	8	[**Wang et al., 2013**]
	Matlab	34	29	16	**This work**
DTW$_{LB}$	Java	1	12	66	[**Schäfer, 2016**]
	Java	0	14	65	[**Wang et al., 2013**]
	Matlab	0	13	66	**This work**

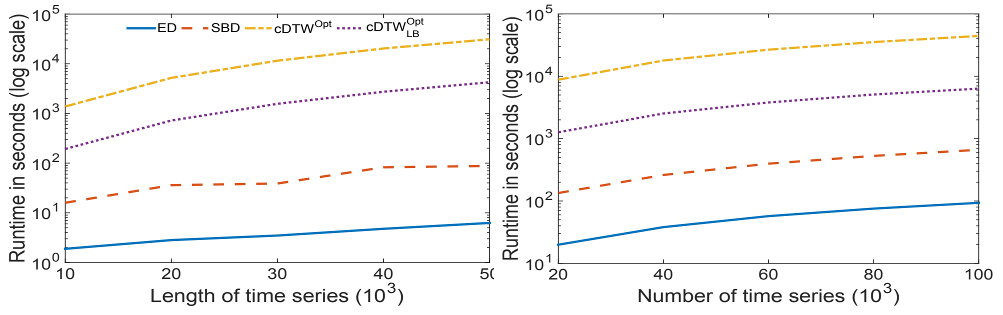
Table 3.3: Comparison of implementations for cDTW^{Opt} and DTW. Column "Language" indicates the programming language. Columns "[0, 10x)," "[10x, 100x)," and "[100x, +∞)" denote the number of datasets over which a distance measure is slower up to 10x, between 10x but no higher than 100x, and at least 100x, respectively, in comparison to ED. "Reference" indicates the work that describes the implementation details.

10x but less than 100x slower in the remaining 7% of the datasets. When we use SBD still with FFT but without the power-of-two-length optimization discussed in Section 3.3.1, the resulting measure, SBD$_{NoPow2}$, is up to 10x slower than ED in 67% of the datasets and 10x but less than 100x slower in 33% of the datasets. Furthermore, SBD$_{NoFFT}$, the version of SBD without FFT, performs up to 10x, 10x but no higher than 100x, and at least 100x slower than ED in 16%, 59%, and 25% of the datasets, respectively. Table 3.2 also shows that DTW and cDTW variants are substantially slower than SBD. Specifically, DTW is up to 100x slower in 40% of the datasets and at least 100x in 60% of the datasets in comparison to SBD. Similarly, cDTW^{opt}, cDTW^5, and cDTW^{10} are more than 10x slower in comparison to SBD in 51%, 65%, and 79% of the datasets, respectively. Even when we speed up the search of 1-NN classification computation, by pruning time series impossible for a match using LB$_{Keogh}$, SBD is still faster. In particular, DTW$_{LB}$ is up to 100x slower in 46% of the datasets and more than 100x in 54% of the datasets in comparison to SBD. cDTW$^{opt}_{LB}$, cDTW$^5_{LB}$, and cDTW$^{10}_{LB}$ are more than 10x slower in comparison to SBD in 24%, 35%, and 54% of the datasets, respectively. Next, we corroborate our performance findings over different implementations of the rival distance measures and perform experiments to better understand the scalability of distance measures.

3.6.2 Robustness of Runtime Results Across Implementations

Considering that the choice of programming language as well as optimizations in implementation may affect the performance of distance measures, we now compare our Matlab implementation of cDTW and DTW variants across two state-of-the-art Java implementations from the literature. Specifically, we compare our performance results for cDTW$_{LB}^{opt}$ and DTW$_{LB}$ with the results as reported in [Schäfer, 2016], which follows the optimizations described in [Rakthanmanon et al., 2012], and with results we obtained in our experimental settings with an implementation that follows the optimizations described in [Wang et al., 2013]. Table 3.3 compares the performance of the three implementations over 79 datasets that are common between [Schäfer, 2016] and our work. DTW$_{LB}$ performs similarly across all implementations. In particular, DTW$_{LB}$ is up to 100x slower than ED in approximately 17% of the datasets and at least 100x slower than ED in approximately 83% of the datasets. Lower bounding methods prune only a small fraction of the DTW computations (on average about 25%) and, hence, there is minor variability in performance across implementations.

In contrast to DTW$_{LB}$, for cDTW$_{LB}^{opt}$ there is some variability in the results across implementations. However, for a vast majority of datasets the results are consistent across implementations. Specifically, cDTW$_{LB}^{opt}$ is up to 100x slower than ED in approximately 89% of the datasets and at least 100x slower than ED in approximately 11% of the datasets for the two Java implementations. The implementation of [Schäfer, 2016], which exploits multiple lower bounding measures in a cascade, is more efficient than [Wang et al., 2013], which uses only one lower bounding method. Our version of cDTW$_{LB}^{opt}$ is up to 100x slower than ED in 80% of the datasets and at least 100x slower than ED in approximately 20% of the datasets. In comparison to the Java implementations, we observe that the performance of cDTW$_{LB}^{opt}$ appears to decrease in a small number of datasets with long time series because, in such datasets, Matlab's underlying optimizations in linear algebra operations benefit ED. Therefore, for approximately 10% of the datasets, our version of cDTW$_{LB}^{opt}$ appears slower than the Java implementations but, importantly, for approximately 18% of the datasets (i.e., in datasets with short time series), cDTW$_{LB}^{opt}$ appears faster than the Java implementations.



(a) Varying m for n=500 (b) Varying n for m=4K

Figure 3.7: Runtime of ED, SBD, cDTW^{Opt}, and cDTW$^{Opt}_{LB}$ as a function of (a) the number of time series n and (b) the time series length m.

Overall, we can conclude that the performance results of our implementation of DTW and cDTW variants are consistent with what has been reported previously in the literature, with some small variability in the results of cDTW due to underlying optimizations in programming languages and choices of lower bounding methods.

Even though our performance results are consistent across implementations, we believe that several factors hinder the process of understanding the runtime performance when solely the UCR datasets are used. In particular, the large variations in the number and length of time series across datasets do not help to provide a clear picture of how distance measures scale as a function of such variable numbers. Moreover, the choice of different parameters to constrain DTW's warping window can significantly affect the runtime performance of the distance measure and the pruning power of lower bounding approaches.

To better understand the runtime performance and scalability of distance measures, we perform an additional experiment using the synthetic CBF dataset [Saito, 1994]. This dataset enables experiments with varying values of (a) the number of time series n and (b) the time series length m, without changing any of its general properties. We report the CPU runtime[7] over values for n and m that are orders of magnitude larger than those for the majority of the datasets in the UCR archive [Keogh et al., 2015]. Specifically, in Figure 3.7a we vary m from 10,000 to 50,000 while we set $n = 500$. We observe that SBD is

[7]For this experiment, we distributed the computation across 8 processes. The code was extracted and compiled into Matlab's C/C++ MEX files.

an order of magnitude slower (8-18x) than ED with varying lengths while cDTW$_{LB}^{Opt}$ is two orders of magnitude slower (102-679x) and cDTW^{Opt} is two to three orders of magnitude slower (732-4969x) than ED. SBD remains an order of magnitude faster (12x-48x) than cDTW$_{LB}^{Opt}$ and one to two orders of magnitude faster (87-354x) than cDTW^{Opt}. Similarly, in Figure 3.7b we vary n from $20,000$ to $100,000$ while we set $m = 4,000$. We observe that all measures scale linearly with the number of time series when m is static and SBD remains less than an order of magnitude slower (6.9x) than ED while cDTW$_{LB}^{Opt}$ is an order of magnitude slower (62x) and cDTW^{Opt} is two orders of magnitude slower (464x) than ED.

3.6.3 Evaluation of k-Shape and k-MS Against Scalable Methods

Having shown the robustness of SBD, we now compare k-Shape and k-MS, our time-series clustering algorithms discussed in Section 3.3.3, against other scalable time-series clustering algorithms.

Comparison against k-AVG+ED: Table 3.4 reports the performance of variants of k-means against k-AVG+ED, using their Rand Index on the 85 datasets (see Section 3.5). From all these variants of k-means, only k-Shape and k-MS outperform k-AVG+ED with statistical significance, as we show next. In most cases, replacing ED in k-means with other distances not only does not improve accuracy significantly, but in certain cases results in substantially lower performance. For example, k-AVG+SBD achieves higher accuracy than k-AVG+ED in 56% of the datasets, but the differences in accuracy are not statistically significant. Interestingly, when DTW is combined with k-means, in k-AVG+DTW, the performance is significantly worse than with k-AVG+ED.

Even in cases where both the distance measure and the centroid computation method of k-means are modified, the performance does not improve significantly in comparison to k-AVG+ED. k-DBA outperforms k-AVG+DTW with statistical significance in 67 out of 85 datasets. Both of these approaches use DTW as distance measure, but k-DBA also modifies its centroid computation method. This modification significantly improves the performance

Algorithm	>	=	<	**Better**	**Worse**	**[0,10x)**	**[10x,100x)**	**[100x,+∞)**
***k*-AVG+SBD**	48	5	32	✗	✗	53	32	0
***k*-AVG+DTW**	24	2	59	✗	✓	1	11	73
KSC	32	2	51	✗	✓	1	3	81
***k*-DBA**	38	1	46	✗	✗	0	0	85
***k*-Shape+DTW**	32	2	51	✗	✗	0	7	78
***k*-Shape**	57	6	22	✓	✗	17	55	13
***k*-MS**	64	3	18	✓	✗	3	46	36

Table 3.4: Comparison of k-means variants against k-AVG+ED. Columns ">," "=," and "<" denote the number of datasets over which a method is better, equal, or worse, respectively, in comparison to k-AVG+ED. "Better" indicates that a method outperforms k-AVG+ED with statistical significance whereas "Worse" indicates that k-AVG+ED outperforms a method with statistical significance. Columns "[0, 10x)," "[10x, 100x)," and "[100x, $+\infty$)" denote the number of datasets over which a method is slower up to 10x, between 10x but no higher than 100x, and at least 100x, respectively, in comparison to k-AVG+ED.

of k-DBA over that of k-AVG+DTW, with an average improvement in Rand Index of 19%. Despite this improvement, k-DBA still does not perform better than k-AVG+ED in a statistical significant manner.[8] Another algorithm that modifies both the distance measure and the centroid computation method of k-means is KSC. In contrast to k-DBA, KSC is significantly worse than k-AVG+ED. We note that this finding contradicts our previous analysis on a subset of 48 datasets [Paparrizos and Gravano, 2015] where k-AVG+ED was shown to perform better in 54% of the datasets, but not in a statistically significant manner. Our current analysis is based on almost twice as many datasets as our analysis in [Paparrizos and Gravano, 2015], which provides substantially more evidence to the Wilcoxon test. (The p-value for Wilcoxon was very close to the confidence level in [Paparrizos and Gravano, 2015], which resulted in considering the difference in accuracy of k-AVG+ED and KSC negligible.)

Comparison of *k*-MS against *k*-Shape: Having shown that inadequate modifications of the distance and the centroid computation method of k-means do not improve accuracy, we now evaluate k-Shape and k-MS, the only k-means variants with significant improvements in accuracy over k-AVG+ED. Specifically, k-Shape performs better than k-AVG+ED in

[8]Even when multiple refinements of k-DBA's centroids are performed per iteration, k-DBA still does not outperform k-AVG+ED. In particular, performing five refinements per iteration improves the Rand Index by 4% but runtime increases by 30% according to experiments over 48 datasets [Paparrizos and Gravano, 2015]. (These 48 datasets are a subset of the 85 datasets in our experiments.)

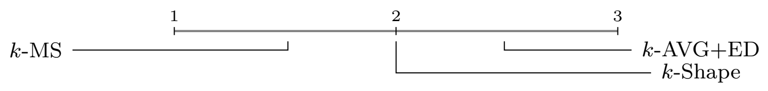


Figure 3.8: Ranking of the best performing variants of k-means, namely, k-MS and k-Shape, based on the average of their ranks across datasets.

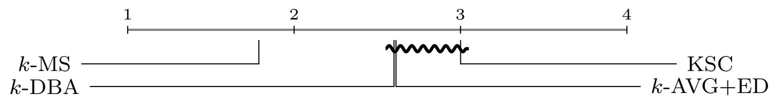


Figure 3.9: Ranking of k-means variants based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

67% of the datasets and k-MS performs better than k-AVG+ED in 75% of the datasets. Importantly, in both of these comparisons, Wilcoxon indicates the superiority of k-Shape and k-MS against k-AVG+ED. Comparing now the performance of k-Shape and k-MS, our results show that k-MS performs better, equally, and worse in 59, 4, and 22 datasets, respectively, in comparison to k-Shape. Importantly, Wilcoxon indicates that the difference in accuracy of k-MS and k-Shape is statistically significant. Therefore, we can conclude that k-Shape and k-MS are the only k-means variants that significantly outperform k-AVG+ED and, importantly, k-MS is the only variant that significantly outperforms k-Shape.

Comparison against k-DBA and KSC: Both k-DBA and KSC are similar to k-Shape and k-MS in that they all modify both the centroid computation method and the distance measure of k-means. Therefore, to understand the impact of these modifications, we compare them against our k-Shape and k-MS algorithms. k-Shape performs equally or better in 59 datasets in comparison to KSC and performs equally or better in 58 datasets in comparison to k-DBA. In both of these comparisons, the statistical test indicates the superiority of k-Shape. Similarly, k-MS performs equally or better in 62 datasets in comparison to KSC and performs equally or better in 61 datasets in comparison to k-DBA. In all these comparisons, the statistical test indicates the superiority of k-Shape and k-MS.

Statistical analysis: In addition to the pairwise comparisons performed with the Wilcoxon test, we further evaluate the significance of the differences in algorithm performance when considered all together. Figure 3.8 shows the average rank across datasets of k-MS, k-Shape, and k-AVG+ED, the best performing scalable clustering methods. k-MS is the top method, with an average rank of 1.51, meaning that *k*-MS achieved better rank in the majority of the datasets. The Friedman test rejects the null hypothesis that all algorithms behave similarly, and, hence, we proceed with a post-hoc Nemenyi test, to evaluate the significance of the differences in the ranks. We observe that the difference in ranks of each method are statistically significant: *k*-Shape is significantly better than *k*-AVG+ED and, in turn, *k*-MS is significantly better than *k*-Shape. Figure 3.9 shows the average rank across datasets of *k*-MS and *k*-means variants that modify both the centroid computation method and the distance measure of *k*-means. As before, *k*-MS is ranked first, with an average rank of 1.78. We observe that the ranks of KSC, *k*-DBA, and *k*-AVG+ED do not present a statistically significant difference, whereas *k*-MS, which is ranked first, is significantly better than the others. We note that according to the Wilcoxon test, as mentioned earlier, KSC is significantly worse than *k*-AVG+ED. However, when KSC, *k*-DBA, *k*-AVG+ED, and *k*-MS are considered collectively, KSC performs similarly to *k*-DBA and *k*-AVG+ED. (We observe similar findings for *k*-Shape in our analysis over 85 datasets. We omit figures due to space limitations. In [Paparrizos and Gravano, 2015], we provide such comparisons on a subset of 48 datasets.) To conclude, modifying *k*-means with inappropriate distance measures or centroid computation methods might lead to unexpected results. The same holds for *k*-Shape, where the use of DTW, in *k*-Shape+DTW, results in a significant drop in performance (i.e., *k*-Shape outperforms *k*-Shape+DTW with statistical significance in 60 out of 85 datasets).

Efficiency: *k*-Shape and *k*-MS are the only algorithms that outperform all *k*-means variants, including the simple, yet robust, *k*-AVG+ED. We now investigate whether the superiority of *k*-Shape and *k*-MS in terms of accuracy has an associated penalty in efficiency. Table 3.4 shows the number of datasets over which each method is slower than *k*-AVG+ED. *k*-Shape is the fastest method that significantly outperforms *k*-AVG+ED. *k*-MS is slower than *k*-Shape due to the additional cost required to compute more centroids. However, *k*-MS remains only up to 10x slower than *k*-Shape in 93% of the datasets. Importantly, *k*-MS remains significantly faster than all *k*-means variants that modify both the distance and the centroid computation method of *k*-means. Specifically, KSC performs up to 10x,

Algorithm	>	=	<	**Better**	**Worse**
H-S+ED	17	1	67	✗	✓
H-S+cDTW	21	0	64	✗	✓
H-S+SBD	20	1	64	✗	✓
H-A+ED	16	0	69	✗	✓
H-A+cDTW	22	0	63	✗	✓
H-A+SBD	23	0	62	✗	✓
H-C+ED	28	0	57	✗	✓
H-C+cDTW	33	0	52	✗	✓
H-C+SBD	35	0	50	✗	✓
S+ED	27	1	57	✗	✓
S+cDTW	36	1	48	✗	✓
S+SBD	61	0	24	✓	✗
PAM+ED	43	1	41	✗	✗
PAM+cDTW	55	2	28	✓	✗
PAM+SBD	58	2	25	✓	✗

Table 3.5: Comparison of hierarchical, spectral, and k-medoids variants against k-AVG+ED. Columns ">," "=," and "<" denote the number of datasets over which an algorithm is better, equal, or worse, respectively, in comparison to k-AVG+ED. "Better" indicates that an algorithm outperforms k-AVG+ED with statistical significance whereas "Worse" indicates that k-AVG+ED outperforms an algorithm with statistical significance.

10x but less than 100x, and at least 100x slower than k-Shape, in 20%, 78%, and 2% of the datasets, respectively. KSC performs up to 10x slower than k-MS in 60% of the datasets, and 10x but less than 100x slower than k-MS in 40%. k-DBA performs at least 100x slower than k-Shape in 74% of the datasets and at least 100x slower than k-MS in 44% of the datasets.

3.6.4 Evaluation of k-Shape and k-MS Against Non-Scalable Methods

Up to now, we have focused our evaluation on scalable clustering algorithms. In order to show the robustness of k-Shape and k-MS in terms of accuracy beyond scalable approaches, we now ignore scalability and compare k-Shape and k-MS against clustering methods that scale quadratically with the number of time series, namely, hierarchical, spectral, and k-medoids methods.

Comparison against k-AVG+ED: Table 3.5 reports the performance of non-scalable methods against k-AVG+ED. Among all existing state-of-the-art methods that use ED or cDTW as distance measures, only partitional methods perform similarly to or better than k-AVG+ED. In particular, PAM+cDTW is the only method that outperforms k-AVG+ED with statistical significance. PAM+ED achieves better performance in 51% of the datasets in comparison to k-AVG+ED; however, this difference is not statistically significant. Moreover, PAM+cDTW performs better in 51 datasets, equal in 2 datasets, and worse in 32 datasets relative to PAM+ED. For this comparison, the statistical significance test indicates the superiority of PAM+cDTW. We note that constraining the warping window for DTW does not affect significantly the accuracy of clustering methods. Figure 3.10 shows the ranking of PAM+DTW, PAM+cDTW^5, PAM+cDTW^{10}, and k-AVG+ED across all datasets. PAM+DTW, PAM+cDTW^5, and PAM+cDTW^{10} do not present significant difference in accuracy, but PAM+DTW is ranked first. This is contradictory to the findings for the classification task where DTW was ranked last (see Figure 3.5).

All combinations of hierarchical clustering, with all different linkage criteria, perform poorly in comparison to k-AVG+ED. Interestingly, k-AVG+ED outperforms all of them with statistical significance. We observe that the major difference in performance among hierarchical methods is attributed to the linkage criterion and not to the distance measure. This highlights the importance of the clustering method, in addition to the distance measure for time-series clustering. Similarly to hierarchical methods, spectral methods also perform poorly against k-AVG+ED. S+cDTW performs better in more datasets than S+ED in comparison to k-AVG+ED: comparing S+cDTW with S+ED, S+cDTW achieves similar or better accuracy in 41 datasets, but this difference is not significant. Importantly, k-AVG+ED is significantly better than both S+ED and S+cDTW. Therefore, for hierarchical and spectral methods, these distance measures have a small impact on their performance.

k-Shape and k-MS against H-C+cDTW, S+cDTW, and PAM+cDTW: Among all methods using cDTW as distance measure that we evaluated, only PAM+cDTW outperforms k-AVG+ED with statistical significance. Similarly to k-AVG+ED, k-Shape and k-MS also outperform H-C+cDTW and S+cDTW with statistical significance. Therefore, we compare this approach with k-Shape and k-MS. PAM+cDTW performs equally or better in 45 datasets in comparison to k-Shape and equally or better in 41 datasets in comparison

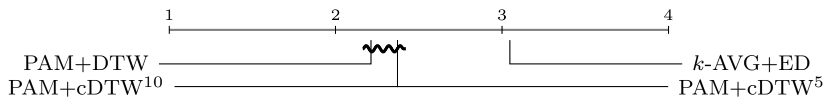


Figure 3.10: Ranking of partitional methods that outperform k-AVG+ED based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

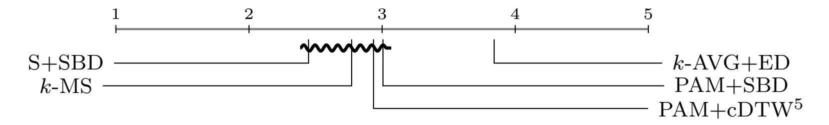


Figure 3.11: Ranking of methods that outperform k-AVG+ED based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

to k-MS, but these differences are not significant.

For completeness, we also evaluate SBD with hierarchical, spectral, and k-medoids methods. For hierarchical methods, H-C+SBD is better than H-A+SBD, and, in turn, H-A+SBD is better than H-S+SBD, all with statistical significance. S+SBD, in contrast to S+ED and S+cDTW, outperforms k-AVG+ED in 61 out of 85 datasets, with statistical significance. We note that S+SBD is the method that outperforms k-AVG+ED in the largest number of datasets (72%) than any other method evaluated in this work. S+SBD is also significantly better than S+ED and S+cDTW, but S+SBD does not outperform k-Shape or k-MS. Similarly, PAM+SBD performs better in 58, equal in 2, and worse in 25 datasets in comparison to k-AVG+ED. The statistical test suggests that PAM+SBD is significantly better than k-AVG+ED but not better than k-Shape or k-MS.

Statistical analysis: We evaluate the significance of the differences in algorithm performance for all algorithms that significantly outperform k-AVG+ED. Our analysis indicates similar behavior for k-Shape and k-MS and, therefore, we omit k-Shape to simplify our analysis and visualizations. Figure 3.11 shows that k-MS, PAM+SBD, PAM+cDTW, and S+SBD do not present a significant difference in accuracy, whereas k-AVG+ED, which is ranked last, is significantly worse than the others.

From our extensive evaluation of existing clustering approaches for time series that use ED, cDTW, or DTW as distance measures, PAM+cDTW is the only approach that achieves similar — but not better — results compared to k-Shape and k-MS. In contrast

Algorithm	$>$	$=$	$<$	Better	Worse
DBSCAN^3+ED	25	0	60	✗	✓
DBSCAN^{Best}+ED	32	0	53	✗	✓
DBSCAN^3+SBD	26	0	59	✗	✓
DBSCAN^{Best}+SBD	33	0	52	✗	✓
DBSCAN^3+cDTW	30	0	55	✗	✓
DBSCAN^{Best}+cDTW	37	0	48	✗	✓
TADPole^3	35	1	49	✗	✗
TADPole^{Best}	41	1	43	✗	✗
U-Shapelets$^{0.5}$	31	0	54	✗	✓
U-Shapelets^{Best}	44	0	41	✗	✗

Table 3.6: Comparison of variants of density-based and shapelet-based methods against k-AVG+ED. Columns "$>$," "$=$," and "$<$" denote the number of datasets over which an algorithm is better, equal, or worse, respectively, in comparison to k-AVG+ED. "Better" indicates that an algorithm outperforms k-AVG+ED with statistical significance whereas "Worse" indicates that k-AVG+ED outperforms an algorithm with statistical significance. "Rand Index" denotes the accuracy achieved in the 85 datasets.

to k-Shape and k-MS, PAM+cDTW has two drawbacks that make it an unrealistic choice for time-series clustering: (i) its distance measure, cDTW, requires tuning to improve its performance and reduce the computation cost; and (ii) the computation of the dissimilarity matrix that PAM+cDTW requires as input makes it unable to scale in both time and space. For example, the matrix computation alone is already two orders of magnitude slower than the computation required by k-Shape.

3.6.5 Evaluation of k-Shape and k-MS Against Outlier-Aware Methods

Previously, we focused our evaluation on state-of-the-art scalable and non-scalable clustering methods that do not exploit any particular mechanism to handle outliers in the datasets. We now evaluate k-Shape and k-MS against two outlier-aware methods, namely, DBSCAN and TADPole (see Section 3.5).

Comparison against k-AVG+ED: Table 3.6 reports the performance of outlier-aware methods (see Section 3.5) against k-AVG+ED, using their Rand Index on the 85 datasets (see Section 3.5). From all these approaches, none outperforms k-AVG+ED and, importantly, k-AVG+ED significantly outperforms all variants of DBSCAN. In particular, k-AVG+ED achieves better performance in 71%, 69%, and 65% of the datasets in comparison to DBSCAN^3+ED, DBSCAN^3+SBD, and DBSCAN^3+cDTW, respectively. DBSCAN^3+cDTW is the strongest of the combinations of DBSCAN with other distance measures. Specifically, DBSCAN^3+cDTW performs at least as well as DBSCAN^3+ED in 50 datasets and at least as well as DBSCAN^3+SBD in 45 datasets. However, the Wilcoxon test suggests that these differences in accuracy are not statistically significant.

Similarly to DBSCAN, TADPole does not outperform k-AVG+ED. Specifically, k-AVG+ED performs at least as well as TADPole^3 in 50 datasets, but this difference is not significant. Comparing DBSCAN and TADPole, we observe that TADPole^3 significantly outperforms all DBSCAN variants. In particular, TADPole^3 outperforms DBSCAN^3+cDTW, DBSCAN^3+SBD, and DBSCAN^3+ED in 54, 56, and 55 datasets, respectively.

As discussed in Section 3.5, both DBSCAN and TADPole require as input two parameters. For fairness in the comparison, we now also report experimental results for the DBSCAN and TADPole variants when they receive as input the best performing parameters for each dataset, as described in Section 3.5. DBSCAN^{Best}+cDTW performs at least as well as DBSCAN^{Best}+ED and DBSCAN^{Best}+SBD in 46 datasets, but these differences are not statistically significant. Interestingly, k-AVG+ED performs at least as well as DBSCAN^{Best}+cDTW in 48 datasets, at least as well as DBSCAN^{Best}+SBD in 52 datasets, and at least as well as DBSCAN^{Best}+ED in 53 datasets, and all these differences in accuracy are statistically significant. In contrast, k-AVG+ED performs equally or better in 44 in comparison to TADPole^{Best}, but this difference in accuracy is not statistically significant.

Comparison against k-Shape and k-MS: Among all density-based methods we evaluated, none outperformed k-AVG+ED with statistical significance. Therefore, even though k-Shape and k-MS are significantly better than k-AVG+ED, we evaluate k-Shape and k-MS against all density-based methods. Similarly to k-AVG+ED, k-Shape and k-MS also significantly outperform DBSCAN^3+ED, DBSCAN^3+SBD, and DBSCAN^3+cDTW. In particular, k-Shape and k-MS perform at least as well as DBSCAN^3+cDTW in 74% of the datasets. In contrast to k-AVG+ED, k-Shape significantly outperforms TADPole^3 in 70% of the datasets and k-MS significantly outperforms TADPole^3 in 74% of the datasets.

Statistical Analysis: Up to now, we evaluated density-based methods pairwise. We now

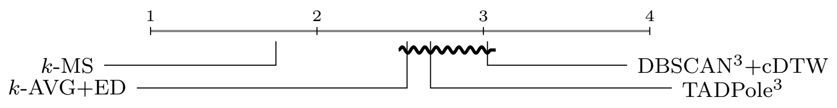


Figure 3.12: Ranking of TADPole^3, DBSCAN^3, k-AVG+ED, and k-MS based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

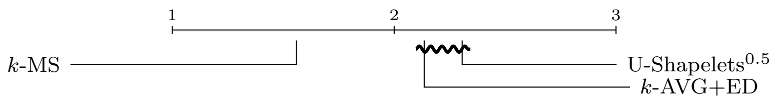


Figure 3.13: Ranking of U-Shapelets$^{0.5}$, k-AVG+ED, and k-MS based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

evaluate the significance of the differences in algorithm performance for the best performing density-based methods when considered collectively. The Friedman test rejects the null hypothesis that all algorithms behave similarly, and, hence, as before, we proceed with a post-hoc Nemenyi test. Figure 3.12 shows that TADPole^3 and DBSCAN^3+cDTW do not exhibit a significant difference in accuracy. However, in contrast to TADPole^3, DBSCAN^3+cDTW is significantly worse than k-AVG+ED. k-MS, which is ranked first, is significantly better than both density-based methods. (We omit similar results for k-Shape.)

3.6.6 Evaluation of k-Shape and k-MS Against a Shapelet-Based Method

Up to now, we have evaluated k-Shape and k-MS against scalable, non-scalable, and outlier-aware time-series clustering methods. To conclude our analysis, we now evaluate our algorithms against U-Shapelets, a shapelet-based method that exploits patterns in subsequences of the entire time series (see Section 3.5).

Comparison against k-AVG+ED: The last two rows of Table 3.6 report the performance of U-Shapelets variants against k-AVG+ED, using their Rand Index on the 85 datasets (see Section **??**). These two variants do not outperform k-AVG+ED and, importantly, k-AVG+ED significantly outperforms U-Shapelets$^{0.5}$. In particular, k-AVG+ED performs better in 54 and worse in 31 datasets in comparison to U-Shapelets$^{0.5}$. For completeness, we note that U-Shapelets^{Best} significantly improves the performance of U-Shapelets. In particular, U-Shapelets^{Best} performs better in 57 and equal in 28 datasets in comparison to

U-Shapelets$^{0.5}$. U-Shapelets^{Best} performs better in 44 and worse in 41 datasets in comparison to k-AVG+ED but the difference in accuracy is not statistically significant.

Comparison against DBSCAN^3+cDTW and TADPole^3: We now evaluate the performance of U-Shapelets$^{0.5}$ against density-based methods. U-Shapelets$^{0.5}$ performs better in 52, equal in 1, and worse in 32 datasets in comparison to the best performing DBSCAN variant, namely, DBSCAN^3+cDTW. The Wilcoxon text suggests that this difference in accuracy is not statistically significant, but with a p-value close to the confidence level. Similarly, TADPole^3 performs at least as well as U-Shapelets$^{0.5}$ in 47 datasets, but this difference in accuracy is not statistically significant.

Comparison against k-Shape and k-MS: Among all shapelet-based methods we evaluated, none outperformed k-AVG+ED. Therefore, even though k-Shape and k-MS are significantly better than k-AVG+ED, we evaluate k-Shape and k-MS against the best performing shapelet-based method: both k-Shape and k-MS outperform U-Shapelets$^{0.5}$ in 57 out of 85 datasets. The Wilcoxon test suggests that this difference in accuracy is significant.

Statistical Analysis: Having shown the superiority of k-Shape and k-MS against U-Shapelets$^{0.5}$, we now conclude our analysis with an evaluation of the significance of the differences in algorithm performance when considered collectively. The Friedman test rejects the null hypothesis that all algorithms behave similarly, and, hence, as before, we proceed with a post-hoc Nemenyi test. Figure 3.13 shows that k-AVG+ED and U-Shapelets$^{0.5}$ do not present a significant difference, whereas k-MS, which is ranked first, is significantly better than k-AVG+ED and U-Shapelets$^{0.5}$. (We omit similar results for k-Shape.)

3.6.7 Evaluation of NSC Against Data Editing Methods

Up to now, we have demonstrated the robustness of k-Shape and k-MS as standalone clustering methods for time series. In this section, we evaluate the performance of k-Shape, our simplest—and competitive—clustering method, as a subroutine to effectively reduce the search space of one-nearest-neighbor algorithms.

We perform three experiments to assess the performance of NSC against rival methods.

Algorithm	***k*=1**	***k*=2**	***k*=3**	***k*=4**
Random	2.919	2.713	3.081	2.978
RT1	3.603	3.706	3.331	3.404
RT2	3.132	3.132	3.147	3.140
RT3	3.684	3.926	3.824	3.853
NSC	1.662	1.522	1.618	1.625
$\mathbf{R}_{Best}$-$\mathbf{R}_{NSC}$	1.257	1.191	1.463	1.353

Table 3.7: Comparison of Random, RT1, RT2, RT3, and NSC. Columns "k=1," "k=2," "k=3," and "k=4" denote the average ranking, in terms of accuracy, of each algorithm over 75 datasets when k centroids represent each class. The last row represents the difference in ranking of the best performing rival method and NSC. Differences higher than 0.55 are considered statistically significant according to the Nemenyi test.

First, following [Petitjean et al., 2014; Petitjean et al., 2016], we evaluate performance when a limited number of centroids are allowed per class. Second, we evaluate the impact of several key characteristics of the methods. Third, we compare the performance of 1-NN classifiers against three variants of NSC, namely, NSC+ED, NSC+SBD, and NSC+cDTW. The first two experiments are over the 75 datasets—out of the 85 datasets—for which all algorithms terminated after running continuously for one week, as discussed in Section 3.5. In contrast, the third experiment, which is based solely on the efficient NSC, is over the full set of 85 datasets.

Table 3.7 reports the average rank, in terms of accuracy, of Random, RT1, RT2, RT3, and NSC when k centroids represent each class in the training set.[9] NSC is the top method for all values of k that we considered, meaning that NSC achieved higher accuracy in the majority of the datasets. Specifically, NSC significantly outperforms each of the rival methods in at least 80% of the datasets for all values of k, according to Wilcoxon. To understand the performance of NSC in comparison with Random, RT1, RT2, and RT3, we also evaluate the significance of their differences in accuracy when considered all together. The Friedman test rejects the null hypothesis that all algorithms behave similarly for all values of k and, hence, we proceed with a post-hoc Nemenyi test. The last row in Table 3.7 shows the difference in ranks of NSC and the best performing method. According to the Nemeyi test, differences higher than 0.55 are significant. Therefore, for all different values

[9]We vary k from 1 to 4 in order to include datasets with a small number of instances per class.

Algorithm	Average Rank	Initialization Cost			Query Time Improvement				Amortization Cost		
		[0,10x)	[10x,100x)	[100x,+∞)	[0,25)	[25,50)	[50,75)	[75,100]	[0,10x)	[10x,100x)	[100x,+∞)
RT1	2.838				48	10	7	10			
RT2	2.561	40	30	5	43	9	10	13	9	17	48
RT3	2.696				45	7	12	11			
NSC	1.905	49	26	0	21	8	11	35	45	16	14

Table 3.8: Comparison of RT1, RT2, RT3, and NSC. Column “Average Rank” denotes the average rank in terms of the number of centroids required to outperform the corresponding 1-NN accuracy. Column “Initialization Cost” denotes the number of datasets over which an algorithm requires up to 10x (column “[0, 10x)”), between 10x but no higher than 100x (column “[10x, 100x)”), and at least 100x (column “[100x, +∞)”) more preprocessing than the version of the Random method with the corresponding distance. Column “Query Time Improvement” indicates the number of datasets over which an algorithm improves the query response time up to 25% (column “[0, 25)”), from 25% to 50% (column “[25, 50)”), from 50% to 75% (column “[50, 75)”), and from 75% to 100% (column “[75, 100)”) relative to the corresponding 1-NN. Column “Amortization Cost” denotes the number of datasets over which an algorithm requires to increase the number of queries up to 10x (column “[0, 10x)”), between 10x but no higher than 100x (column “[10x, 100x)”), and at least 100x (column “[100x, +∞)”) relative to the corresponding 1-NN to amortize the initialization cost.

of k, the difference in the rank of NSC against all rival methods is statistically significant.

Having shown the superiority in terms of accuracy of NSC against other methods, we now explore the impact on four important characteristics of NSC, RT1, RT2, and RT3 in their attempt to outperform their corresponding 1-NN classifier.[10] In Table 3.8, we first evaluate the methods based on the number of instances required to outperform their corresponding 1-NN classifier. NSC is the top method, meaning that NSC required fewer centroids than the other methods to outperform its corresponding 1-NN classifier in the majority of datasets. The Friedman test rejects the null hypothesis that all algorithms behave similarly and, hence, we proceed as before with a post-hoc Nemenyi test. According to Nemenyi, the difference in the rank of NSC against all rival methods is statistically significant.

We then investigate the associated initialization cost for each method to outperform the corresponding 1-NN classifiers. Considering that 1-NN classifiers require no preprocessing of the instances in the training set (i.e., the initialization cost of 1-NN classifiers is negligible), we measure the initialization cost of each method against the corresponding Random

[10]We exclude the Random method from this experiment as Random could not outperform 1-NN classifier with cDTW^5 in the majority of the datasets.

method. NSC performs up to 10x slower than Random in 65% of the datasets, while in the remaining 35% of the datasets NSC performs between 10x but no higher than 100x slower in comparison to Random. In contrast, RT1, RT2, and RT3 perform up to 10x, between 10x but no higher than 100x, and at least 100x slower than Random in 53%, 40%, and 7% of the datasets, respectively.[11] Therefore, we can conclude that NSC is faster at preprocessing the instances in the training set than all other reduction techniques.

Considering that all methods significantly reduce the number of instances in the training set in comparison with 1-NN classifiers, we now evaluate the associated improvement in query runtime in comparison with the corresponding 1-NN classifiers. Specifically, NSC reduces the query response time by at least 50% in 61% of the datasets. In contrast, the best reduction techniques, namely, RT1 and RT2, reduce the query response time by at least 50% in 31% of the datasets. Therefore, NSC reduces the query response time by at least 50% in approximately twice as many datasets as do the reduction techniques.

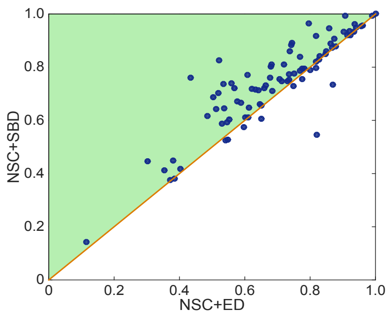
Finally, we report the number of queries each method has to process to amortize the initialization cost in comparison with the associated 1-NN classifiers. NSC increases the number of queries up to 10x in 60% of the datasets and by at least 10x in 40% of the datasets in order to amortize the initialization cost. In contrast, the reduction techniques increase the number of queries up to 10x in 13% of the datasets and by at least 10x in 67% of the datasets. Therefore, we can conclude that NSC is a competitive method to reduce the instances required for training the 1-NN classifiers. NSC is significantly more accurate than rival methods, requires fewer centroids to effectively summarize the search space, and yields important reductions in query response time in comparison with 1-NN classifiers.

Having shown the superiority of NSC against other data editing techniques, we now evaluate the effectiveness of NSC when combined with competitive distance measures. Table 3.9 reports the performance of NSC+SBD and NSC+cDTW against NSC+ED on 85

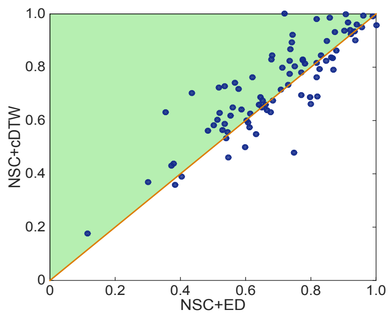
[11]Considering that all reduction techniques are statistically indistinguishable in terms of accuracy and number of instances required to outperform 1-NN classifier with cDTW[5], in Table 3.8 we only report the "Initialization Cost" and the "Amortization Cost" of the best performing reduction technique, namely, RT2, in order to simplify our analysis.

Algorithm	>	=	<	Better	Worse
NSC+cDTW	53	2	30	✓	✗
NSC+SBD	67	3	15	✓	✗

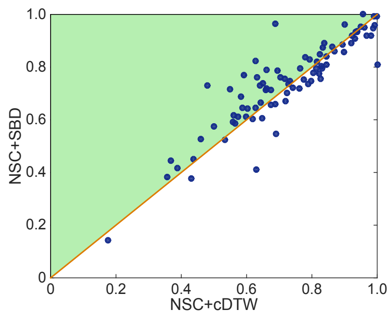
Table 3.9: Comparison of NSC+cDTW and NSC+SBD against NSC+ED. Columns ">," "=," and "<" denote the number of datasets over which an algorithm is better, equal, or worse, respectively, in comparison to NSC+ED. "Better" indicates that an algorithm outperforms NSC+ED with statistical significance whereas "Worse" indicates that NSC+ED outperforms an algorithm with statistical significance.



(a) NSC+SBD vs. NSC+ED



(b) NSC+cDTW vs. NSC+ED



(c) NSC+SBD vs. NSC+cDTW

Figure 3.14: Comparison of NSC+SBD against NSC+ED (a), NSC+cDTW against NSC+ED (b), and NSC+SB against NSC+cDTW (c) over 85 datasets. Circles above the diagonal indicate datasets over which the method in the vertical axis has better accuracy than the method in the horizontal axis.

datasets. Both NSC+SBD and NSC+cDTW significantly outperform NSC+ED. In particular, NSC+SBD performs at least as well as NSC+ED in 82% of the datasets and NSC+cDTW performs at least as well as NSC+ED in 65% of the datasets. Figure 3.14 presents pairwise differences in accuracy between NSC variants over 85 datasets. Each circle in Figures 3.14a, 3.14b, and 3.14c represents a dataset; the first coordinate of each circle corresponds to the classification accuracy of the method in the horizontal axis and the second coordinate of each circle corresponds to the classification accuracy of the method in the vertical axis. Circles above the diagonal indicate datasets over which the method in the vertical axis has better accuracy than the method in the horizontal axis. Specifically, Figures 3.14a and 3.14b illustrate the superiority of NSC+SBD and NSC+cDTW against NSC+ED, respectively. In contrast, NSC+SBD performs better in 45, equal in 1, and worse in 39 datasets in comparison to NSC+cDTW. Wilcoxon suggests that this difference in accuracy is not statistically significant. Figure 3.14c shows the performance of

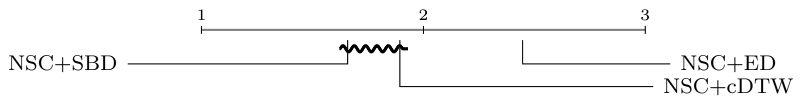


Figure 3.15: Ranking of nearest shape classifier methods based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

NSC+SBD against NSC+cDTW. In addition to the pairwise comparisons performed with Wilcoxon, we further evaluate the significance of the differences in algorithm performance when considered collectively. Figure 3.15 shows the average rank across datasets of each NSC variant. NSC+SBD is the top method, with an average rank of 1.66, meaning that NSC+SBD achieved better rank than NSC+ED and NSC+cDTW in the majority of the datasets. The Friedman test rejects the null hypothesis that all algorithms behave similarly and, hence, we proceed with a post-hoc Nemenyi test, to evaluate the significance of the differences in the ranks. We observe that the ranks between NSC+SBD and NSC+cDTW do not present any significant difference, whereas NSC+ED, which is ranked last, is significantly worse than the others.

Considering now the performance of NSC and 1-NN variants for the same distances collectively, we observe no significant difference between NSC+SBD and 1-NN+SBD, and no significant difference between NSC+ED and 1-NN+ED. However, NSC+cDTW is significantly worse than 1-NN+cDTW, which indicates that k-Shape is not an appropriate clustering method to compute centroids when cDTW is used. Instead, other methods, such as PAM+cDTW or k-DBA, would be preferable in conjunction with cDTW.

In conclusion, k-Shape is an effective method to summarize time series, and 1-NN classifiers can benefit from using k-Shape as a subroutine to significantly reduce their search space. Importantly, these findings have implications in settings where time-series classification methods operate under limited computational resources [Petitjean et al., 2014; Petitjean et al., 2016] or within a tight time budget [Ding et al., 2015; Pelkonen et al., 2015].

3.6.8 Summary of Experimental Evaluation

In short, our experimental evaluation suggests that: (1) cross-correlation measures (e.g., SBD), which are not widely adopted as time-series distance measures, are as competitive as state-of-the-art measures, such as cDTW and DTW, but significantly faster; (2) only the optimally tuned cDTW measure significantly outperforms the DTW measure when 1-NN classification is used to evaluate distance measures, in contrast to the belief that cDTW generally improves the accuracy of DTW; (3) for time-series clustering, the DTW measure often achieves better performance than the cDTW measure, which contradicts the findings under 1-NN classification; (4) the *k*-means algorithm with ED, in contrast to what has been reported in the literature, is a robust approach for time-series clustering, but inadequate modifications of its distance measure and centroid computation can reduce its performance; (5) the choice of clustering method, such as partitional versus density-based methods, which was believed to be less important than that of distance measure, is as important as the choice of distance measure; (6) the intrinsic characteristics of clustering algorithms, such as the linkage criterion in hierarchical clustering, are more important than the choice of distance measures; (7) the attempts to isolate and ignore abnormal behaviors in time series can significantly hurt the performance of outlier-aware clustering methods; (8) *k*-Shape and *k*-MS are highly accurate and efficient methods for time-series clustering; and (9) exploiting clustering methods, such as *k*-Shape, as subroutines to reduce the search space of 1-NN algorithms can lead to fast and accurate classification methods for time series.

3.7 Conclusions

In this chapter, we addressed the problem of domain-independent, accurate, and scalable time-series clustering (Section 3.2.4). We summarize the contributions of this chapter as follows: (i) we showed how to derive SBD, a scale-, translate-, and shift-invariant distance measure, in a principled manner from cross-correlation and how to efficiently compute this measure by exploiting intrinsic characteristics of Fourier transform algorithms (Section

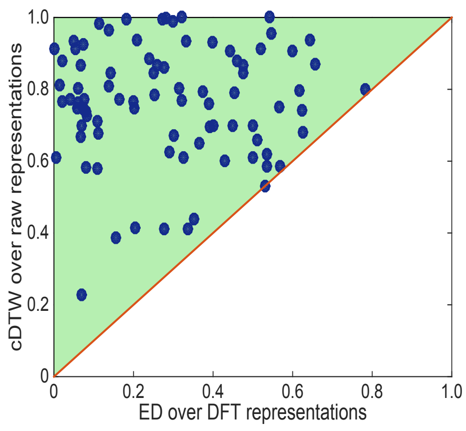
3.3.1); (ii) we presented SE and MSE, two novel methods to compute a single centroid or multiple centroids per cluster when the SBD distance measure is used (Section 3.3.2); (iii) we developed k-Shape and k-MS, two novel algorithms for time-series clustering that rely on the SBD distance measure to compare time series; k-Shape relies on the SE method to compute a single centroid per cluster based on the entire set of time series in each cluster, whereas k-MS relies on MSE to compute multiple centroids per cluster in order to consider the proximity and spatial distribution of time series in each cluster (Section 3.3.3); (iv) we presented NSC, a one-nearest-neighbor classification algorithm that relies on k-Shape as subroutine to reduce the search space of one-nearest-neighbor algorithms (Section 3.4); and (v) we performed an extensive experimental evaluation of SBD, k-Shape, k-MS, and NSC against state-of-the-art time-series distance measures, clustering algorithms, and classification algorithms (Sections 3.5 and 3.6).

Our findings show that SBD is an efficient and parameter-free distance measure that achieves similar results to the most accurate but computationally expensive distance measures that require parameter tuning. k-Shape and k-MS outperform all state-of-the-art scalable and non-scalable partitional, hierarchical, spectral, density-based, and shapelet-based clustering approaches, with only one method achieving similar performance. Interestingly, this method is significantly slower than both k-Shape and k-MS and, importantly, its distance measure requires tuning. Finally, exploiting clustering methods, such as k-Shape, as subroutines to reduce the search space of 1-NN algorithms can lead to fast and accurate classification methods for time series. Consequently, we believe that SBD, k-Shape, k-MS, and NSC emerge as domain-independent, accurate, and scalable approaches for time-series comparison, clustering, and classification. Importantly, our approaches can serve as core components in the development of efficient and accurate methods for other tasks over time series. In fact, in Chapter 4 we build on principles behind SBD to develop an efficient kernel function for time-series comparison that significantly outperforms existing kernels that operate over the full length of the time series. Additionally, we show how k-Shape can serve as a core subroutine for learning time-series representations.

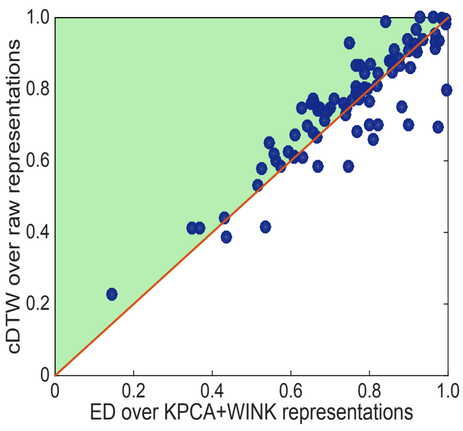
Chapter 4

Efficient Time-Series Representation Learning

In Chapter 3, we presented general purpose, scalable, yet accurate, clustering algorithms for time series. Our algorithms compare time series efficiently and compute centroids effectively under the scaling, translation, and shift invariances. In this chapter, we exploit the simplest version of our time-series clustering algorithms, namely k-Shape, to extract dictionaries of time series that meaningfully summarize time-series collections. Our goal is to develop a generic framework capable to represent the time series as a linear combination of the time series in a dictionary. In contrast to existing time-series representation methods, our approach enables the learning of low-dimensional representations that preserve the similarities, as defined by a user-specified similarity function, of time-series in the high-dimensional space. Therefore, our representations seamlessly integrate with existing data mining methods to solve tasks of critical importance in time-series analysis. First, we provide an overview for that problem and review the associated challenges (Section 4.1). Second, we describe necessary background for linear and non-linear dimensionality reduction methods (Section 4.2). Third, we present our efficient, data-aware, time-series representation learning framework (Section 4.3) and show how to leverage learned time-series representation to solve major time-series mining tasks (Section 4.4). Then, we summarize our extensive experi-



(a) Raw vs. DFT



(b) Raw vs. KPCA+WINK

Figure 4.1: Comparison of classification accuracies across 85 datasets using raw representations against representations of length 20 computed with DFT and KPCA+WINK. Circles over the diagonal indicate datasets over which raw representations outperform compact representations.

mental evaluation (Sections 4.5 and 4.6). Finally, we conclude with the contributions of this chapter (Section 4.7).

4.1 Overview and Motivation

Time series are often the outcome of sequentially recording time-varying measurements of an underlying process [Esling and Agon, 2012]. Recent technological advances permit the collection of enormous amounts of time-varying measurements across scientific and industrial applications [Palpanas, 2015; Mahdavinejad et al., 2017]. For example, data-intensive applications in astronomy [Alam et al., 2015] and neuroscience [Biswal et al., 2010] involve millions of time series. Similarly, infrastructure monitoring systems of large-scale internet services handle billions of time-stamped measurements per day [Loboz et al., 2010; Pelkonen et al., 2015; Jasta, 2016]. With sensors becoming cheap and ubiquitous and with the explosion of Internet of Things applications, the rapid growth of time-series volumes is expected to continue [Mahdavinejad et al., 2017]. Consequently, time-series analysis algorithms will have to operate over increasingly massive time-series collections.

State-of-the-art methods for most time-series mining tasks, such as clustering [Liao, 2005] and classification [Bagnall et al., 2017], cannot scale to millions of time series. The temporal ordering and the high dimensionality of the observations, combined with the large volumes of time series, introduce severe challenges not shared with the analysis of other types of data [Yang and Wu, 2006; Esling and Agon, 2012]. Specifically, the temporal ordering may introduce distortions (see Section 4.2.3), which complicate the definition of similarity functions to compare time series that agree with human perception. In addition, the high dimensionality increases the computation and storage requirements of methods operating directly over time series. Finally, the large volumes of time series can make effective methods over moderate-sized collections appear impractical in large-scale settings. To address these challenges, the design of scalable and effective time-series mining methods involves decisions for three core components [Esling and Agon, 2012]: (i) the *representation* method to construct low-dimensional representations that preserve fundamental characteristics of time series; (ii) the *comparison* method, which should offer invariances to inherent distortions in time series; and (iii) the *indexing* method, which organizes time series to facilitate fast retrieval from massive collections.

There has been significant effort to propose and optimize methods for each of the three components separately [Esling and Agon, 2012], which led to a mismatch among the characteristics preserved by representations methods, the invariances offered by effective, yet computationally expensive, similarity functions, and the properties imposed on similarity functions by indexing mechanisms. Therefore, existing time-series mining methods suffer from a number of drawbacks: (i) these methods become prohibitively expensive as they operate directly over raw time series to avoid information loss from direct exploitation of representations (e.g., this is the case for most classification methods [Bagnall et al., 2017]); (ii) these methods sacrifice accuracy for efficiency as they directly exploit representations (e.g., this is the case for online clustering methods that only support less-effective ℓ_p-norms [Ding et al., 2015]); or (iii) these methods follow a complicated, two-step approach to indirectly exploit representations to prune part of the expensive pairwise comparisons (e.g., this is the case for querying methods that exploit representations that do not preserve invariances offered by the chosen similarity function [Yi et al., 1998; Kim et al., 2001; Sakurai et al., 2005b; Keogh and Ratanamahatana, 2005; Rakthanmanon et al., 2012]).

The two-step approach is the most prominent paradigm to accelerate time-series mining methods while addressing all aforementioned challenges. The requirement to perform two steps is due to the dependence on the seminal GEMINI framework [Agrawal et al., 1993; Faloutsos et al., 1994], which laid the foundations for fast time-series retrieval under Euclidean distance (ED). Specifically, GEMINI (i) constructs low-dimensional representations; and (ii) defines a measure over such representations to *lower bound* (i.e., prune part of) the ED comparisons in the high-dimensional space. For different distance measures, two-step approaches [Yi et al., 1998; Kim et al., 2001; Sakurai et al., 2005b; Keogh and Ratanamahatana, 2005] additionally show that ED lower bounds the new distance. The plethora of representation and comparison methods [Esling and Agon, 2012; Wang et al., 2013; Bagnall et al., 2017], along with the difficulty in developing effective lower bounding measures, impact the sustainability of such methodology.

To simplify and unify the design of scalable time-series mining methods, we need a new primitive to automatically learn low-dimensional representations that builds on and extends the principles of GEMINI. Specifically, given a comparison method, the learned representation: ($\mathcal{P}1$) preserves the pairwise similarities; ($\mathcal{P}2$) lower bounds the comparison method; ($\mathcal{P}3$) permits reusing part of its coordinates; ($\mathcal{P}4$) supports efficient and memory-tractable similarity computation for available and new data; and ($\mathcal{P}5$) supports efficient and memory tractable eigendecomposition of the data-to-data matrix. Through these principles, the learned representation: (i) leverages methods relying on feature-vectors ($\mathcal{P}1$); (ii) permits time-series querying and indexing ($\mathcal{P}2$); (iii) scales methods under limited resources ($\mathcal{P}3$); (iv) enables operations in online settings ($\mathcal{P}4$); and (v) exploits highly effective methods relying on eigendecomposition ($\mathcal{P}5$).

A promising direction to develop such framework arises with the use of kernel methods, which have been successfully used in the analysis of complex, real-world, data. [Cortes and Vapnik, 1995; Hofmann et al., 2008; Schölkopf et al., 1998; Schölkopf and Smola, 2002;

Bach and Jordan, 2005]. In contrast to other methods that operate directly over data, kernel methods interact with data only through pairwise comparisons. Due to the difficulty in designing kernel functions to assess the similarity between time series, kernel methods remain largely unexplored in the analysis of time series. However, given a suitable kernel function, kernel methods show great promise for unifying the way we model data, learn representations of the data, and design algorithms. Specifically, given a kernel matrix, consisting of pairwise similarities computed with a chosen kernel function, Kernel Principal Component Analysis (KPCA) [Schölkopf et al., 1997] constructs representations that preserve the properties of this kernel function. To illustrate this point, in Figure 4.1 we compare the one-nearest-neighbor (1NN) classification accuracies across 85 datasets [Keogh et al., 2015] using raw time-series representations against representations of length 20 computed by Discrete Fourier Transform (DFT), a state-of-the-art representation method [Wang et al., 2013; Schäfer and Högqvist, 2012], and KPCA with WINK (KPCA+WINK), our kernel function that we discuss later. DFT representations show a significant reduction in accuracy in comparison to the state-of-the-art constrained Dynamic Time Warping (cDTW) [Sakoe and Chiba, 1978] distance measure over raw time series. Instead, KPCA+WINK representations achieve accuracy comparable to cDTW. Despite the remarkable accuracy, KPCA+WINK suffers from high space (quadratic) and runtime (cubic) requirements, with respect to the number of time series (see Section 4.2.4).

To alleviate the burdens associated with the high memory and runtime costs of (i) computing the kernel matrix and (ii) performing dimensionality reduction with KPCA, we present a Generic RepresentAtIon Learning (GRAIL) framework to automatically learn low-dimensional representations in linear space and time. Considering the importance of shift- and warp-invariant comparison methods (see Section 4.2.3), we first develop WINK, a novel, highly efficient, kernel function. Then, GRAIL relies on (i) the Nyström method [Nyström, 1930; Williams and Seeger, 2001; Drineas and Mahoney, 2005] to construct a temporal representation to approximate the kernel matrix; and (ii) matrix sketching [Liberty, 2013] over the temporal representation to approximate the eigendecomposition of the kernel matrix and learn the final representation. The effectiveness of Nyström critically depends on the choice of landmark time series, a difficult combinatorial problem. On the other side, the compactness and the quality of the learned representations rely on an accurate estimation of kernel function parameters. To address these issues, GRAIL computes landmark time series using k-Shape, our highly accurate and scalable time-series clustering algorithm (see Chapter 3). Based on the landmark time series, GRAIL optimizes necessary parameters and constructs low-dimensional representations by expressing time series as a linear combination of the landmark time series. GRAIL representations seamlessly integrate with and, therefore, accelerate, the vast literature of kernel methods [Schölkopf and Smola, 2002]. In particular, we focus on five tasks of critical importance in time-series analysis [Esling and Agon, 2012], namely, (i) clustering; (ii) classification; (iii) querying; (iv) sampling; and (v) visualization of time series.

To demonstrate the robustness of WINK and GRAIL, we perform an extensive evaluation on 85 datasets and compare their performance against state-of-the-art methods on all five tasks. We use public datasets and make our source code available to ensure the reproducibility of our results. In summary, we show that kernel classifiers using WINK can significantly outperform 1NN classifiers with state-of-the-art distance measures [Esling and Agon, 2012; Wang et al., 2013; Bagnall et al., 2017], which debunks a long-standing perception in the time-series literature that 1NN classifiers are difficult to beat [Xi et al., 2006; Wang et al., 2013; Bagnall et al., 2017]. GRAIL representations are more compact than current representations and achieve better pruning power than hand-crafted lower bounding measures. Importantly, GRAIL representations, when combined with suitable kernel methods, significantly outperform state-of-the-art approaches in all five aforementioned tasks.

We start with an in-depth review of the relevant background (see Section 4.2). We continue as follows:

- We develop WINK, a kernel function to compare time series under shift- and warp-invariances (Section 4.3.1).
- We construct landmark time series for Nyström using an effective time-series clustering

method (Section 4.3.2).

- We present a method to estimate kernel function parameters and compactness of representations (Section 4.3.3).
- We learn GRAIL representations by approximating the eigendecomposition of the kernel matrix (Section 4.3.4).
- We show how GRAIL representations accelerate kernel methods for major time-series mining tasks (Section 4.4).
- We evaluate our ideas by conducting an extensive experimental evaluation (Sections 4.5 and 4.6).

Finally, we conclude and discuss the implications of our work (Section 4.7). We now review necessary background and define our problem of focus in this chapter.

4.2 Preliminaries

In this section, we describe linear dimensionality reduction methods (Section 4.2.1). We summarize time-series representation methods (Section 4.2.2) and common time-series distortions and distance measures (Section 4.2.3). Then, we review nonlinear dimensionality reduction methods (Section 4.2.4) along with relevant approximate methods (Section 4.2.5). Finally, we present our problem of focus (Section 4.2.6).

4.2.1 Linear Dimensionality Reduction

Dimensionality reduction is the process of mapping high-dimensional vectors into a low-dimensional space while preserving some property of interest [Cunningham and Ghahramani, 2015; Golub and Van Loan, 2012]. Depending on the mapping, we divide dimensionality reduction methods into *linear* and *nonlinear* methods [Lee and Verleysen, 2007]. We now review linear methods. Section 4.2.4 discusses nonlinear methods.

Linear dimensionality reduction: Consider n real-valued m-dimensional vectors $X = [\vec{x}_1, \ldots, \vec{x}_n]^T \in \mathbb{R}^{n \times m}$ and a target dimensionality $k < m$. The goal is to produce low-dimensional vectors $Z_k = [\vec{z}_1, \ldots, \vec{z}_n]^T \in \mathbb{R}^{n \times k}$ such that pairwise similarities, expressed as inner products, in the low-dimensional space closely approximate pairwise similarities in the high-dimensional space. Formally, the objective is to learn a linear transformation $P \in \mathbb{R}^{m \times k}$ and construct $Z_k = XP$ by optimizing the following problem:

$$\underset{Z \in \mathbb{R}^{n \times k}}{\arg\min} \; ||XX^T - Z_k Z_k^T||_F^2 \tag{4.1}$$

where $||A||_F = \sqrt{\sum_i^n \sum_i^n A_{ij}^2}$, for any square matrix $A \in \mathbb{R}^{n \times n}$, denotes the Frobenius norm. This is the problem that cornerstone methods such as Singular Value Decomposition (SVD) [Golub and Van Loan, 2012], Principal Component Analysis (PCA) [Pearson, 1901; Hotelling, 1933], and Multidimensional Scaling (MDS) [Torgerson, 1952; Cox and Cox, 2000] solve optimally [Golub and Van Loan, 2012; Cunningham and Ghahramani, 2015]. SVD, PCA, and MDS produce the same Z_k, as we will see, but differ in the way they operate over X. Specifically, SVD operates directly over X, PCA operates over the covariance matrix $C = X^T X \in \mathbb{R}^{m \times m}$, and MDS operates over the Gram matrix $K = XX^T \in \mathbb{R}^{n \times n}$. To see how SVD, PCA, and MDS relate to each other, we first express the SVD of X as follows:

$$X = U \Sigma V^T \tag{4.2}$$

where $U = [\vec{u}_1, \ldots, \vec{u}_n] \in \mathbb{R}^{n \times n}$ is a column-orthonormal matrix such that $\vec{u}_i$, with $i \in [1, n]$, are the left singular vectors, $V = [\vec{v}_1, \ldots, \vec{v}_m] \in \mathbb{R}^{m \times m}$ is a column-orthonormal matrix such that $\vec{v}_i$, with $i \in [1, m]$, are the right singular vectors, and $\Sigma = diag(\sigma_1, \ldots, \sigma_m) \in \mathbb{R}^{n \times m}$ is a matrix containing the singular values in descending order along its diagonal (i.e., $\sigma_1 \geq \ldots \geq \sigma_m$), with zeroes everywhere else. A matrix V is column-orthonormal if its columns are mutually orthogonal unit vectors (i.e., $V^T V = I$, where I is the identity matrix). Then, we express the eigendecomposition of any square matrix A as $Q \Lambda Q^T \in \mathbb{R}^{d \times d}$, where $Q = [\vec{q}_1, \ldots, \vec{q}_d] \in \mathbb{R}^{d \times d}$ is a column-orthonormal matrix with the eigenvectors of A and $\Lambda = diag(\lambda_1, \ldots, \lambda_d) \in \mathbb{R}^{d \times d}$ is a matrix containing the eigenvalues of A in descending order along its diagonal (i.e., $\lambda_1 \geq \ldots \geq \lambda_d$). Let $C = Q_C \Lambda_C {Q_C}^T$ and $K = Q_K \Lambda_K {Q_K}^T$ be the eigendecompositions of C and K, respectively. Given the SVD of X (Equation 4.2), the following two relations hold:

$$C = X^T X = V\Sigma U^T U \Sigma V^T = V\Sigma^2 V^T \tag{4.3}$$

$$K = XX^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T \tag{4.4}$$

Therefore, the eigenvectors of C are the same as the left singular vectors of X (i.e., $Q_C = V$), the eigenvectors of K are the same as the right singular vectors of X (i.e., $Q_K = U$), and the eigenvalues of K and C are the squares of the singular values of X (i.e., $\Lambda_K = \Lambda_C = \Sigma^2$). Based on the above, MDS constructs $Z_m^{MDS} = Q_K\sqrt{\Lambda_K} = U\Sigma$, PCA constructs $Z_m^{PCA} = XQ_C = XV$, and SVD constructs $Z_m^{SVD} = XV$. From Equation 4.2, we know that $XV = U\Sigma$ and, therefore, $Z_m^{MDS} = U\Sigma = XV = Z_m^{PCA} = Z_m^{SVD}$. By retaining the first $k < m$ eigenvalues and eigenvectors, we obtain the following low-dimensional representation:

$$Z_k = U_{1:n,1:k}\Sigma_{1:k,1:k} = XV_{1:m,1:k} \tag{4.5}$$

Despite producing identical representations, PCA and SVD require $\mathcal{O}(min\{n \cdot m^2 + m^3, m \cdot n^2 + n^3\})$ time, whereas MDS requires $\mathcal{O}(n^2)$ time to construct K and $\mathcal{O}(n^3)$ time to perform the eigendecomposition on K. For linear dimensionality reduction, it is common to use PCA or SVD due to their reduced computational cost. However, for nonlinear dimensionality reduction, the construction of K is unavoidable and, therefore, MDS is the prevalent option (see Section 4.2.4).

4.2.2 Time-Series Representation Methods

Due to the high dimensionality of time series, representation methods [Golub and Van Loan, 2012; Cunningham and Ghahramani, 2015; Percival and Walden, 2006; Jain, 1989; Oppenheim and Schafer, 2009] aim to reduce it to lower the storage and computational costs of time-series analysis methods. Since the debut of the GEMINI framework [Agrawal et al., 1993; Faloutsos et al., 1994], research on representation methods has focused on exploring trade-offs between low-dimensional representations, such as reconstructing quality, sensitivity to noise, compactness, and computational cost. Depending on the transformation applied to time series and on the output format, we divide representation methods into *data-agnostic* and *data-aware* methods and into *numeric* and *symbolic* methods [Esling and Agon, 2012]. In Section 2.4, we provided an in-depth discussion of the state of the art for time-series representation methods that we will not repeat here for brevity.

Despite their optimality [Golub and Van Loan, 2012; Cunningham and Ghahramani, 2015], linear dimensionality reduction methods are prohibitively expensive (see Section 4.2.1). Therefore, approaches such as DFT, PAA, and SAX are used in practice. All previously mentioned methods construct representations to preserve or lower bound ED. Recent experimental evaluations of distance measures [Ding et al., 2008; Wang et al., 2013; Bagnall et al., 2017] show that ED is inappropriate for most applications. Next, we review alternative distance measures.

4.2.3 Invariances and Distance Measures

As we discussed in Sections 2.2 and 2.3, distance measures handle the majority of distortions by preprocessing time series before comparison [Keogh and Kasetty, 2003; Keogh et al., 2004; Argyros and Ermopoulos, 2003; Oppenheim and Schafer, 2009; Goldin and Kanellakis, 1995]. However, for many important distortions, preprocessing is not effective and, therefore, sophisticated distance measures offer invariances during comparison. For example, to satisfy the *shift invariance*, our SBD distance measure (see Section 3.3.1) compares out-of-phase time series, whereas DTW [Sakoe and Chiba, 1978] compares time series with misaligned regions. Our extensive experimental evaluation in Sections 3.5 and 3.6 showed that SBD and DTW achieve similar classification accuracy but with significant benefits for SBD in terms of efficiency.

4.2.4 Nonlinear Dimensionality Reduction

Linear dimensionality reduction methods fail to effectively model data with complex, nonlinear structures, where kernel methods thrive [Cortes and Vapnik, 1995; Hofmann et al., 2008; Schölkopf et al., 1997; Schölkopf et al., 1998; Schölkopf and Smola, 2002]. Specifically, kernel methods use a function $\phi : \mathbb{R}^m \to \mathcal{H}$ to implicitly map data into a Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H}$, of high (and often infinite) dimension. Because explicit computation of coordinates in $\mathcal{H}$ is infeasible, kernel methods invoke the "kernel trick" [Aizerman et al., 1964], which permits efficient interaction with data through pairwise comparisons, $k(\vec{x},\vec{y}) = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle_{\mathcal{H}}$, or the corresponding distance metric, $D_k(\vec{x},\vec{y}) = k(\vec{x},\vec{x}) + k(\vec{y},\vec{y}) - 2k(\vec{x},\vec{y})$ [Schölkopf, 2001]. As a consequence, to perform dimensionality reduction in $\mathcal{H}$, where direct computation of SVD or PCA is infeasible, KPCA [Schölkopf et al., 1997; Schölkopf et al., 1998] performs eigendecomposition over the Gram matrix K, where $K_{ij} = k(\vec{x_i},\vec{x_j}) = \phi(\vec{x_i})\phi(\vec{x_j})^T$. Therefore, KPCA is the nonlinear counterpart of MDS, discussed in Section 4.2.1. Similarly to MDS, KPCA suffers from high space requirements. Specifically, KPCA requires $\mathcal{O}(n^2)$ storage, where n is the number of data instances. Furthermore, its runtime is prohibitively expensive: it requires $\mathcal{O}(n^2)$ time to construct K and $\mathcal{O}(n^3)$ time to perform an eigenvalue decomposition on K. In addition, KPCA requires to careful tune kernel function parameters that affect the quality and compactness of the low-dimensional representation. In Section 4.3.3, we propose a simple, yet effective, method to tune kernel parameters. Next, we review approximations for kernel methods.

4.2.5 Approximations for Kernel Methods

To alleviate the burdens associated with the high memory and runtime costs of kernel methods, the Nyström method [Nyström, 1930; Williams and Seeger, 2001; Drineas and Mahoney, 2005] approximates the Gram matrix K using a set of landmark vectors, whereas the Random Fourier Features (RFF) method [Rahimi and Recht, 2008] approximates K through randomized feature maps.

The Nyström method: The objective of Nyström is to approximate $k(\vec{x_i}, \cdot) = \phi(\vec{x_i}) \ \forall \ i \in$

$[1,n]$ as a linear combination of d landmark vectors $G = [\vec{g}_1, \ldots, \vec{g}_d]^T \in \mathbb{R}^{d \times m}$ [Oglic and Gärtner, 2017]:

$$\underset{a \in \mathbb{R}^{d \times n}}{\arg\min} \sum\nolimits_{i=1}^{n} ||K(\vec{x}_i, \cdot) - \sum\nolimits_{j=1}^{d} a_{j,i} K(\vec{g}_j, \cdot)||_{\mathcal{H}}^2 \tag{4.6}$$

For each $\vec{x}_i$, this is a convex problem depending on a single column of a and, therefore, the optimal solution is $a = W^{-1}E^T$, where $W \in \mathbb{R}^{d \times d}$, with $W_{ij} = k(\vec{g}_i, \vec{g}_j) \forall i,j \in G$, $E \in \mathbb{R}^{n \times d}$, with $E_{ij} = k(\vec{x}_i, \vec{g}_j) \forall i \in X, j \in G$, and $W^{-1} = Q_W \Lambda_W^{-1} Q_W^{-1}$ is the inverse of W. Nyström constructs $\hat{K} \approx K$ as follows:

$$\hat{K} = EW^{-1}E^T = (EQ_W \Lambda_W^{-0.5})(EQ_W \Lambda_W^{-0.5})^T = Z_d Z_d^T \tag{4.7}$$

Despite the closed form solution for this problem, the selection of landmark points is a difficult combinatorial problem with a large body of work [Williams and Seeger, 2001; Drineas and Mahoney, 2005; Zhang et al., 2008; Kumar et al., 2012; Gittens and Mahoney, 2013; Li et al., 2016].

The Random Fourier Features method: Unlike Nyström, which implicitly lifts data to $\mathcal{H}$, RFF aims to explicitly approximate $k(\vec{x}, \vec{y})$. Specifically, RFF constructs randomized feature maps $F : \mathbb{R}^m \to \mathbb{R}^d$ such that the inner product of two transformed vectors approximates their kernel function: $\langle F(\vec{x}), F(\vec{y}) \rangle \approx k(\vec{x}, \vec{y})$. Through that step, RFF converts X into Z_d and approximates K with $\hat{K} = Z_d Z_d^T$. Despite the benefits of explicitly embedding data in low-dimensional space, RFF requires the custom construction of feature maps for any given kernel function [Maji and Berg, 2009; Avron et al., 2014; Hamid et al., 2014; Kar and Karnick, 2012].

Nyström is a data-aware method computing landmark vectors from available data and requires $\mathcal{O}(n \cdot d^2)$ time to construct Z_d, where n is the number of data instances and d is the number of landmark vectors. In contrast, RFF is a data-agnostic method sampling feature maps from a distribution independent from available data and requires $\mathcal{O}(n \cdot m \cdot d)$ time to construct Z_d, where n is the number of data instances, m is the length of each data instance, and d is the number of landmark vectors. The benefits from representing

$\hat{K} = Z_d Z_d^T$ in both methods are twofold: (i) significant memory savings as Z_d requires only $\mathcal{O}(n \cdot d)$ space, where n is the number of data instances and d is the number of landmark vectors; and (ii) substantial computational savings during the learning process.

An in-depth evaluation of Nyström and RFF [Yang et al., 2012] shows impressive theoretical and empirical benefits for Nyström in comparison to RFF. Considering also the difficulty to generalize RFF on arbitrary kernel functions, we rely on Nyström to efficiently learn representations for time series.

4.2.6 Problem Definition

We address the problem of efficiently learning data-aware, low-dimensional representations that preserve the invariances offered by a given kernel function, $k(\vec{x},\vec{y})$, for any $\vec{x}$ and $\vec{y}$ in the original high-dimensional space. Considering that existing time-series representation methods cannot natively preserve important invariances, such as shift and warp (see Section 4.2.3), we focus on learning representations for time-series kernel functions. Furthermore, the learned representations satisfy the following principles: ($\mathcal{P}1$) preserve similarities (i.e., $\langle Z_k(\vec{x}), Z_k(\vec{y})\rangle \approx k(\vec{x},\vec{y})$); ($\mathcal{P}2$) offer lower bounding measure (i.e., $ED(Z_k(\vec{x}), Z_k(\vec{y})) < D_k(\vec{x},\vec{y})$); ($\mathcal{P}3$) permit operations over $Z_r = [Z_k(1), \ldots, Z_k(r)]$, with $r < k$; ($\mathcal{P}4$) support efficient and memory tractable similarity computation of available and new data; and ($\mathcal{P}5$) support efficient and memory tractable eigendecomposition of Gram matrix K. Now, we introduce GRAIL, our generic, data-aware, and scalable representation learning framework.

4.3 The GRAIL Framework

Our objective is to simplify and unify the design of scalable time-series mining methods by automatically constructing representations that natively preserve the invariances offered by any chosen comparison method. This is fundamentally different from the time-series literature where representation methods are agnostic to the similarity function used in subsequent learning processes [Esling and Agon, 2012]. To achieve this goal, we present GRAIL, a generic framework for efficient representation learning given a user-specified kernel function. GRAIL representations satisfy five principles, namely, principles $\mathcal{P}1$ through $\mathcal{P}5$ in Section 4.2.6, to seamlessly integrate with and accelerate existing kernel and time-series mining methods. Considering that the most successful distance measures for time series offer shift- or warp-invariances [Wang et al., 2013; Esling and Agon, 2012], we first develop an efficient Warp-INvariant Kernel (WINK) function to compare time series (Section 4.3.1). Given a kernel function, GRAIL proceeds in three steps. First, GRAIL computes a dictionary of landmark time series that effectively summarizes available data using the k-Shape clustering algorithm of Chapter 3 (Section 4.3.2). Second, GRAIL estimates necessary parameters such that both the variance of the data in the low-dimensional space and the compactness of the learned representations are maximized (Section 4.3.3). Third, given the landmark time series and the estimated parameters, GRAIL constructs data-aware representations by expressing each time series as a linear combination of the landmark time series (Section 4.3.4).

4.3.1 Warp-Invariant Kernel for Time Series

As discussed before, comparing time-series under shift- and warp-invariances is of critical importance for effective time-series analysis. GRAIL interacts with time series using kernel functions that satisfy the positive semi-definiteness property [Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002], which we discuss next. Unfortunately, successful methods for time-series comparison under these invariances, such as SBD and DTW (see Section 4.2.3), cannot construct, based on their current formulation, proper kernel functions [Wachman et al., 2009; Cuturi, 2011]. Importantly, state-of-the-art kernel functions for time series, such as the Global Alignment (GA) kernel [Cuturi et al., 2007; Cuturi, 2011], scale quadratically with the time-series length and, therefore, are prohibitively expensive to use in practice. Our novel WINK function is intended to circumvent this efficiency limitation.

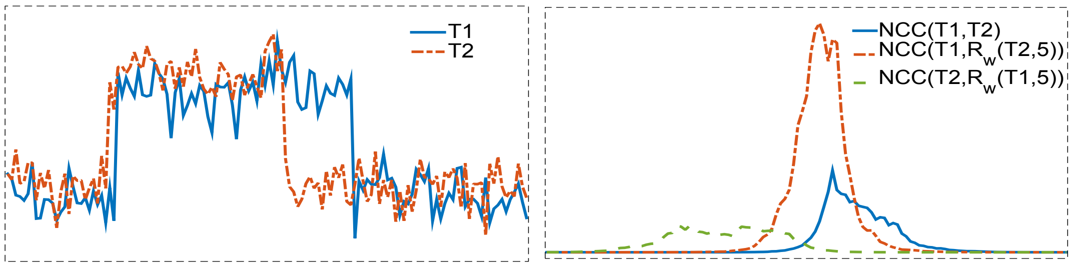
The positive semi-definiteness property of kernel functions (often referred to as Mercer's condition [Mercer, 1909]) constitutes a critical condition for the existence of RKSH (see Section 4.2.4) and leads to convex solutions for many learning problems involving kernels [Cortes and Vapnik, 1995; Bach and Jordan, 2005]. In simple terms, a symmetric function $k(\cdot,\cdot)$ is a kernel if its Gram matrix K, formed by a finite set of data points, has non negative eigenvalues. Unfortunately, for many functions it is challenging to show that they satisfy Mercer's condition. For example, it required multiple attempts to define a proper DTW-like kernel function, such as GA [Shimodaira et al., 2002; Bahlmann et al., 2002; Haasdonk and Bahlmann, 2004; Cuturi et al., 2007; Cuturi, 2011]. To design WINK, we rely on Haussler's convolution kernel [Haussler, 1999], a seminal framework for engineering new positive semi-definite kernels [Shin and Kuboyama, 2008].

The central idea behind convolution kernels is to infer the similarity of two data objects based on the similarity of their parts. Specifically, convolution kernels combine positive semi-definite kernels (i.e., return the sum of the values of the kernels) computed on all pairs of the components of the composite data objects. More formally, we assume that a time series $\vec{x} \in \mathcal{X}$ is composed of P parts $x_p \in \mathcal{X}_p$. Let us also assume that a kernel $k' : \mathcal{X}_p \times \mathcal{X}_p \rightarrow \mathbb{R}$ exists and that a relation $R(\vec{x})$ denotes the possible ways in which we can transform $\vec{x}$ into $x_1, \ldots, x_P$. Then, we can define a kernel $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ as follows [Haussler, 1999; Hofmann et al., 2008; Shin and Kuboyama, 2008]:

$$k(\vec{x},\vec{y}) = \sum_{(x',\vec{x}) \in R(\vec{x})} \sum_{(y',\vec{y}) \in R(\vec{y})} k'(x',y') \tag{4.8}$$

Therefore, we need to define two relations, $R_s(\vec{x})$ and $R_w(\vec{x})$, to decompose time series in such a way so that WINK can be shift- and warp-invariant. For $R_s(\vec{x})$, we follow [Wachman et al., 2009] to construct a shift-invariant kernel. For $R_w(\vec{x})$, we rely on efficient resampling to stretch time series [Smith, 2007].

Shift-invariant kernel: $R_s(\vec{x})$ needs to decompose time series $\vec{x}$ into all the shifted versions of $\vec{x}$. By using a circular shift operator over $\vec{x}$ as our relation, we can rearrange the coordinates of $\vec{x} = (x_1, \ldots, x_m)$ as follows: $R_s(\vec{x}, S) = (x_{(1+S) \mod m}, \ldots, x_{(m+S) \mod m})$. This



(a) Time series $T1$ and $T2$ (b) NCC sequences

Figure 4.2: Examples of NCC sequences produced by WINK.

process closely relates to how SBD (see Section 4.2.3) finds the shift that maximizes the inner product between $\vec{y}$ and $R_s(\vec{x},S), \forall S \in [1,m]$. Instead of finding the best shift, [Wachman et al., 2009] proposed to weight higher (lower) the inner products in shifted positions with large (small) values by setting $k'(\vec{x},\vec{y},\gamma) = \exp^{\gamma\langle\vec{x},\vec{y}\rangle}$, where $\gamma \geq 0$, in Equation 4.8. Finally, by using the same normalization and acceleration mechanism as with SBD, k_s compares time series in $\mathcal{O}(m \cdot log(m))$ time as follows:

$$k_s(\vec{x},\vec{y},\gamma) = \sum e^{\gamma NCC(\vec{x},\vec{y})} \tag{4.9}$$

where $NCC(\vec{x},\vec{y}) = \frac{\mathcal{F}^{-1}\{\mathcal{F}(\vec{x})*\mathcal{F}(\vec{y})\}}{||\vec{x}||\cdot||\vec{y}||}$. This formulation might result in a Gram matrix with off-diagonal values higher than values on the diagonal (i.e., a time series may be more similar to another time series than to itself), which contradicts how methods often operate over similarity matrices. Instead, we define our Shift-INvariant Kernel (SINK) as follows:

$$k'_s(\vec{x},\vec{y},\gamma) = \frac{k_s(\vec{x},\vec{y},\gamma)}{\sqrt{k_s(\vec{x},\vec{x},\gamma)\cdot k_s(\vec{y},\vec{y},\gamma)}} \tag{4.10}$$

The normalizations we introduced in Equations 4.9 and 4.10 are critical for the effectiveness of SINK. Specifically, the normalization of the cross-correlation sequence in Equation 4.9 eliminates numerical issues that arise due to the exponentiation. Importantly, the normalization in Equation 4.10 leads to statistically significant improvements in accuracy of SINK in comparison to k_s (see Sections 4.5 and 4.6).

Warp-invariant kernel: $R_w(\vec{x})$ needs to decompose time series $\vec{x}$ into all the warped versions of $\vec{x}$. In contrast to $R_s(\vec{x},S)$, where the number of all shifts to compute is small (i.e., $S \in [1,m]$), for $R_w(\vec{x})$, there are exponentially many warping paths to consider (see Section 4.2.3). To design a computationally tractable relation, we focus on a simpler form of warping: stretching of time series. Specifically, we can generate new coordinates for time series within a range of known coordinates through interpolation. Considering that $R_s(\vec{x},S)$ already computes $\mathcal{F}(\vec{x})$ and $\mathcal{F}(\vec{y})$, we rely on a simple, yet effective, interpolation method [Smith, 2007] that inserts zeros in the middle, $m_{mid} = \lceil \frac{m+1}{2} \rceil$, of the Fourier coordinates of a time series. Specifically, if $\mathcal{F}(\vec{x}) = (X_1,\dots,X_m)$, we generate a time series of length m' using $R_w(\vec{x},m') = \mathcal{F}^{-1}(X_{(1:m_{mid})}, 0_{(1:m'-m)}, X_{(m_{mid}+1:m)})$ and define a warp-invariant kernel k_w based on SINK, as follows:

$$k_w(\vec{x},\vec{y},\gamma) = \sum_{(x',\vec{x})\in R_w(\vec{x},\cdot)} \sum_{(y',\vec{y})\in R_w(\vec{y},\cdot)} k_s'(x',y',\gamma) \tag{4.11}$$

This formulation of k_w provides the flexibility to pose restrictions on the possible m' values in $R_w(\vec{y},m')$ to consider (i.e., the number of different stretched versions of each time series). To design a highly efficient warp-invariant kernel, we retain the original time series along with its stretched version for a user-specified parameter m', as follows:

$$k_w'(\vec{x},\vec{y},\gamma,m') = \sum_{i,j\in\{0,m'\}} k_s'(R_w(\vec{x},i),R_w(\vec{y},j),\gamma) \tag{4.12}$$

Algorithm 8 shows how we can compute WINK in a few lines of code using modern mathematical software. In Figure 4.2, we display an example of how WINK constructs cross-correlation sequences (Figure 4.2b) for a time series $T2$ that requires stretching to better match the shape of $T1$ before comparison (Figure 4.2a). To guarantee that WINK is a symmetric function (i.e., $k_w'(\vec{x},\vec{y},\gamma,m') = k_w'(\vec{y},\vec{x},\gamma,m')$), we consider stretched versions of both time series. We observe that the values of the cross-correlation sequence where only $T2$ is stretched (dotted-and-dashed line) takes higher values than the version where only $T1$

is stretched (dashed line) or the version where none of the time series are stretched (solid line). In particular, for low values of γ (e.g., 1 or 2), all three cross-correlation sequences contribute similarly to the final kernel value of WINK, whereas for high values of γ (e.g., 10 or 20), one cross-correlation sequence dominates the other two and, therefore, contributes more than the other two to the final kernel value. We now turn to the critical problem of computing a dictionary of landmark time series.

4.3.2 Time-Series Dictionary Learning

Given a kernel function, the first step of GRAIL is to compute a dictionary of landmark time series that Nyström requires in order to express time series as a linear combination of landmark time series (see Section 4.2.5). Unfortunately, the selection of landmark time series is a difficult combinatorial problem with significant impact on the quality and compactness of the learned representations. Due to this difficulty, a large body of work focuses on empirical and theoretical analysis of sampling and projection methods for Nyström [Williams and Seeger, 2001; Drineas and Mahoney, 2005; Kumar et al., 2012; Gittens and Mahoney, 2013]. The most popular sampling method for Nyström is uniform sampling without replacement (i.e., random sampling) [Williams and Seeger, 2001]. Interestingly, random sampling produces effective representations with minimal time and space overhead in comparison to more expensive sampling and projection methods [Kumar et al., 2012; Gittens and Mahoney, 2013]. Importantly, random sampling operates independently of the kernel function of choice and, therefore, is an appealing method to construct landmark time series.

However, as noted earlier, the careful selection of landmark time series reduces the approximation error for Nyström and leads to more compact representations. Therefore, for GRAIL, we explore the value of using the centroids of time-series clustering methods to construct landmark time series. Recent empirical and theoretical analysis shows that the approximation error for Nyström is bounded by the quantization error of mapping each

Algorithm 8: Warp-Invariant Kernel

```
Input  : $\vec{x}$ is a 1-by-m $z$-normalized time series
         $\vec{y}$ is a 1-by-m $z$-normalized time series
         $\gamma$ is a scaling parameter
         $m'$ denotes percentage of stretching for $\vec{x}$ and $\vec{y}$
Output: $Sim$ is the kernel value between $\vec{x}$ and $\vec{x}$
function $Sim$ = $WINK(\vec{x},\vec{y},\gamma,m')$:
    $cc = WNCC(\vec{x},\vec{y},m')$ ;
    $Sim = 2 \cdot sum(exp(\gamma \cdot cc(1,:)))$ ;
    for $i \leftarrow 2$ to $size(cc,1)$ do
        $Sim = Sim + sum(exp(\gamma \cdot cc(i,:)))$ ;
def $cc$ = $WNCC(\vec{x},\vec{y},m')$:
    $Lenx' = ceil(len(\vec{x}) \cdot \frac{100+m'}{100})$ ;
    $FFTL = 2^{nextpower2(2*Lenx'-1)}$ ;
    $X = FFT(\vec{x},FFTL)$ ;
    $Y = FFT(\vec{y},FFTL)$ ;
    $cc = NCC(X,\vec{x},\vec{y},FFTL)$ ;
    if $m' \geq 0$ then
        $cc2 = NCC(X,\vec{x},Interp(Y,Lenx'),FFTL)$ ;
        $cc3 = NCC(X,\vec{y},Interp(X,Lenx'),FFTL)$ ;
        $cc = [cc,cc2,cc3]$ ;
def $cc$ = $NCC(X,\vec{x},\vec{y},FFTL)$:
    $cc = IFFT(X * FFT(\vec{y},FFTL))$ ;
    $cc = \frac{cc}{norm(\vec{x}) \cdot norm(\vec{y})}$;
def $x'$ = $Interp(X,m')$:
    $k = ceil(\frac{len(X)+1}{2}$ ;
    $X' = [X(1:k), zeros(1,m'-len(X)), X(k+1:len(X))]$ ;
    $x' = real(IFFT(X')) \cdot \frac{m'}{len(X)}$ ;
```

vector $\vec{x}_i \in X$ to the closest d landmark vectors $\vec{g}_i \in G$ [Zhang et al., 2008]:

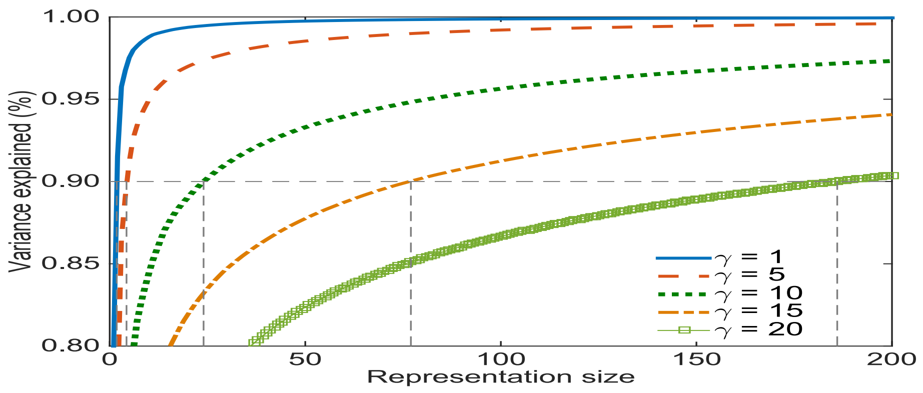
$$||K - EW^{-1}E^T||_F \leq \sum_{i=1}^{n} ||\vec{x}_i - g_{map(i)}||^2 \tag{4.13}$$

where $map(i) = \arg\min_{j=1,\ldots,d} ||\vec{x}_i - \vec{g}_j||^2$. Clustering methods relying on the principles of k-means [MacQueen, 1967] minimize the exact same quantization error and, therefore, the cluster centroids can serve as landmark vectors for Nyström. Unfortunately, k-means is not suitable to cluster time series as it cannot compare time series under shift- or warp-invariance. Instead, k-Shape efficiently clusters time series under shift-invariance and the cluster centroids effectively summarize time series for distance measures offering shift and warp invariances, as we saw in Chapter 3. Therefore, for GRAIL, we use the cluster centroids of k-Shape as landmark time series. Our extensive evaluation (see Sections 4.5 and 4.6) corroborates earlier findings on the effectiveness of random sampling [Kumar et al., 2012; Gittens and Mahoney, 2013] but, importantly, shows that k-Shape significantly outperforms all state-of-the-art methods for selecting landmark vectors.

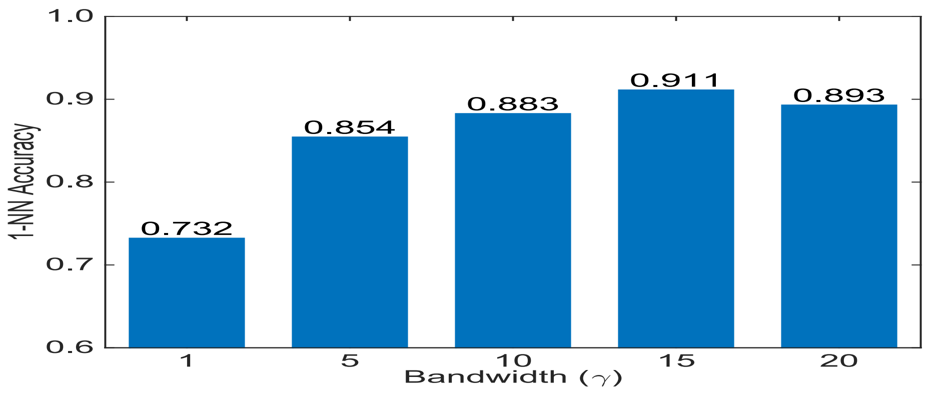
Next, we discuss how we can rely on the landmark time series to estimate necessary parameters for GRAIL.

4.3.3 Parameter Tuning

After the construction of landmark time series, GRAIL proceeds with the estimation of kernel function parameters, which affect (i) the compactness and (ii) the quality of the learned representations. To illustrate this point, Figure 4.3 explores how different γ values of WINK, given a fixed m' value, affect those two factors for the StarLightCurve dataset [Keogh et al., 2015]. Specifically, Figure 4.3a measures the cumulative variance explained in varying representation sizes for different γ values. Figure 4.3b presents the 1NN classification accuracies of representations capturing 90% of the variance in time series for different γ values. We observe that small γ values require only a few eigenvectors to explain 90% of the variation in the similarities of time series, whereas large γ values require significantly more eigenvectors (i.e., small γ values lead to more compact representations). On the other hand, when all representations explain the same targeted amount of variance in data, different γ values lead to significant differences in the quality of the representations for various tasks, such as for classification in Figure 4.3b. Therefore, selecting small γ values due to the storage and computation benefits might lead to significant reductions in accuracy. Due to this difficulty in tuning kernel function parameters, the majority of the literature relies on supervised tuning of such parameters, which is not realistic for the unsupervised setting we consider in this work.



(a) Cumulative variance



(b) Classification accuracies

Figure 4.3: Comparison of the compactness and the classification accuracy of learned representations with different γ values for WINK for the StarLightCurve dataset.

For GRAIL, we propose a simple, yet effective, method to estimate the bandwidth γ of WINK that relies on two key observations. First, considering that some loss of information is unavoidable in the low-dimensional space, time-series similarities with large variance are much more likely to be preserved in the low-dimensional space than time-series similarities with low variance. Therefore, we want to select γ values such that the variance is maximized. Second, to determine how effectively we can capture the variance in the low-dimensional space, we need to consider how much of the variance each eigenvalue explains. Therefore, we want to select γ values such that for a small number of r eigenvalues, such as $r = 10$, the variance explained is maximized. Our method combines those two observations in a scoring function to permit effective selection for γ given an m' value. Specifically, considering the eigendecomposition of matrix $W = Q_W \Lambda_W Q_W^T$, where W consists of the pairwise kernel

Algorithm 9: Parameter Tuning

```
Input : $G$ is a $d$-by-$m$ matrix containing $d$ landmark sequences
        $GV$ is a 1-by-$l$ vector containing $l$ values for $\gamma$
        $m'$ denotes percentage of stretching
Output: $score$ is the score of selected $\gamma$
        $gamma$ is the selected kernel scaling parameter
function [score, gamma] = GammaSel($G,GV,m'$):
    foreach $\gamma \in GV$ do
        $Wtmp = []$;
        for $i = 1$ to $d$ do
            for $j = 1$ to $d$ do
                $W(i,j) = WINK(G(i,:),G(j,:),\gamma,m')$;
            $Wtmp = [Wtmp, W(i,:)]$;
        $GVar(\gamma) = var(Wtmp)$;
        $[Q,L] = eig(W)$;
        $VarTop10(g) = \frac{sum(L(1:10))}{sum(L)}$;
    $[score, gamma] = max(GVar \cdot VarTop10)$;
```

values of the landmark time series (see Section 4.3.2), we compute the scoring function of each γ value as follows:

$$Score(\gamma,m') = var(W,\gamma,m') \cdot \frac{\sum_{i=1}^{r} \Lambda_W(i)}{\sum_{i=1}^{d} \Lambda_W(i)} \tag{4.14}$$

where $var(W,\gamma,m')$ denotes the variance of the kernel values in W computed with WINK using γ and m'. Algorithm 9 shows how we can tune parameters for WINK according to our scoring function (Equation 4.14). In our extensive evaluation (Sections 4.5 and 4.6), we show that our lightweight method for parameter tuning leads to representations with similar compactness and accuracy to representations tuned in a supervised manner. In the next section, we show how GRAIL learns time-series representations.

4.3.4 Time-Series Representation Learning

We now present GRAIL, our generic representation learning framework. Prior to learning any representations, GRAIL chooses (i) a kernel function for time-series comparison, such as WINK (Section 4.3.1); (ii) a method for constructing landmark time series, such as random selection of landmark time series from available data or through the use of time-series clustering (Section 4.3.2); and (iii) a method to estimate necessary parameters, such as our method for parameter tuning for kernel functions (Section 4.3.3). Then, GRAIL proceeds in two steps: (i) constructs a temporal representation, Z_d, where d is the number of landmark time series, to approximate the Gram matrix $\hat{K} = Z_d Z_d^\top$ using the Nyström method (Section 4.2.5); and (ii) exploits a matrix sketching algorithm over Z_d, namely, the Frequent Directions (FD) method [Liberty, 2013], to efficiently approximate the eigendecomposition of $\hat{K}$ and learn the final representation Z_k, where $k \leq d$, which satisfies principles $\mathcal{P}1$ through $\mathcal{P}5$ in Section 4.2.6.

Approximation of $\hat{K}$: As noted earlier, Nyström requires as input two matrices: (i) matrix $W \in \mathbb{R}^{d \times d}$, which contains all pairwise similarities between landmark time series; and (ii) matrix $E \in \mathbb{R}^{n \times d}$, which contains pairwise similarities between the n time series and the d landmark time series. Once we construct those two matrices, Nyström computes the inverse of $W^{-1} = Q_W \Lambda_W^{-1} Q_W^{-1}$ and constructs Z_d as follows:

$$Z_d = E Q_W \Lambda_W^{-0.5} \tag{4.15}$$

The dimensionality d of Z_d corresponds to the number of landmark time series extracted from the entire dataset and, therefore, d is significantly smaller than the size n of the entire dataset. Unfortunately, for very large datasets, Nyström might require to use thousands of landmark time series to accurately approximate K. In such cases, the dimensionality of the learned representation might surpass the dimensionality of the original time series, which defeats the original purpose of producing low-dimensional representations.

Representation Learning: To eliminate this issue, we propose to use FD to approximate the eigendecomposition of K using Z_d and, subsequently, reduce the dimensionality from d to k by retaining only the top eigenvalues and eigenvectors of K. An alternative approach is to directly exploit the eigendecomposition of matrix W, which Nyström computes to construct Z_d, to approximate the eigendecomposition of K. Unfortunately, this approach offers a poor approximation of the eigendecomposition of K. Importantly, it may lead to non-orthogonalized eigenvectors of K, as required by definition (see Section 4.2.1), which impacts the effectiveness of learned representations. Therefore, assuming the eigendecomposition of $K = Q\Lambda Q^T$, we can approximate $\hat{Q} = Z_d Q_C \Lambda_C^{-0.5}$ in $\mathcal{O}(nd^2)$, where $Q_C \in \mathbb{R}^{d\times d}$ and $\Lambda_C \in \mathbb{R}^{d\times d}$ are the eigenvectors and eigenvalues of $C = Z_d^T Z_d \in \mathbb{R}^{d\times d}$. Unfortunately, for large number d of landmark time series, such computation becomes prohibitively expensive. We rely on FD to approximate matrix C by retaining only a sketch of size b of Z_d in $\mathcal{O}(ndb)$, where $b \leq d \leq n$, and we construct $Z_k = \hat{Q}_{1:n,1:k}\Lambda_{C1:k,1:k}$. This procedure leads to a number of benefits: (i) guarantees that Q remains orthonormal and, therefore, permits lower bounding of the original kernel function ($\mathcal{P}2$); (ii) permits the estimation of the variance explained in each coordinate of Z_k and, therefore, we can reuse part of the coordinates of Z_k ($\mathcal{P}3$); and (iii) enables efficient approximation of the eigenvalues and eigenvectors of K and, therefore, seamless integration with existing algorithms relying on such cornerstone operation.

Algorithm 10 shows how to learn representations using GRAIL with just a few lines of code using modern mathematical software. GRAIL expects as input the time-series collection, the number of landmark time series to construct, two vectors with values for γ and m' to perform a grid search and estimate suitable parameters for WINK, and the fraction of the variance to explain in low-dimensional space in order to determine the final dimensionality of Z_k. Alternatively, the algorithm can be easily modified to get a user-specified value for the dimensionality of Z_k.

Next, we review how GRAIL representations integrate with and, therefore, accelerate existing kernel methods.

4.4 Time-Series Mining with GRAIL

In this section, we describe how to use GRAIL representations to perform major time-series mining tasks. Specifically, we focus on five tasks, namely, querying and indexing, classification, clustering, sampling, and visualization.

Algorithm 10: Representation Learning with GRAIL

```
Input  : $X$ is an $n$-by-$m$ matrix containing $n$ time series
         $k$ is the number of landmark time series to extract
         $f$ is a scalar used to estimate the dimensionality $dim$ of $Z_k$
         $GV$ is a 1-by-$l$ vector containing $l$ values for $\gamma$
         $R$ is a 1-by-$p$ vector containing $p$ values for $m'$
Output: $Z_k$ is an $n$-by-$dim$ matrix
function $Z_k = GRAIL(X, k, GV, R, f)$:
    $D = k-Shape(X,k)$;
    for $i = 1$ to $p$ do
        $[gscore(i), gvalue(i)] = GammaSel(D, GV, i)$;
    $[maxscore, pos] = max(gscore)$ ;
    $BG = gvalue(pos)$ ;
    $BW = pos$ ;
    for $i = 1$ to $k$ do
        for $j = 1$ to $k$ do
            $W(i,j) = WINK(D(i,:), D(j,:), BG, BW)$ ;
    for $i = 1$ to $n$ do
        for $j = 1$ to $k$ do
            $E(i,j) = WINK(X(i,:), D(j,:), BG, BW)$ ;
    $[Q,L] = eig(W)$ ;
    $Z_d = E * Q * L^{-0.5}$;
    $b = 0.5 * d$;
    $B = zeros(b,d)$;
    for $i = 1$ to $n$ do
        $B = [B; Z_d(i,:)]$;
        if $B$ has no zero valued rows then
            $[U,S,V] = SVD(B)$ $B = sqrt(max(0, S^2 - S^2_{\frac{b}{2},\frac{b}{2}} * I_b)) * V^T$;
    $[Q,L] = EIG(B^T * B)$;
    $U = Z_d * Q * L^{-0.5}$;
    $dim = find(\frac{cumsum(L)}{sum(L)} > f)$;
    $Z_k = U(:,1:dim) * L(1:dim,1:dim)^{-0.5}$;
```

Querying and Indexing: GRAIL representations satisfy two important properties, namely, $\mathcal{P}1$ and $\mathcal{P}2$, to facilitate natively querying and indexing of time series. In particular, GRAIL constructs Z_k such that in low-dimensional metric space it preserves the similarities of time series in the original high dimensional space (i.e., $\langle Z_k(\vec{x}), Z_k(\vec{y})\rangle \approx WINK(\vec{x},\vec{y})$) and, importantly, permits to use ED in the low-dimensional space as a lower bounding measure of $D_k(\vec{x},\vec{y}) = WINK(\vec{x},\vec{x}) + WINK(\vec{y},\vec{y}) - 2WINK(\vec{x},\vec{y})$ in the original high-dimensional space (i.e., $ED(Z_k(\vec{x}), Z_k(\vec{y})) < D_k(\vec{x},\vec{y})$). Therefore, GRAIL representations can exploit the GEMINI framework and existing indexing mechanisms [Esling and Agon, 2012] to organize and efficiently retrieve time series given a time-series query.

Classification: GRAIL representations approximate the Gram matrix K as $K = Z_k Z_k^T$, which enables the use of the Woodbury formula [Williams and Seeger, 2001] to invert K efficiently as follows:

$$(K - \sigma I)^{-1} = \frac{1}{\sigma}(I - Z_k(\sigma \cdot I + Z_k^T Z_k)^{-1} Z_k^T) \tag{4.16}$$

where $\sigma \geq 0$ is a regularization parameter and I is the identity matrix. Many algorithms rely on the this expensive operation, including the least-squares Support Vector Machines (SVM) [Suykens et al., 1999] classification algorithm. With the Woodbury formula we can significantly reduce the cost to invert K from $\mathcal{O}(n^3)$ to $\mathcal{O}(nk^2)$, a substantial reduction. Importantly, as Z_k is embedded in a low-dimensional metric space, we can consider Z_k as a matrix containing feature vectors in its rows. Such interpretation enables the use of any existing linear method for classification, such as logistic regression and linear SVM [Fan et al., 2008].

Clustering: In the process of producing Z_k, GRAIL also approximates the eigenvalues and eigenvectors of matrix K. Many algorithms rely on the eigendecomposition of K, including the seminal spectral clustering algorithm [Ng et al., 2002]. Therefore, we can easily exploit the approximation of Q produced by GRAIL and provide it as input to vanilla k-means algorithm, which results in a very efficient computation of spectral clustering without ever having to construct the Gram matrix K.

Sampling: A similar methodology applies to sampling. Determinantal Point Processes (DPP) [Kulesza and Taskar, 2010; Affandi et al., 2013] are appealing models to select a subset of time series from a dataset when diversity is desired. Specifically, a DPP defines a distribution over subsets of time series of a base set of all the time series in the dataset and randomly selects a subset, which is likely to contain dissimilar time series. The dissimilarity between time series is measured by the Gram matrix K. Once K is given, DPP computes the eigenvalues and eigenvectors of K. A dual representation for DPP was proposed in [Kulesza and Taskar, 2010], which enables the use of representations such as Z_k that satisfies $K = Z_k Z_k^T$. Therefore, as in the case of spectral clustering, we exploit the approximation of Q produced by GRAIL and provide it as input to the dual DPP formulation of the problem.

Visualization: Last, we explore the use of methods to visualize datasets. The most prominent such method in the literature is the KPCA algorithm that the GRAIL framework relies on. Therefore, by using GRAIL-FD we can get for free the approximation of Q and visualize the projection of data in the top two eigenvectors of Q to explore the structure of time-series datasets.

We now turn to the experimental evaluation of the factors determining the effectiveness of GRAIL and on the aforementioned time-series mining tasks.

4.5 Experimental Settings

In this section, we review in detail the settings for the evaluation of (i) our kernel function, WINK; (ii) k-Shape as a dictionary learning algorithm; (iii) our parameter selection approach; (iv) GRAIL representations; and (v) five time-series mining methods that exploit GRAIL representations.

Datasets: As in Chapter 3, we use the largest public collection of class-labeled time-series datasets, namely, the UCR collection [Keogh et al., 2015]. It consists of 85 datasets, both synthetic and real, which span several different domains. (Appendix C provides the exact names of the 85 datasets used in our analysis along with some relevant characteristics.) Each dataset contains from 40 to $16,637$ sequences. The sequences in each dataset have equal length, but from one dataset to another the sequence length varies from 24 to $2,709$. These datasets are annotated and every sequence can belong to only one class. In the context of clustering, and for the sake of convenience, the class label for a sequence is often interpreted as identifying the cluster where the sequence belongs. Furthermore, the datasets are already z-normalized and split into training and test sets. As we will see, we use this split of the datasets to evaluate classification algorithms.

Platform: We ran our experiments on a cluster of 302 servers with an identical configuration: Dual Intel Xeon E5-2650v4 (12-core with 2-way SMT) processor with clock speed at 2.2 GHz and up to 256 GB RAM. Each server runs Red Hat Enterprise Linux 6.6 (64-bit) and Matlab R2014a (64-bit).

Implementation: We implemented all approaches under the same framework, in Matlab, for a consistent evaluation in terms of both accuracy and efficiency. For repeatability purposes, we make all datasets and source code available.[1]

Baselines: We evaluate WINK, our kernel function; our k-Shape clustering method as a dictionary learning algorithm; our parameter tuning method (GRAIL-PT); our GRAIL representations (GRAIL+Rep); our GRAIL-LB method to lower bound WINK; our GRAIL-SVM linear classification method; our GRAIL-SC spectral clustering algorithm; our GRAIL-DPP sampling method; and our GRAIL-KPCA visualization method.

Specifically, we compare WINK against the strongest state-of-the-art distance measures (see Section 2.3) and kernel functions for time series:

- **ED:** a simple, efficient — yet accurate — distance measure [Faloutsos et al., 1994]
- **SBD:** our efficient and parameter-free distance measure developed in Chapter 3
- **cDTW:** the constrained version of DTW, with improved accuracy and efficiency

[1]`http://www.cs.columbia.edu/~jopa/grail.html`

[Sakoe and Chiba, 1978]

- **GA:** a warp-invariant global alignment kernel [Cuturi, 2011]

Following [Ding et al., 2008; Wang et al., 2013], we use the 1NN classifier, which is a simple and parameter-free classifier, to evaluate distance measures. Importantly, 1NN classifiers, combined with elastic distance measures, such as cDTW, achieve state-of-the-art classification performance [Xi et al., 2006; Wang et al., 2013; Bagnall et al., 2017]. For kernel functions, we use the Support Vector Machines (SVM) classifier [Cortes and Vapnik, 1995] as implemented by [Chang and Lin, 2011] to evaluate WINK (SVM+WINK) and GA (SVM+GA) kernels. Additionally, we compare these approaches to the recently proposed Ensemble Elastic (EE) classifier that combines 11 1NN classifiers using several elastic distance measures [Lines and Bagnall, 2015].

We compare k-Shape against state-of-the-art sampling and projection methods developed or used to compute landmark vectors for Nyström. Specifically, we consider the following sampling methods:

- **Random:** a simple and efficient uniform sampling method [Williams and Seeger, 2001]

- **AFKMC2:** an approximate version of the k-means++ sampling method [Bachem et al., 2016]

- **GibbsDPP:** an approximate version of the DPP method that relies on Gibbs sampling [Li et al., 2016]

- **LevScore:** a non-uniform sampling method based on leverage scores [Gittens and Mahoney, 2013]

In addition to the sampling methods, we consider two projection methods: (i) a projection method to sketch Gram matrix K based on the Fourier transform (SRFT) [Gittens and Mahoney, 2013]; and (ii) a projection method to sketch Gram matrix K based on a Gaussian Process method (GP) [Gittens and Mahoney, 2013].

We evaluate GRAIL-PT, our parameter tuning method, along with three other approaches for kernel function parameter estimation:

- **MinVariance:** a simple method to always select kernel function parameters such that the variance in data is minimized
- **MaxVariance:** a simple method to always select kernel function parameters such that the variance in data is maximized
- **LOOCAcc:** a supervised method to always select kernel function parameters such that the leave-one-out classification accuracy is maximized

We compare GRAIL+Rep, our learned representations, against representation methods learned using the exact KPCA method (KPCA+Rep), the optimal representation learning method for kernel functions. In both approaches, Rep denotes the learned representation. For example GRAIL+Z95per denotes the compact GRAIL representation, Z_k, that explains 95% of the variance in Z_d, the GRAIL representation constructed using d landmark time series. Similarly, KPCA+Z95per denotes the representation that explains 95% of the variance in time series.

To understand the performance of GRAIL representations in practice, we evaluate learned representations, when combined with suitable kernel methods (see Section 4.4), in five major time-series mining tasks, namely, querying and indexing, classification, clustering, sampling, and visualization. Specifically, for querying and indexing, we compare GRAIL-LB, our lower bounding method for WINK, against three state-of-the-art lower bounding methods for ED and cDTW distance measures:

- **DFT-LB:** a method that uses the Fourier transform to represent time series and uses the first-k fourier coefficients to lower bound ED [Faloutsos et al., 1994]
- **Keogh-LB:** a lower bounding method for cDTW [Keogh and Ratanamahatana, 2005] [Sakoe and Chiba, 1978]

GRAIL-LB and DFT-LB operate over low-dimensional representations, whereas Keogh-LB operates over the original raw time-series representation.

For classification, we evaluate GRAIL-SVM, our classifier that exploits linear SVM, against the classifiers we considered above to evaluate WINK. Specifically, we compare GRAIL-SVM against the three 1NN classifiers with ED, SBD, and cDTW distance measures, and the two SVM classifiers with WINK and GA kernels.

For clustering, we compare SC-GRAIL, our spectral clustering method, against the two strongest clustering methods determined through our extensive experimental evaluation in Chapter 3:

- **k-AVG+ED:** the original, highly efficient, but less-accurate, k-means clustering method [MacQueen, 1967]
- **k-Shape:** our efficient and highly accurate time-series clustering method (see Chapter 3)

For sampling, we compare GRAIL-DPP, our DPP-based sampling method, against two state-of-the-art methods for approximate sampling using the k-means++ mechanism and the DPP method:

- **AFKMC2:** an approximate version of the k-means++ sampling method [Bachem et al., 2016]
- **GibbsDPP:** an approximate version of the DPP method that relies on Gibbs sampling [Li et al., 2016]

Finally, for visualization, we compare GRAIL-KPCA, our approximate KPCA-based visualization method, against the exact representations produced by KPCA over the Gram matrix K.

Parameter settings: Among the distance measures discussed above, cDTW requires setting a parameter to constrain its warping window. We compute the optimal window by performing a leave-one-out classification step over the training set of each dataset (cDTW^{opt}).

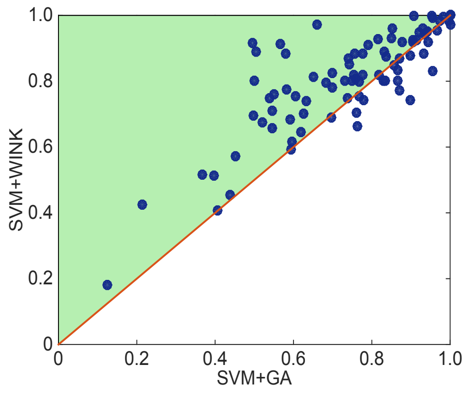
To evaluate WINK and GA kernels using SVM, we need to set a regularization parameter C. We tune the C value in the training set of each dataset using a grid search with power of two C values ranging from -10 to 20 with step 1. For both kernels, we consider scaling parameters ranging from 1 to 20. WINK also requires the tuning of the stretching parameter for time series. We consider values ranging from 0% to 10% of the length of the time series. For dictionary learning, all methods select or construct the same number of landmark time series for each dataset, which corresponds to 40% of the number of time series in each dataset. For parameter estimation, we evaluate all methods by learning representations using the exact KPCA to avoid any effects from the dictionary learning process. As before, we consider scaling parameters for WINK ranging from 1 to 20 and stretching parameters ranging from 0% to 10% of the length of the time series. For the evaluation of the learned representations, we extract landmark time series using k-Shape and GRAIL-PT to estimate parameters for WINK.

Once we evaluate the components and the learned representations of GRAIL, we focus on the evaluation of the learned representations when combined with suitable kernel methods to solve time-series mining tasks. Specifically, for querying and indexing, GRAIL-LB and DFT-LB operate over low-dimensional representations with a fixed number of coordinates, namely, 20 coordinates, across each dataset. Keogh-LB operates over raw time-series using cDTW^5, the cDTW with window 5% of the length of the time series of each dataset. For classification, we set the regularization parameter C of GRAIL-SVM using the same values for grid search as before and exploit GRAIL representations constructed using d landmark time series. For clustering, sampling, and visualization we use compact GRAIL representations, Z_k, that explain 90% of the variance in Z_d, the GRAIL representation constructed using d landmark time series. For clustering and sampling, we use the number of classes in each dataset as the number of clusters and the number of samples for the dataset, respectively.

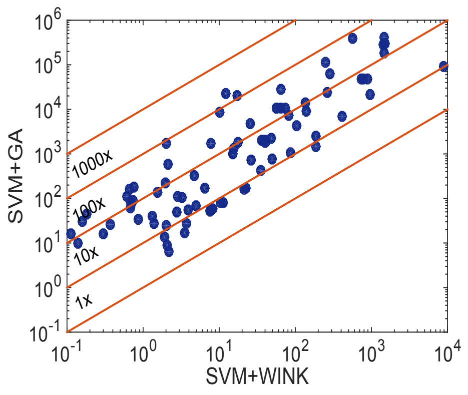
Metrics: We compare our approaches on runtime and accuracy. For runtime, we compute CPU time utilization and measure the time ratios for our comparisons for each dataset.

To evaluate classifiers, we report the classification accuracy (i.e., number of correctly classified instances over all instances) by performing classification over the training and test sets of each dataset. To evaluate representations we report the approximation error using the Frobenius norm between the original kernel matrix K and the approximated kernel $\hat{K} = ZZ^T$. To visualize our results over the 85 datasets, we employ a min-max normalization to the approximation error to bound it between 0 and 1, and report 1 minus the approximation error, as an accuracy value, to follow the convention of reporting accuracy results throughout this thesis. We use the Rand Index [Rand, 1971] to evaluate clustering accuracy over the fused training and test sets of each dataset. For the clustering and sampling methods considered in this chapter, we report the average Rand Index over 10 runs; in every run we use a different random initialization.

Statistical analysis: Following [Batista et al., 2014; Giusti and Batista, 2013], we analyze the results of every pairwise comparison of algorithms over multiple datasets using the Wilcoxon test [Wilcoxon, 1945] with a 99% confidence level. According to [Demšar, 2006], the Wilcoxon test is less affected by outliers than is the t-test [Rice, 2006], as Wilcoxon does not consider absolute commensurability of differences. Moreover, using pairwise tests to reason about multiple algorithms is not fully satisfactory because sometimes the null hypotheses are rejected due to random chance. Therefore, we also use the Friedman test [Friedman, 1937] followed by the post-hoc Nemenyi test [Nemenyi, 1963] for comparison of multiple algorithms over multiple datasets, as suggested in [Demšar, 2006]. The Friedman and Nemenyi tests require more evidence to detect statistical significance than the Wilcoxon test [Giusti and Batista, 2013] (i.e., the larger the number of methods, the larger the number of datasets required) and, hence, as we already use all 85 datasets for the Wilcoxon test, we report statistical significant results with a 95% confidence level.



(a) Classification accuracy



(b) Runtime comparison

Figure 4.4: Comparison of SVM+WINK and SVM+GA classifiers over 85 datasets.

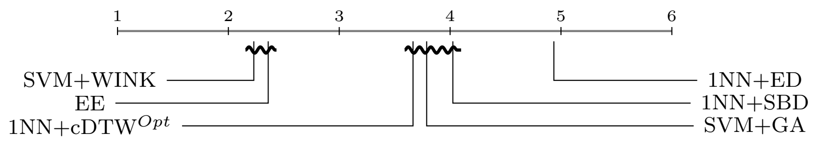


Figure 4.5: Ranking of classifiers based on the average of their ranks across datasets.

4.6 Experimental Results

In this section, we report our experimental results. Firstly, we evaluate WINK against state-of-the-art distance measures and kernel functions (Section 4.6.1). Secondly, we compare k-Shape against other dictionary learning methods (Section 4.6.2). Thirdly, we evaluate our parameter tuning method (Section 4.6.3) and our representations (Section 4.6.4). Then, we evaluate GRAIL representations on five tasks: (i) querying and indexing (Section 4.6.5); (ii) classification (Section 4.6.6); (iii) clustering (Section 4.6.7); (iv) sampling (Section 4.6.8); and (v) visualization (Section 4.6.9). Finally, we highlight our findings (Section 4.6.10).

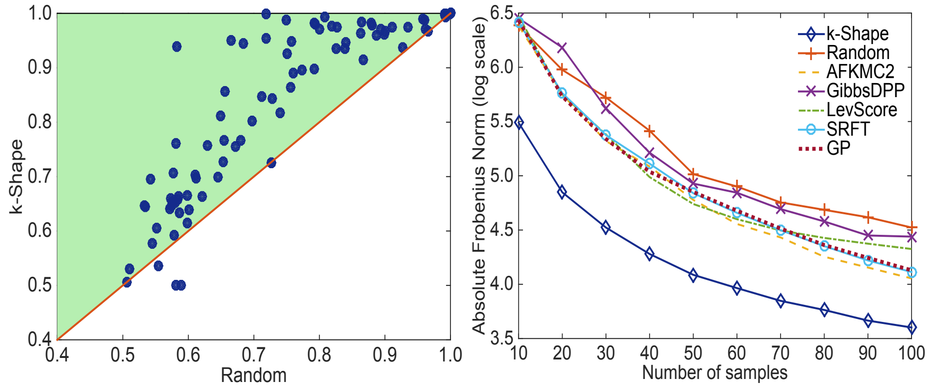
4.6.1 Evaluation of WINK Against Other Distance Measures

We evaluate WINK, our time-series kernel function discussed in Section 4.3.1, in terms of runtime and accuracy in comparison to other state-of-the-art time-series distance measures and kernel functions.

Comparison against GA: To understand if WINK is a competitive kernel function for time series comparison, we first evaluate it against GA, the state-of-the-art kernel function for time series, using the classification accuracy across 85 datasets (Section 3.5). Specifically, we compare the classification accuracy of SVM classifiers combined with the WINK and GA kernels. Figure 4.4a presents the pairwise differences in accuracy between WINK and GA where each circle represents a dataset. The first coordinate of each circle corresponds to the classification accuracy of SVM+WINK and the second coordinate of each circle corresponds to the classification accuracy of SVM+GA. SVM+WINK outperforms SVM+GA in 62 of 85 datasets (i.e., 62 circles are above the diagonal in Figure 4.4a) and the Wilcoxon test suggests this difference in accuracy is statistically significant.

Comparison against ED, SBD, and cDTW: Having shown that SVM+WINK significantly outperforms SVM+GA, we now evaluate SVM+WINK against state-of-the-art distance measures combined with the 1NN classifier. Specifically, SVM+WINK performs at least as well as 1NN+ED, 1NN+SBD, and 1NN+cDTW^{Opt} in 77, 75, and 65 datasets, respectively, and the Wilcoxon test suggests these differences in accuracy are statistically significant. In contrast, SVM+GA performs at least as well as 1NN+ED, 1NN+SBD, and 1NN+cDTW^{Opt} in 55, 45, and 42 datasets, respectively, but the Wilcoxon test suggests that only the difference in accuracy against 1NN+ED is statistically significant. Therefore, SVM+WINK is the only classifier that significantly outperforms 1NN and SVM classifiers combined with state-of-the-art distance measures. Interestingly, SVM+WINK performs at least as well as EE, an ensemble classifier that combines 11 elastic distance measures, in 43 out of 85 datasets but with no statistical significance over EE.

Statistical analysis: To better understand the performance of SVM+WINK in comparison with the other classifiers, we evaluate the significance of their differences in accuracy when considered all together. Figure 4.5 shows the average rank across datasets for each classifier. SVM+WINK is the top classifier meaning that SVM+WINK performed best in the majority of the datasets. The Friedman test rejects the null hypothesis that all classifiers behave similarly and, therefore, we proceed with a post hoc Nemenyi test to evaluate the significance of the differences in the ranks. We observe two clusters of methods whose ranks



(a) k-Shape against Random (b) Approximation error on StarLightCurve

Figure 4.6: Comparison of dictionary learning algorithms.

do not present a statistically significant difference: SVM+WINK and EE form the first cluster, while 1NN+SBD, 1NN+cDTW^{Opt}, and SVM+GA form the second cluster. The methods in the first cluster are significantly better than the methods in the second cluster and, in turn, the methods in the second cluster are significantly better than 1NN+ED.

Efficiency: WINK is the only comparison method that, when combined with SVM classifiers, significantly outperforms all standalone comparison methods combined with either SVM or 1NN classifiers. We now investigate whether this superiority in accuracy has an associated penalty in efficiency. Figure 4.4b shows how much slower GA is than WINK for each dataset. We observe that WINK is significantly faster than GA across all datasets with the differences ranging between one order of magnitude for medium-sized datasets to three orders of magnitude for datasets with very long time series. We observe similar trends against DTW variants because DTW and GA show comparable runtime performance and, therefore, we omit the results. However, SBD is two to four times faster than WINK.

4.6.2 Evaluation of k-Shape Against Other Dictionary Learning Methods

Having shown the robustness of WINK, we now evaluate the performance of k-Shape as a dictionary learning algorithm against sampling and projection methods.

Comparison against sampling methods: First, we evaluate k-Shape again Random, the simplest, yet effective, method to sample time series. Figure 4.6a compares the two

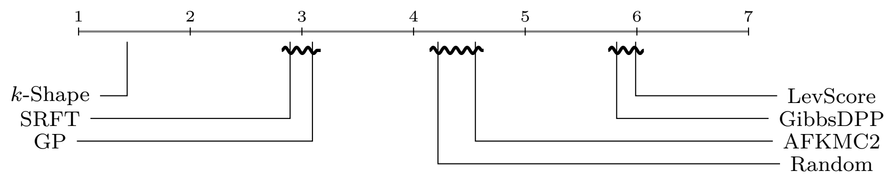


Figure 4.7: Ranking of dictionary learning algorithms based on the average of their ranks across datasets.

methods using their approximation error (transformed into accuracy as discussed in Section 4.5). k-Shape outperforms Random in 78 out of 85 datasets and the Wilcoxon test suggests that this difference in accuracy is statistically significant. Similarly, k-Shape outperforms the AFKMC2, GibbsDPP, and LevScore sampling methods in 82, 80, and 82 datasets, respectively. For all three pairwise comparisons the Wilcoxon test suggests that the differences in accuracy are statistically significant. To ensure that the remarkable performance of k-Shape is not an artifact of choosing k landmark time series as the number of classes per datasets, we perform an additional experiment where we vary the number of k landmark time series from 10 to 100. Figure 4.6b presents the approximation error (transformed into accuracy as discussed in Section 4.5) on a randomly chosen dataset, namely, the StarLightCurve dataset. We observe that k-Shape outperforms all sampling methods across all different values of k and that Wilcoxon suggests that all differences in accuracy are statistically significant. Importantly, we observe similar behaviors across all datasets.

Comparison against projection methods: Having shown the superiority of k-Shape against sampling methods, we now evaluate the performance of k-Shape again two projection methods. k-Shape outperforms SRFT and GP in 75 and 74 datasets, respectively, and all differences in accuracy are statistically significant according to Wilcoxon. As before, Figure 4.6b compares k-Shape against SRFT and GP for different values of k. k-Shape significantly outperforms both methods across all different values of k and datasets.

Statistical analysis: To verify the superiority of k-Shape against the other methods, we evaluate the significance of their differences in accuracy when considered all together. Figure 4.7 shows the average rank across datasets for each method. k-Shape is the top method, meaning that k-Shape performed best in the majority of the datasets. The Friedman test

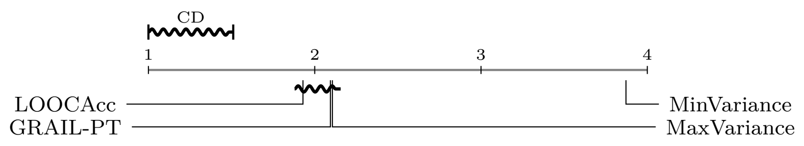


Figure 4.8: Ranking of parameter tuning methods based on the average of their ranks, using accuracy as measure, across datasets.

rejects the null hypothesis that all methods behave similarly and, therefore, we proceed with a post hoc Nemenyi test to evaluate the significance of the differences in the ranks. We observe three clusters of methods whose ranks do not present a statistically significant difference: SRFT and GP, the two projection methods, form the first cluster; AFKMC2 and Random, two sampling methods, form the second cluster; and GibbsDPP and LevScore, two sampling methods, form the third cluster. The methods in the first cluster are significantly better than the methods in the second cluster and, in turn, the methods in the second cluster are significantly better than the methods in the third cluster. Therefore, we can conclude that projection methods outperform sampling methods for the dictionary learning task and, importantly, k-Shape is the only method that significantly outperforms all projection and sampling methods.

4.6.3 Evaluation of Parameter Tuning Method for WINK

Having shown the robustness of WINK and k-Shape, the first two core components of our GRAIL framework, we now turn our focus to the evaluation of GRAIL-PT, our parameter tuning method. Specifically, we evaluate GRAIL-PT against three parameter tuning methods using the learned representations to assess their impact in terms of classification accuracy using 1-NN classifiers and in terms of their dimensionality.

Comparison in terms of classification accuracy: Figure 4.8 shows the average rank across datasets for each method using their classification accuracy as measure. LOOCAcc, a supervised method to select parameters using the training set of each dataset, is ranked first, as expected, meaning that LOOCAcc performed best in the majority of the datasets. Interestingly, we observe that GRAIL-PT, our unsupervised method to tune parameters,

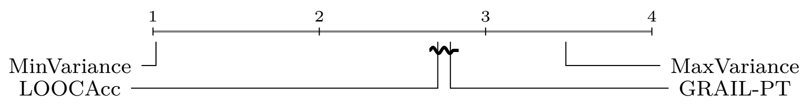


Figure 4.9: Ranking of parameter tuning methods based on the average of their ranks, using length as measure, across datasets.

and MaxVariance achieve similar classification accuracy to LOOCAcc. According to the Friedman test followed by a post-hoc Nemenyi test to evaluate the significance of the differences in the ranks, only MinVariance achieves a significant reduction in terms of accuracy in comparison to the other methods.

Comparison in terms of dimensionality: To better understand the performance of parameter tuning methods, we evaluate the learned representations in terms of their dimensionality. Figure 4.9 presents the average rank across datasets for each method using as measure the dimensionality (i.e., the length) of the learned representations. We observe that MinVariance is ranked first meaning that MinVariance produces the most compact representations in comparison to all other methods. However, as we have seen previously, MinVariance also leads to a significant loss in terms of classification accuracy. In contrast, we observe that GRAIL-PT produces representations as compact as those from LOOCAcc, the supervised method to tune parameters. Importantly, GRAIL-PT significantly outperforms MaxVariance in terms of the dimensionality of the learned representations, despite the similar performance of GRAIL-PT and MaxVariance in terms of classification accuracy. Therefore, we can conclude that GRAIL-PT is the only unsupervised method that produces accurate and compact representations, similar to those produced in a supervised manner by LOOCAcc. In contrast, MaxVariance and MinVariance produce either very high-dimensional representations or very low-dimensional representations, respectively. Very low-dimensional representations are desirable. Unfortunately, MinVariance is not competitive in terms of accuracy when selecting such low-dimensional representations.

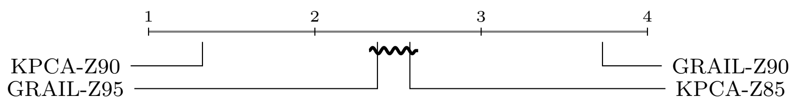


Figure 4.10: Ranking of representation methods based on the average of their ranks, using the approximation error as measure, across datasets.

4.6.4 Evaluation of GRAIL Against Other Representations

Previously, we focused our evaluation on the three core components of GRAIL: (i) the comparison method; (ii) the dictionary learning method; and (iii) the parameter tuning method. We now evaluate the performance of the GRAIL representations against representations learned used the exact KPCA algorithm.

Comparison against KPCA representations: Figure 4.10 presents the average rank across datasets for two KPCA representations explaining 85% (KPCA-Z85) and 90% (KPCA-Z90) of the variance in datasets and for two GRAIL representations explaining 90% (GRAIL-Z90) and 95% (GRAIL-Z95) of the variance in datasets using the approximation error as measure. We observe that KPCA-Z90 is ranked first, meaning that KPCA-Z90 had the best approximation error in the majority of the datasets. GRAIL-Z95 is ranked second, followed by KPCA-Z85, but the difference in their approximation error is not statistically significant according to the Friedman test followed by a post hoc Nemenyi test. Therefore, we observe that GRAIL representations must explain larger percentages of the variance in data to achieve performance similar to that of exact representations, an expected result considering that GRAIL is an approximate method for the exact KPCA.

4.6.5 Evaluation of GRAIL for Querying and Indexing

Having shown the robustness of all critical components of our GRAIL framework, we now focus our evaluation on the performance of the learned representation for different time-series mining tasks. We first evaluate GRAIL representations for time-series querying and indexing.

Comparison against other lower bounding methods: We evaluate GRAIL-LB, our lower bounding method for WINK, against state-of-the-art lower bounding methods for ED

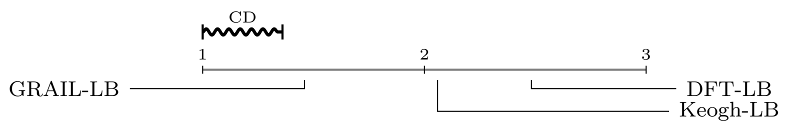


Figure 4.11: Ranking of lower bounding methods based on the average of their ranks, using the pruning power as measure, across datasets.

(DFT-LB) and cDTW[5] (Keogh-LB). Figure 4.11 shows the average rank across datasets using as measure the pruning power (i.e., the number of comparisons the methods avoid over all pairwise comparisons). We observe that GRAIL-LB is ranked first, meaning that GRAIL-LB performs better in the majority of the datasets. Keogh-LB is ranked second, followed by DFT-LB. The Friedman test rejects the null hypothesis that all methods behave similarly and, therefore, we proceed with a post hoc Nemenyi test to evaluate the significance of the differences in the ranks. We observe no clusters, meaning that GRAIL-PT significantly outperforms Keogh-LB and, in turn, Keogh-LB significantly outperforms DFT-LB. Therefore, GRAIL-LB shows a significant improvement in comparison to DFT-LB, the state-of-the-art method that operates over low-dimensional representations. Importantly, GRAIL-LB also outperforms Keogh-LB, a method that operates over the original raw representation of time series.

4.6.6 Evaluation of GRAIL for Classification

Having evaluated GRAIL-LB for time-series querying and indexing, we now turn our focus to time-series classification.

Comparison against other classification methods: We evaluate our learned representations for the task of time-series classification against the state-of-the-art classifiers we considered to evaluate WINK (see Section 4.6.1). Figure 4.12 shows the average rank across datasets based on classification accuracy. SVM+WINK is ranked first, meaning that SVM+WINK performed better in the majority of the datasets. GRAIL-SVM is ranked second and, according to the Nemenyi test, the difference in accuracy between GRAIL-SVM and SVM+WINK is not statistically significant. The same holds for GRAIL-SVM

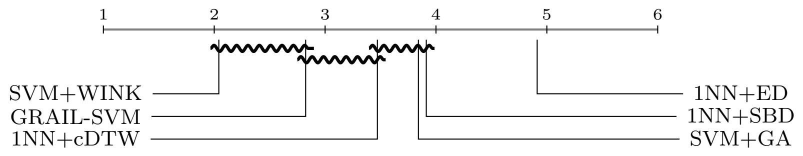
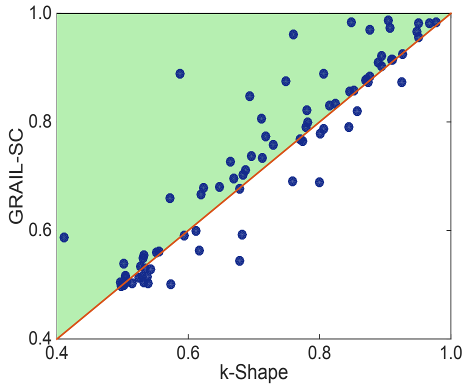
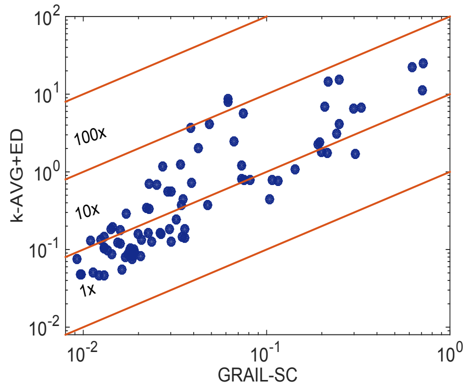


Figure 4.12: Ranking of classification methods based on the average of their ranks, using accuracy as measure, across datasets.



(a) Rand Index accuracy



(b) Runtime

Figure 4.13: Comparison of clustering algorithms.

and 1NN+cDTW^{Opt}, which is ranked third. This result indicates that the loss in classification accuracy of the approximate GRAIL-SVM method is negligible in comparison to the exact SVM+WINK method. Importantly, despite this loss in accuracy, GRAIL-SVM still performs at least as well as 1NN+cDTW^{Opt}.

4.6.7 Evaluation of GRAIL for Clustering

Previously, we evaluated our learned representations for time-series querying and classification. We now focus on time-series clustering, another important time-series mining task.

Comparison against k-Shape and k-AVG+ED: We evaluate GRAIL-SC, our spectral clustering algorithm operating over our GRAIL representations, against k-Shape and k-

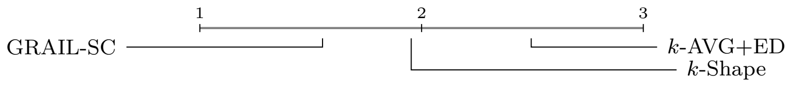


Figure 4.14: Ranking of clustering methods based on the average of their ranks, using Rand Index as measure, across datasets.

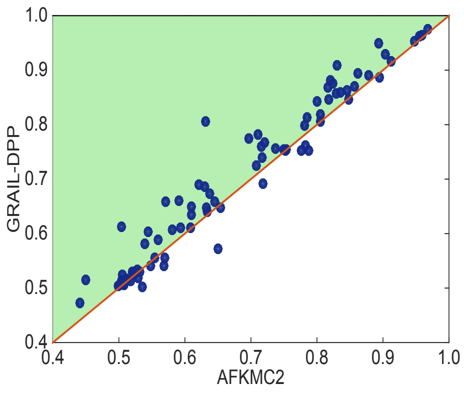
AVG+ED. Figure 4.13a presents the pairwise differences in Rand Index accuracy between GRAIL-SC and *k*-Shape, where each circle represents a dataset. We observe that GRAIL-SC outperforms *k*-Shape in 57 datasets and the Wilcoxon test suggests that this difference is statistically significant. Similarly, GRAIL-SC significantly outperforms *k*-AVG+ED in 66 datasets. Interestingly, GRAIL-SC is the only algorithm operating over time-series of reduced dimensionality whereas *k*-Shape and *k*-AVG+ED operate over the original raw time-series representations.

Statistical analysis: To verify the superiority of GRAIL-SC against the other methods, we evaluate the significance of their differences in accuracy when considered all together. Figure 4.14 shows the average rank across datasets for each method. GRAIL-SC is the top method, meaning that GRAIL-SC performed best in the majority of the datasets. The Friedman test rejects the null hypothesis that all methods behave similarly and, therefore, we proceed with a post hoc Nemenyi test to evaluate the significance of the differences in the ranks. We observe no clusters, meaning that GRAIL-SC, which is ranked first, significantly outperforms *k*-Shape and, in turn, *k*-Shape significantly outperforms *k*-AVG+ED.

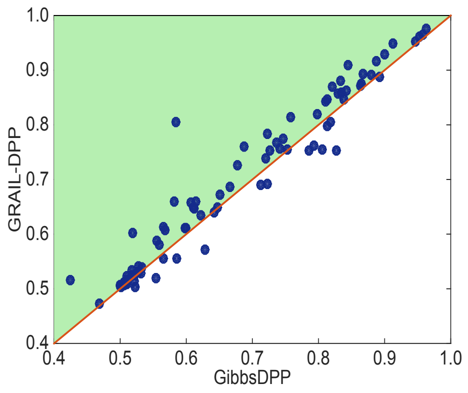
Efficiency: GRAIL-SC is the only clustering method operating over compact representations that significantly outperforms all standalone clustering methods operating over the raw time-series representations. We now investigate whether this superiority in accuracy has an associated penalty in efficiency. Figure 4.13b shows how much slower *k*-AVG+ED is than GRAIL-SC, for each dataset. We observe that GRAIL-SC is significantly faster than *k*-AVG+ED across all datasets, with the differences ranging between one order of magnitude to two orders of magnitude. Therefore, GRAIL-SC does not only lead to more accurate clustering of time series but, importantly, GRAIL-SC leads to significantly faster methods for time-series clustering, as long as GRAIL representations have been precomputed.

4.6.8 Evaluation of GRAIL for Sampling

Having shown the performance of GRAIL representations for querying, classification, and clustering of time series, we now focus our evaluation on sampling, an important subroutine



(a) Approximation error



(b) Approximation error

Figure 4.15: Comparison for sampling methods.

for many time-series mining methods.

Comparison against AFKMC2 and GibbsDPP: We evaluate GRAIL-DPP, our DPP-based sampling method that operates over GRAIL representations, against two state-of-the-art methods for approximate sampling, for the k-means++ mechanism (AFKMC2) and the DPP method (GibbsDPP). Figure 4.15a presents the pairwise differences in approximation error (transformed into accuracy as discussed in Section 4.5) between GRAIL-DPP and AFKMC2. GRAIL-DPP outperforms AFKMC2 in 54 out of 85 datasets and the Wilcoxon test suggests that this difference in accuracy is statistically significant. Similarly, Figure 4.15b shows the pairwise differences in approximation error between GRAIL-DPP and GibbsDPP. GRAIL-DPP significantly outperforms GibbsDPP in 56 out of 85 datasets.

Statistical analysis: To verify the superiority of GRAIL-DPP against the other sampling methods, we evaluate the significance of their differences in accuracy when considered all together. Figure 4.16 shows the average rank across datasets for each method. GRAIL-DPP is ranked first, meaning that GRAIL-DPP performed better in the majority of the datasets. Interestingly, AFKMC2 and GibbsDPP show no statistically significant difference in accuracy according to the Nemenyi test.

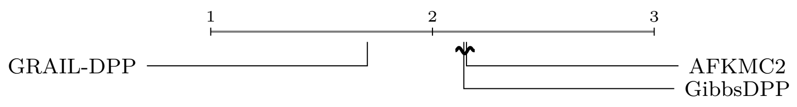


Figure 4.16: Ranking of sampling methods based on the average of their ranks, using the approximation error as measure, across datasets.

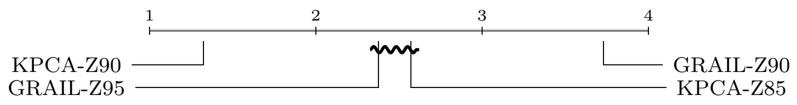


Figure 4.17: Ranking of representation methods based on the average of their ranks, using the approximation error as measure, across datasets.

4.6.9 Evaluation of GRAIL for Visualization

Finally, we evaluate the performance of GRAIL representations against representations produced using the exact KPCA algorithm for the purpose of visualization.

Comparison against KPCA representations: Figure 4.17 presents the average rank across datasets for two visualizations produced by KPCA explaining 85% (KPCA-Z85) and 90% (KPCA-Z90) of the variance in datasets and for two GRAIL representations explaining 90% (GRAIL-Z90) and 95% (GRAIL-Z95) of the variance in datasets using the approximation error as measure. We note that we use the exact same settings as we did to evaluate the regular representations in Section 4.6.4. The difference in this experiment is that we proceed with an additional normalization step across all representations and, therefore, we observe similar behaviors as before. Specifically, KPCA-Z90 is ranked first, meaning that KPCA-Z90 had the best approximation error in the majority of the datasets. GRAIL-Z95 is ranked second, followed by KPCA-Z85, but the difference in their approximation error is not statistically significant according to the Friedman test followed by a post hoc Nemenyi test.

4.6.10 Summary of Experimental Evaluation

In short, our experimental evaluation suggests that: (1) kernel methods combined with suitable kernel functions, such as WINK, significantly outperform state-of-the-art distance measures combined with 1NN classifiers, which debunks a long-standing perception in the time-series literature that 1NN classifiers are difficult to beat [Xi et al., 2006; Wang et al., 2013; Bagnall et al., 2017]; (2) cluster centroids, such as those computed using k-Shape, can effectively serve as dictionaries of landmark time series for representation learning tasks; (3) unsupervised tuning of kernel function parameters leads to accurate and compact time-series representations; (4) GRAIL learns time-series representations with reasonable accuracy loss in comparison to learning representations using exact KPCA; (5) GRAIL representations achieve excellent pruning of time-series comparisons; (6) GRAIL representations, combined with linear classifiers, significantly outperform state-of-the-art classifiers operating over raw time series; (7) GRAIL representations, combined with approximate spectral clustering, significantly outperform state-of-the-art clustering methods operating over raw time series; (8) GRAIL representations, combined with approximate DPP sampling method, significantly outperform state-of-the-art sampling methods operating over raw time series; and (9) GRAIL representations, combined with approximate KPCA, achieve performance comparable to exact KPCA.

4.7 Conclusions

In this chapter, we addressed the problem of efficiently learning data-aware, low-dimensional representations of time series that preserve the invariances offered by a given kernel function (Section 4.2.6). We summarize the contributions of this chapter as follows: (i) we developed WINK, a kernel function to compare time series under shift- and warp-invariances (Section 4.3.1); (ii) we constructed landmark time series for Nyström using our effective time-series clustering method developed in Chapter 3 (Section 4.3.2); (iii) we presented a method to estimate kernel function parameters and compactness of representation (Section 4.3.3); (iv) we learned GRAIL representations by approximating the eigendecomposition of the kernel matrix (Section 4.3.4); (v) we showed how GRAIL representations accelerate kernel methods for major time-series mining tasks (Section 4.4); and (vi) we evaluate our ideas by conducting an extensive experimental evaluation (Sections 4.5 and 4.6).

Our findings show that kernel classifiers using WINK can significantly outperform 1NN classifiers with state-of-the-art distance measures, which debunks a long-standing perception in the time-series literature that 1NN classifiers are difficult to beat [Xi et al., 2006; Wang et al., 2013; Bagnall et al., 2017]. For indexing, GRAIL representations are more compact and have better pruning power than current time-series representations. k-Shape is a highly accurate method to extract dictionaries for time series, which significantly outperforms all state-of-the-art methods. Our unsupervised parameter tuning method selects parameters that lead to representations that have similar accuracy and length than those representations we computed using a supervised approach. GRAIL representations require the explanation of a larger percentage of the variance in data to match the performance of exact representations, because one is an approximate method and the other is an exact method. In all five tasks that were part of our analysis, GRAIL representations show remarkable accuracy and efficiency.

Chapter 5

An End-to-End System for Predicting the Impact of Scientific Concepts

In previous chapters, we presented a set of general purpose methods to solve important time-series mining tasks, such as clustering and classification. Our emphasis was to develop scalable and accurate methods to solve these tasks while operating solely over time-series collections. However, in many complex, real-world applications, we need to measure different facets of an underlying process, which requires operations over different types of data (e.g., graphs, text, and time series). These operations often capture user-defined characteristics in the form of feature vectors (i.e., sets of features). Consequently, we need to combine features produced by different operators over different types of data to (i) construct more advanced features to reason about the temporal relationships, trends, and patterns of simpler features; and (ii) transform these features into actionable knowledge (e.g., predicting an outcome). Importantly, because these operations often produce thousands of features, combining features exhaustively may result in an explosion of new features that we cannot afford to store or preprocess. Therefore, lightweight solutions are required to combine features in real time.

In this chapter, as well as in Chapter 6, we study large-scale, real-world applications that provide answers in real time and combine knowledge from different types of data. Considering these requirements, our objective is to (i) develop generic, simple, and lightweight methodologies to effectively extract features from time-varying measurements that seamlessly integrate with techniques developed for other types of data; and (ii) demonstrate the impact of exploiting methods from time-series analysis to significantly improve the accuracy of prediction models used in such large-scale, real-world applications.

To achieve this goal, in this chapter we study the problem of predicting the future impact of scientific concepts and describe an end-to-end system developed for this task by a team of researchers at Columbia and several other universities. First, we review the challenges of this ambitious, yet important, problem (Section 5.1). Second, we describe the architecture of the system to facilitate (i) extraction of characteristics from articles in the scientific literature; and (ii) real-time prediction of the future impact of scientific concepts (Section 5.2). Then, we outline the large number of characteristics that our system extracts from the metadata and the full text of scientific articles and we present a variety of principled statistical measures capable of capturing patterns in time-varying characteristics of scientific articles (Section 5.3). Finally, we summarize the results of our large-scale experimental evaluation (Sections 5.4 and 5.5) and conclude with the contributions of this chapter (Section 5.6).

5.1 Overview and Motivation

Even though trillions of US dollars are allocated every year in research and development [Grueber and Studt, 2012], only a small percentage of this amount is devoted to technologies that will eventually have high impact for society. Therefore, it is critical to identify research concepts that hold the most promise as early as possible. To achieve this goal, a framework is required to predict whether a new scientific achievement will be accepted in future years.

To address this problem, we were part of a large team that built a system to predict the impact of scientific concepts. Specifically, given a technical term (e.g., *microRNA* or *rewiring*), our system collects information from available articles in a reference period and predicts if the given concept will gain prominence in the scientific literature in the future. For example, our system, by examining scientific articles published between 1997 and 2003 related to the term *microRNA*, predicts that *microRNA* gains prominence in scientific articles published in the later years that we study (2004–2007). Thus, our approach predicts that *microRNA* will have scientific impact. In contrast, by examining scientific articles related to *rewiring* in the same time period, our system predicts that this term will not be prominent in scientific articles published in 2004–2007.

Unlike much previous work on citation prediction (see Chapter 7.3), our system collects information from both the metadata and the full text of the available scientific articles. From full text, our system identifies concepts, relations, citation sentiment, and the rhetorical function of sentences. From metadata, our system computes the citation and author collaboration networks. Then, the system computes several measures over the extracted information from the articles and provides these measures as input to a time-series analysis component, a critical component of the system that analyzes the evolution of the features over time using a variety of principled time-series analysis methods. Finally, the system combines all features using logistic regression and computes an overall prominence score for the input technical term, to predict its impact in the literature. We define impact as a function of the relative growth of term appearances over unique documents (see further discussion on this definition in Section 5.3.3).

We conduct a large-scale evaluation over two datasets to (i) show the relative contribution of features drawn from the full text of the articles in comparison to features drawn from their metadata; and (ii) demonstrate the importance of using features from time-series analysis for this task. Specifically, one dataset consists of 3.8 million articles with full text from Elsevier and metadata from Thompson Reuter's Web of Science (WOS) service. The other dataset consists of 48 million articles with metadata only from the WOS service.

The WOS dataset includes abstracts for each scientific article plus metadata such as title, authors, publication venue, year of publication, and citations.

Our experiments address two questions: (i) whether a very large amount of metadata enables better prediction even without the text features, making them redundant; and (ii) whether tracking features over time improves the accuracy of metadata-based and text-based features. Our experiments suggest that it is well worth the effort to obtain the full text of scientific articles and to exploit the power of natural language analysis. Importantly, patterns extracted from features tracked over time, especially those associated with the full text of articles, were critically important for this prediction task.

We first present the architecture of the system we built to (i) extract characteristics from articles in the scientific literature and (ii) perform real-time predictions of the future impact of scientific concepts (Section 5.2). We continue as follows:

- We describe the large number of characteristics that our system extracts from the metadata and the full text of scientific articles (Section 5.3.1).
- We present a principled methodology to capture patterns in short, sparse, and noisy time series that we implement in the time-series analysis component, a critical component of our system (Section 5.3.2).
- We overview the simple, yet effective, prediction model of our system (Section 5.3.3).
- We summarize our large-scale experimental evaluation (Sections 5.4 and 5.5).

Finally, we conclude and discuss the implications of our work (Section 5.6). The material described in this chapter appears in [McKeown et al., 2016].

We now provide an overview of the prediction system.

5.2 System Architecture

Our system predicts the impact of a scientific concept, represented as a technical term, using features derived from the full text of scientific articles as well as more traditional

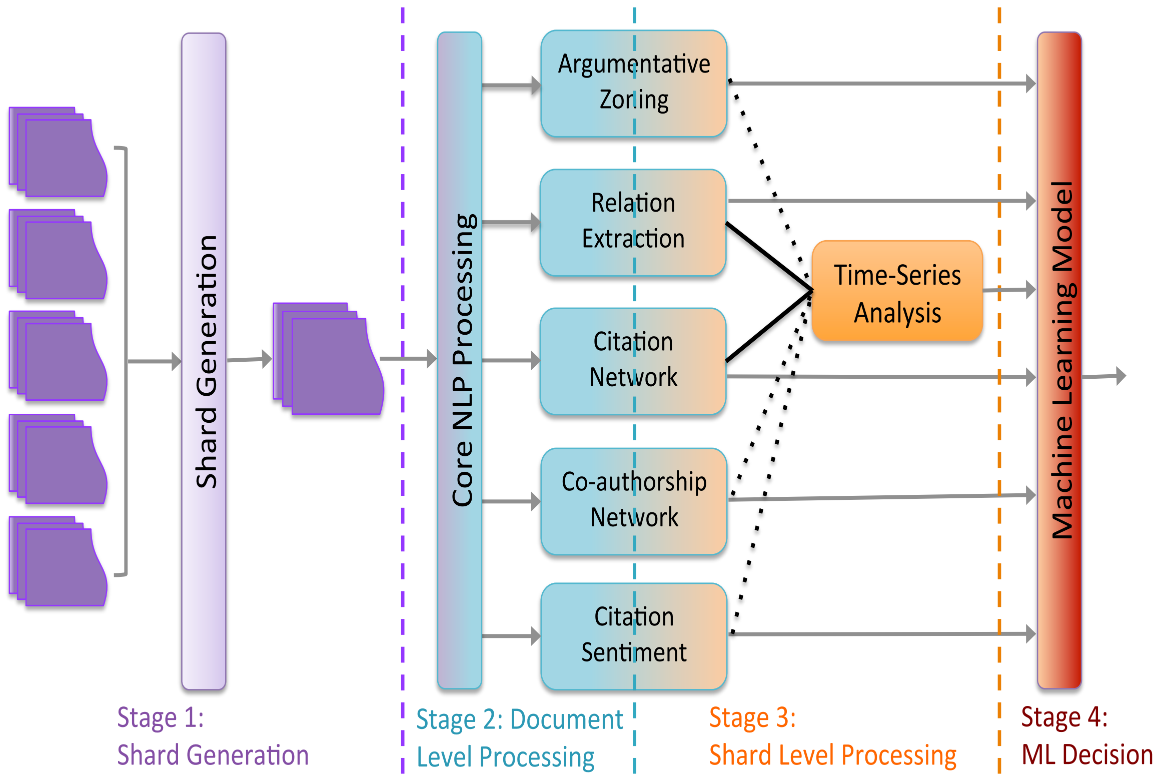


Figure 5.1: System architecture.

features derived from the metadata of the documents. The technical terms used as input refer to specific scientific concepts and are assumed to have no synonyms.

The system is designed as a four-staged pipeline. Given an input term, our system first computes the set of documents relevant to the term by determining when there is an exact match between the term and the words of either the title or the abstract. As shown in Figure 5.1, this first stage, called *shard generation*, produces a set of relevant documents that we call the *shard*. In Stage 2, we process each document in the shard using natural language processing, to produce annotations representing sentence segmentation, part-of-speech tagging, and parsing of citation sentences. We then annotate each document with the rhetorical function of each sentence using *argumentative zones* [Teufel, 2010], entities and relations expressed in the text, and sentiment toward citations. In Stage 3, we compute aggregate values for these annotations across the shards and build a coauthorship network and a citation network for the documents in the shard. Importantly, we also generate time series for each feature over the years in the reference period, and produce additional features from various functions applied to the time series, which we discuss next. Finally, in Stage 4, our machine learning module uses the features to predict the scientific impact in the forecast period.

The system was developed together by a large team of researchers at Columbia and several other universities. Specifically, we had a prominent role in the design and development of the system in general, as well as of Stage 3 in the pipeline, where we developed the time-series analysis component (see Section 5.3.2).

5.3 Early Prediction of Scientific Impact

Our objective is to develop a methodology to extract characteristics from scientific articles and to analyze the evolution of these static characteristics over time. To achieve that, we first describe the large set of characteristics that our system extracts from the metadata and the full text of scientific articles (Section 5.3.1). Then, we present a variety of principled statistical measures capable of capturing patterns in time-varying characteristics (Section 5.3.2). Finally, we describe our simple prediction mechanism (Section 5.3.3).

5.3.1 Feature Extraction from Metadata and Full Text

Next, we present the set of features we consider from the metadata of scientific articles. Then, in Section 5.3.1.2, we describe the set of features we consider from the full text of scientific articles.

5.3.1.1 Feature Extraction from Metadata

This section and the research described therein were primarily developed and written by Dragomir Radev's group at the University of Michigan. We include this section for context for our time-series work in the project.

Our system uses the metadata available for each article to compute some simple features

Feature Type	**Feature Description**
Basic	number of nodes
	number of edges
	number of weakly-connected components
	size of largest weakly-connected component
Clustering	average Watts-Strogatz coefficient [Watts and Strogatz, 1998]
	average Newman coefficient [Newman, 2010]
Centrality	average degree
	average closeness centrality [Freeman, 1978]
	average betweenness centrality [Freeman, 1977]
Distances	diameter
	average shortest path
Degree distribution	degree assortativity [Newman, 2003]
	in-/out-/total-degree power law exponent [Newman, 2010]
	in-/out-/total-degree Newman power law exponent [Newman, 2010]
	sin-/out-/total-degree power law R^2 [Newman, 2010]

Table 5.1: A list of features computed for each network.

and other more complex features based on networks. For the simple features, which we call *acceptance* features, we consider the number of unique papers, authors and their countries, institutions, conferences, journals, and books. Additionally, we compute the mean number of authors per paper, the number of papers with two or more authors, the number of papers with authors affiliated with multiple institutions, and the number of papers with authors from different institutions. For the network-based features, network theory [Newman, 2010] provides a number of tools to model aggregate information in relational data. Several recent papers have focused on applying network techniques to analyze bibliometric data [Batagelj and Cerinšek, 2013; Viana et al., 2013; Fu et al., 2014; Pan et al., 2012]. The use of networks to model bibliometric data such as collaborations between authors and citations between papers is based on the view of science as a social process [Sun et al., 2013]. We derive network features (specified in Table 5.1) from two kinds of networks, namely, author collaboration networks and citation networks. In an author collaboration network, nodes represent authors and undirected edges represent the fact that two authors co-authored at least one paper.[1] In a citation network, nodes represent documents and directed edges record that one document cites the other (only within a given shard). Citation links between papers indicate topical similarity [Small, 1973; Kessler, 1965]. Many dense clusters in

[1]We used the author resolution results produced by [Wick et al., 2013].

a citation network may represent fragmented communities of research where documents position themselves relative to papers in the same cluster and do not frequently cite other papers in the area. Similarly, a low clustering coefficient may indicate that a field tends to make large, disruptive advances [Funk and Owen-Smith, 2012], rather than incremental improvements.

In contrast, collaborations provide a more direct probe into the social dynamics of research on a given topic. For example, dense clusters in this network represent close-knit communities that exist among the authors in a field. Similarly, an author with high betweenness centrality, an indicator of the node's centrality in the network, may act as a bridge between two different communities that do not frequently collaborate.

Given a shard, these networks can be built efficiently using our metadata database. The citation network is built by querying a database table that contains resolved citations between papers;[2] the author collaboration network is built by querying a table containing authors of each paper.

5.3.1.2 Feature Extraction from Full Text

Our full text features are computed based on aggregates of information extracted from the text of each article: entities and relations, argumentative zoning, and citation sentiment. The entity extraction research described in this section was primarily developed—and the associated description included here written by—Michael Collins's group at Columbia University. The relation extraction research described in this section was primarily developed—and the associated description included here written by—Luis Gravano's group at Columbia University. The argumentative zoning and citation sentiment research described in this section was primarily developed—and the associated description included here written by—Simone Teufel's group at Cambridge University. We include these important building blocks of our system here for context for our time-series work in the project.

[2]Resolving a citation is the process of using the bibliographic text to locate the cited paper in the database.

Entities and Relations: We identify two types of textual information, namely, *entities* and *relations*. The information that we extract enables a more refined analysis of crucial aspects around a given topic than would be possible using the original unannotated text. For example, we can extract the number of algorithms that have been implemented for a given input problem, and use it as evidence of the depth in which this problem has been studied. Similarly, we can gauge the interest in a research topic based on the diversity of funding agencies involved in the topic. Entities (e.g., focus, techniques, and domains [Gupta and Manning, 2010]) and relations (e.g., protein–protein interaction [Bui et al., 2011]) involving them have been extracted from scientific articles, although to the best of our knowledge, they have not been used in scientific prominence prediction systems.

The entity detection module produces annotations consisting of an entity *type* (e.g., algorithm, dataset, gene, virus, protein, database) and a mention (e.g., CRF, an instance of algorithm; BRCA1, an instance of gene). We recognize a total of 15 entity types. Some of the entity types are general to all domains (e.g., method, problem, theory) and others are specific to the most frequently occurring family of domains in the corpus (e.g., medical, genomic, biology). We define the *primary type* as the entity type corresponding to the scientific term given as input to our system if it matches one of our 15 entity types. Otherwise, it is the entity type with the highest document frequency in the shard. We can now measure how cohesive a shard is by using the proportion of articles containing a mention of the primary entity type in the shard. We can also measure how diverse it is by counting the number of distinct mentions of the primary entity type in the shard. If the term is an entity, we also compute as features the frequency and corresponding rank of the term with respect to other entities of the same type—both absolute and normalized—and the ratio between the frequency of the input term and the most frequent entity of its same type.

To annotate the entities, we use a dictionary-based tagger [Neelakantan and Collins, 2014]. Dictionaries are compiled for every named entity type using large amounts of unlabeled data and a small number of labeled examples. For every named entity type, first we construct a high recall, low precision list of candidate phrases by applying simple rules on the unlabeled data collection. Using Canonical Correlation Analysis (CCA) [Hotelling, 1936], we represent each candidate phrase in a low-dimensional, real-valued space. Finally, we learn a binary SVM [Joachims, 1998] in the low-dimensional space with few labeled examples to classify the candidate phrases. We filter out the noisy phrases from the high recall, low precision list of candidate phrases using the learned SVM to get a high recall, high precision dictionary.

Table 5.2 lists the relations that we extract. For the Funding relation, we produce frequency- and average-based features indicating the number of funding agencies and the number of grants in each article. In addition, we produce Boolean features indicating whether there are multiple grants or institutions supporting the research reported by articles in the shard. For the other relations, we extract all the mentions of each type in an article and then produce numeric features indicating their frequency and average in the shard. To annotate the relations, we use two different methods. For funding information, we can, in some cases, retrieve it directly from the article metadata. However, in most cases, especially for older articles, we can only obtain this information from text, as follows. We first use string matching to locate the acknowledgment section of the article in question, where the funding information for the article usually resides. Then, we use two supervised Conditional Random Fields (CRF) models [Lafferty et al., 2001] to identify the funding agencies and grant numbers. Finally, we build ⟨funding agency, grant⟩ pairs by combining the agencies and grants that co-exist in a sentence in order of appearance. To annotate the remaining relations, we use a supervised sentence classification approach [Bach and Badaskar, 2007]. Since only a few of the sentences in an article will likely include a mention of these relations, for efficiency we only classify the sentences that mention at least one of the 10 most relevant terms according to their weight in the SVM classification model. In our experiments, we used an SVM-based classifier trained on stemmed terms along with their respective POS tags as features, from a manually annotated dataset. The accuracy of our classifiers ranges from a value of 0.72 for the F-measure for the Novelty Claims relation to a value of 0.89 for the Funding relation.

Relation Name	**Entities**	**Meaning**
Funding	⟨grant, funding agency⟩	a *grant* by *funding agency* supports an article
Novelty Claims	⟨method, problem⟩	an article applies a *method* to a *problem*
Dataset Purpose	⟨dataset, purpose⟩	an article *proposes* or *uses* a *dataset*

Table 5.2: Relations extracted by the system.

Argumentative Zoning: The argumentative zoning (AZ) component marks up each sentence in a scientific document according to its rhetorical function. We expect that an entity's prominence in the scientific community is reflected in the way scientists write about it, such as whether the entity is presented as a novel contribution (AZ category **Own Work**) or a well-established concept in the literature (AZ category **Background**). The relevance of such rhetorical categories comes from the hypothesis that the first occurrence of new ideas should be in some paper's goal statement [Myers, 1992]. However, as the idea emerges and gets accepted, it is mentioned in other areas of papers referring to the original idea—thereby "travelling" through other rhetorical categories. When the new idea is competing against other existing ideas, it will occur in contrast and comparison statements [MacRoberts and MacRoberts, 1984]. If it is adopted by other researchers in the field, it will be mentioned as the basis for their work, indicating a different phase of acceptance (or a different status of the cited idea). If the concept becomes widely accepted, it will be found with increasing frequency in rhetorically neutral sentences and eventually even in background sections [Swales, 1990]. These ideas are formalized in the "argumentative zoning" theory of Teufel [Teufel, 2010], whereby the text of an article is partitioned into zones defined by their rhetorical function.

The core functionality of the AZ component in our system is automatically labeling each sentence in an article with a category specifying the rhetorical status of that sentence. We use six categories: **Aim**, **Own Work**, **Background**, **Contrast**, **Basis** and **Other**; for more details on these categories, see [Teufel, 2010]. The document-level AZ system takes a document as input and labels every sentence with one of the six categories listed above, using a Maximum Entropy Markov Model classifier suitable for sequential labeling. The features extracted for each sentence include internal information about the words, n-grams and citations it contains as well as external information about its absolute and relative position in the document, the section in which it appears, and whether a string from an extensive pattern lexicon matched. This system has been trained using a manually annotated set of documents from the computer science and chemistry domains. Using cross-validation on the chemistry subset of the data, the system's accuracy has been measured at 75%.

To produce AZ indicator values for a concept term, we aggregate over the AZ labels of all sentences that contain a mention of the term. The aggregate indicators we produce are the absolute count totals of each AZ label in the set and the relative count proportions of each AZ label in the set, for a total of 12 indicators.

Citation Sentiment: The Citation Sentiment component labels each sentence containing a citation as expressing positive, negative or objective sentiment towards the cited entity. It implements the hypothesis that emerging ideas will initially be cited in the context of strong opinions, whether these are negative or positive ones [Small, 2011]. We also hypothesize that the more an idea is accepted in a scientific community, the more it will be presented as an "objective fact." As might be expected, most citations in scientific articles are objective in terms of sentiment (86% of sentences in the annotated corpus described below); this may be an indication that positive or negative citations are somewhat rare, and may be important.

Similarly to the AZ component, the citation sentiment module first assigns sentence-level labels and later aggregates over them to produce feature values for the entity of interest. The sentence-level classifier, based on Athar [Athar, 2011], is a Support Vector Machine that takes n-gram features and basic negation features as input and outputs one of three sentiment labels: Positive, Negative or Objective. It was trained on Athar's corpus of 8,736 hand-labeled citation sentences. The entity-level feature values are then calculated as total and proportional counts of these labels over a set of sentences that are relevant to the entity of interest. Because citation sentiment is by definition only meaningful in the presence of citations, we aggregate over all sentences that contain the term and also contain a citation. The performance of the citation sentiment component has a value of 0.6 for the macro-average F-measure.

5.3.2 Feature Aggregation and Temporal Pattern Extraction

Until now, we have described a large number of characteristics that our system extracts from the metadata and the full text of scientific articles. To capture the temporal variation of such characteristics, we present a lightweight, yet effective, Time-Series Analysis (TSA) component to extract features from time-varying characteristics. In previous chapters, we presented a set of general purpose methods to perform major tasks that solely rely on time-series analysis. However, in many real-world applications we need to combine methodologies developed for different types of data in real time. Considering that such methodologies often produce thousands of features, combination of features result in an explosion of new features that we cannot afford to store or spend time to preprocess. The real-world application we consider in this chapter belongs to this category.

We develop a TSA component that operates over raw time series, without the need to preprocess or store time series. Specifically, for every feature given as input, TSA computes a time-series sequence that represents its aggregated values per year, instead of its aggregated value for the full time period. Such time-series sequences are short (i.e., corresponding to a small reference period), noisy (i.e., as is the case when terms match documents unrelated to the scientific concept), and with missing values (i.e., as is the case when the years in the reference period do not contain any documents). To circumvent such conditions and capture how these characteristics grow and fade over time, we model time series using six growth functions: Linear, Quadratic, Logistic, Exponential, Gompertz, and Richards. For the Linear and Quadratic functions we use linear least-squares estimates; for the other functions we use non-linear least-squares estimates of their parameters. Once all functions have been fitted to a time series, we select the function with the smallest Akaike Information Criterion (AIC) [Akaike, 1974] value as the best. We use the name of the best-fitted function, as well as its slope, as features for our ML component. We also use as features the coefficients of the first and second degree terms of the Linear and Quadratic functions, respectively, with which we can determine the trend and its rate of change.

In addition to these model-based features, we also consider a variety of statistical measures from the literature to capture global characteristics of time series and detect interesting patterns. In particular, we use 9 such characteristics and compute them as proposed by [Wang et al., 2006]. Briefly, *Seasonality*, *Periodicity*, and *Trend* are features that attempt to detect cycles, the period of those cycles, and the strength of the long-term trend of a time series. *Skewness* measures the degree of asymmetry of data points of a time series around their mean and *Kurtosis* measures the peakness and flatness of data points, relative to a normal distribution. *Serial correlation* measures how noisy a time series is by fitting a white noise model, and is defined as the Box-Pierce Statistic [Box and Cox, 1964]. *Non-linearity* measures the non-linearity structure of time series data, from which we can determine if linear or non-linear models can better forecast the data [Teräsvirta et al., 1993]. *Self-similarity*, which relates to the autocorrelation statistic, measures the long-range dependence of a time series; we compute this feature as the Hurst Exponent [Willinger et al., 1998]. Finally, we use the Lyapunov Exponent, which measures the chaotic behavior of a time series, to detect the degree of randomness [Hilborn, 2000].

5.3.3 Prediction Model and Methodology

This section and the research described therein were primarily developed and written by Hal Daumé's group at the University of Maryland. We include this section for context for our time-series work in the project.

Our system was developed as part of a government funded program to predict the scientific impact of entities such as terms in some future forecast period F given some observations in the reference period R where $R < F$. Scientific impact is quantified by the program in the form of ground truth functions (GTFs) that concentrate on the relative growth of term appearance in unique documents over a baseline count as opposed to absolute growth. Previous work has often looked at absolute growth of counts such as citations [Yogatama et al., 2011]. The underlying motivation of GTFs is to temper variance in count quantity across disciplines (e.g., biology tends to have more publications than pure math) and time (i.e., absolute publication counts increase from past to present). The ground truth function (GTF) considers documents from three trusted sources, namely, *Science*, *Nature*, and the *Proceedings of the National Academy of Sciences (PNAS)*, denoted as *TS-documents*. Formally, the GTF for a term e is defined in terms of document counts for e over R or F:

1. $r(e)$: exponentially weighted average of counts of unique TS-documents containing e for the years leading up to and including R, where the interval used for averaging is the size of the forecast gap and where counts in recent years are weighted more heavily.

2. $f(e)$: exponentially weighted average of counts of unique TS-documents containing e for the years up to and including F, where the interval used for averaging is the size of the forecast gap and where counts in recent years are weighted more heavily.

When $f(e) < \max(1, r(e))$ the GTF is defined to be zero, otherwise it produces values in the range from 0 to 1, as follows:

$$\mathrm{GTF}(e,r,f) = \left(1 - \frac{r(e)}{f(e)}\right)\left(1 - \frac{1}{f(e)}\right) \tag{5.1}$$

The goal of the system is to predict $\mathrm{GTF}(e,r,f)$, having observed e and its derived features in the dataset up to reference period R. In sum, the goal is to predict the ground truth function of e at F (i.e., the relative increase in counts of unique TS-documents in which e appears) having observed e up to R.

Since most papers receive no citations or a very small number of citations, the distribution of GTF values for our datasets tends towards an exponential distribution as the distance between R and F increases. GTFs for terms range from 0 to 1 and thus, it is possible to model the desired prediction using vanilla logistic regression [Hastie et al., 2009].[3]

[3]An alternative would be to train a model to predict the cumulative counts $r(e)$ and $f(e)$, from which the

Though logistic regression is typically used in the literature for classification, it can be directly applied to regression tasks where the output ranges from 0 to 1 by defining the objective function in terms of minimizing the KL divergence between the GTF and the hypothesis.

We now turn to the description of the settings of our large-scale experimental evaluation.

5.4 Experimental Settings

In this section we describe the datasets and the settings of our experimental evaluation.

Datasets: Our dataset includes 3.8 million full text articles published by Elsevier as well as 48 million metadata records from Web of Science (WOS).[4] The metadata includes titles, author names and institutions, in some cases funding, citations with the IDs of cited papers, and abstracts. The full text of the Elsevier articles was parsed into a common XML representation that identifies not only metadata, but in many cases also provides structural markup for the text, such as identifying tables, sections, and paragraphs, and linking in-text citations to the corresponding bibliography entries.

System variants: We conducted experiments to compare systems that use only text-based features with systems that use more traditional metadata features, as well as systems that use both on a dataset of scientific documents published within 1991–2007. We ran each system to forecast scientific impact in four different scenarios varying the forecasting period from one year past the reference period (chosen as 2003) to four years past the reference period, i.e., 2004 to 2007. Finally, each of the above settings was evaluated over a dataset that contained all 48 million documents in the Web of Science metadata records as well as the subset of Elsevier-published documents for which the full text of the document was available. In total, this yields 24 experimental configurations: 3 systems to predict scientific impact on 4 forecast years for 2 datasets.

GTF can be calculated. We adopted our current approach after preliminary experiments on development data.

[4]This dataset was provided by the government sponsor to all teams who were part of the funded program.

Term	2004	2005	2006	2007
dopamine signaling	0.250	0.062	0.208	0.249
lower ros	0.000	0.000	0.000	0.000
rewiring	0.222	0.000	0.000	0.145
wd40	0.000	0.250	0.585	0.629
microrna	0.547	0.857	0.863	0.905
cell self-renewal	0.188	0.393	0.332	0.330
plant homeodomain	0.000	0.000	0.492	0.718

Table 5.3: Example GTF values for four forecast periods.

Scientific concepts: Our experiments were conducted on 5923 terms from a list provided by the evaluators for our funding agency. A term is an n-gram from one to four words; the term population is drawn from abstracts and titles of documents published within the trusted sources (*Nature*, *Science* and *PNAS*) in the time period from 1991 to 2007. Terms were filtered using a common stop word list, low frequency terms[5] and common scientific terms. Some examples are provided in Table 5.3 along with the GTF value defined by Equation 5.1. We selected terms using the following methodogy. We tallied all documents that contain the term in its title or abstract and retained terms for which at least 10% of the computed documents came from the Elsevier collection and therefore had full text. This is the same method used to compute shards and thus, we knew that the shards used for prediction would not be empty when restricted only to the Elsevier collection.

Evaluation methodology and metrics: We used five-fold cross validation, with 90% of the data in each fold used for training and the rest used for testing. We compare our system results to the gold standard GTF values using Pearson correlation r, a measure of linear correlation between two variables, and Spearman rank correlation ρ, a non-parametric measure of the dependence between the ranking of two variables. For analysis, we categorized the features as metadata or full text features. Earlier experiments on development data showed that time-series analysis over argumentative zoning, sentiment and co-authorship was not helpful and, therefore, we did not include these features in the evaluation.

[5]Since we are evaluating impact within our corpus, if a term has low frequency, then it never emerges within the time frame of the corpus. It is possible that it emerges years later, but we will never be able to evaluate whether we can detect this.

Model	R^2	τ
Linear regression	-0.025	0.322
Regression tree	0.160	0.339
Random forest	0.200	0.345
Gradient boosted decision tree	0.235	**0.372**
Support vector regression	0.253	0.355
Logistic regression	**0.263**	**0.372**

Table 5.4: Performance per regression model on held-out development set using metadata features only.

5.5 Experimental Results

In this section, we report our experimental evaluation, which we conducted jointly primarily with Snigdha Chaturvedi (University of Michigan) and Kapil Thadani (Columbia University). Firstly, we compare our prediction model against other types of models (Section 5.5.1). Secondly, we evaluate our three system variants on predicting the impact of scientific concepts over two datasets (Section 5.5.2). Finally, we evaluate the contribution of each feature category and each individual feature (Section 5.5.3).

5.5.1 Evaluation of our Prediction Model Against Other Models

For our predictions we rely on logistic regression because of its simplicity and efficiency. In addition to logistic regression, we also considered other standard regression models for the task of modeling the GTF. Specifically, we considered: linear regression [Hastie et al., 2009], regression trees [Breiman et al., 1984], random forests [Breiman, 2001], gradient boosted decision trees [Friedman, 2001], and support vector regression [Guyon et al., 1993; Cortes and Vapnik, 1995; Smola and Schölkopf, 2004]. Using the metadata features only, we train models on a set of documents with GTFs defined for the period $R = 2003$ and $F = 2007$. Then, we evaluate the models on a held-out dataset over the same period in terms of R^2 and Kendall's τ. The results in Table 5.4 show that logistic regression outperforms all other models in R^2 and is tied with gradient boosted decision trees in τ. Thus, we use logistic regression as the model for our system.

Indicators	Dataset	r	ρ
All indicators	Elsevier	**0.364**	**0.392**
Text only	Elsevier	0.346	0.373
Metadata only	Elsevier	0.194	0.193
All indicators	Complete	**0.393**	**0.428**
Text only	Complete	0.365	0.407
Metadata only	Complete	0.316	0.340

Table 5.5: Comparison of the correlation of the system predictions with the ground truth using all indicators, text only indicators, and metadata only indicators for reference year 2003 and forecast year 2007. Column r corresponds to the Pearson correlation coefficient while column ρ corresponds to the Spearman rank correlation.

5.5.2 Evaluation of System Predictions

Figures 5.2 and 5.3 illustrate how our predictions correlated with the ground truth over the four forecast years. We also show numeric results in Table 5.5 for reference year 2003 and forecast year 2007.

Figure 5.2 shows the results for experiments carried out on full text drawn from Elsevier; the top graph shows Pearson r and the bottom one Spearman ρ. Here metadata features underperform text-based features by a substantial margin as measured by ρ and thus, the benefit of full text in comparison to metadata is clear. Adding text-based features to metadata-only features also substantially improves the results. Our results show that the combination of full text and metadata performs the best, outperforming the text indicators only by a slight margin, as indicated by r. These results illustrate that the metadata features add value.

Figure 5.3 shows the results for experiments carried out on the full dataset, including both Elsevier and WOS records. In this case, the shard, which includes all documents relevant to the term, is substantially larger. We might expect metadata features to outperform text-based features since the citation and co-authorship networks that are built can be more comprehensive: more articles corresponding to the citations will be found in the dataset. Furthermore, the metadata acceptance features will be drawn from all articles, while the text features will only be drawn from a subset. We observe a substantial improvement in metadata alone, but the results still do not surpass those of the text-based and the full set of features. Under Spearman ρ, both the system based on text-based features and the system

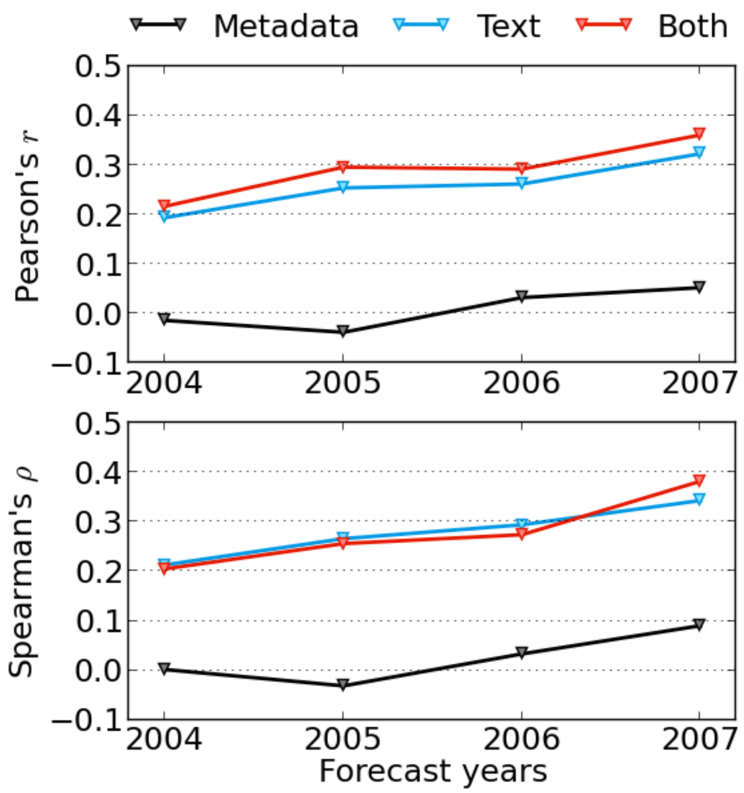


Figure 5.2: Comparison of the correlation of the system predictions with the ground truth on the Elsevier dataset using all indicators, text only indicators, and metadata only indicators for four forecast years. The top figure corresponds to the Pearson correlation coefficient r while the bottom figure corresponds to the Spearman rank correlation ρ.

based on combined text-based and metadata perform significantly better ($p < 0.05$ using the paired permutation test) than metadata only across all forecast years except 2005. Note that the system using text-based features also improves, because the larger dataset contains abstracts and text features are extracted from these. The text-only system and the system using a combination of text and metadata indicators are similar in performance, with the combination of features usually slightly outperforming their text-only counterparts.

5.5.3 Contribution of Individual Features and Feature Categories

Table 5.6 reports an ablation study of the system for 2006, the first year where combined text and metadata features outperform text-only features according to Spearman ρ. We show the performance using individual indicators in isolation, sorted by Pearson r. Text indicators

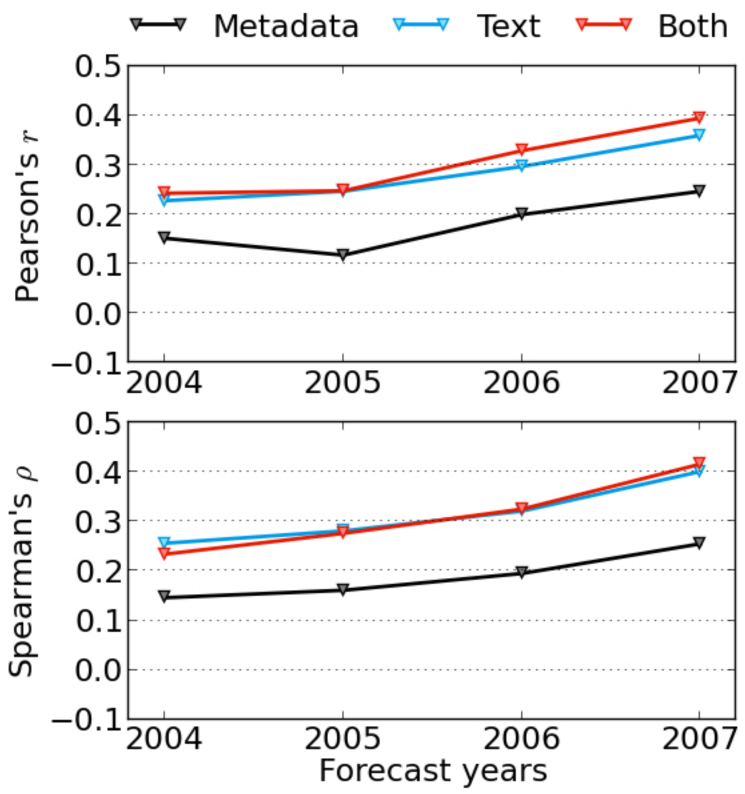


Figure 5.3: Comparison of the correlation of the system predictions with the ground truth on the dataset containing Elsevier and WOS records using all indicators, text only indicators, and metadata only indicators for four forecast years. The top figure corresponds to the Pearson correlation coefficient r while the bottom figure corresponds to the Spearman rank correlation ρ.

are shown in bold. The top performing indicators are times series over entities, acceptance, and relations, of which only acceptance is derived from metadata. Network indicators perform at the bottom of the times-series indicators and near the bottom of the regular indicators. Argumentative zoning, a text indicator that reflects the rhetorical structure of the article, performs near the top of individual indicators. We see two unexpected results: 1) acceptance, which is a metadata indicator, performs well, both in conjunction with time series indicators and without such indicators; and 2) citation sentiment performs poorly. Acceptance simply counts the number of venues, authors, and institutions in the shard of relevant documents: the more places and authors that have published on this topic, the more impact it has had. Other than these two exceptions, the individual results support our

Indicators	r	ρ
TSA:entities	0.301	0.317
TSA:acceptance	0.293	0.313
TSA:relations	0.191	0.195
Acceptance	0.19	0.214
Argumentative Zoning	0.188	0.215
Citation network	0.147	0.193
TSA:networks	0.131	0.148
Coauthorship network	0.123	0.164
Citation sentiment	0.0679	0.078

Table 5.6: Ablation tests for different feature categories for forecast year 2006. Bold text indicates full-text features. Column "r" corresponds to the Pearson correlation coefficient while column "ρ" corresponds to the Spearman rank correlation.

overall results showing that text indicators tend to perform better than metadata indicators.

We see that the time-series indicator over entities is the top performing indicator. We believe this is because we expect the shards centered around prominent entities to be more cohesive and less diverse over time. We hypothesize that cohesiveness increases with the number of mentions of the prominent entity type, while diversity decreases because there are fewer comparisons to other entities of the same type. This occurs precisely because people accept that the prominent entity is important. For example, consider a gene that is in the process of being mapped. We would have discussions of other related genes early on and, later on, when that gene becomes more important, it would appear more in the context of the disease it is important for and the drug that it reveals should be used, as opposed to discussion of other genes. As time goes on, we would also see more documents that mention the gene.

On the other hand, we see that the time-series indicators over argumentative zoning and citation sentiment do not improve the performance in comparison to the regular argumentative zoning and citation sentiment indicators, respectively. We believe this is because the distributions of the frequencies of the predicted labels remain the same when argumentative zoning and citation sentiment indicators are computed over the full reference period and when they are computed for each year separately. Therefore, for efficiency, we omit the computation of time-series indicators over argumentative zoning and citation sentiment indicators.

In addition to the indicator-level ablations, we also looked at individual feature performance using their odds ratios. While the ablations show the overall contribution of an indicator (which combines multiple features), all indicators contain important individual features. For example, although TSA:networks is not among the highest performing indicators, some of its member features (e.g., the slope of the growth function best fitted to the article citation count) are among the best overall. Similarly, the total counts of the AIM and OWN categories from Argumentative Zoning, among others, are some of the most powerful features. Finally, we observe that for the vast majority of the time-series indicators, model-based features perform better than the statistical measures that capture global characteristics of the time series. In Chapter 6, we use these insights and focus solely on the simplest model-based features to facilitate temporal feature extraction on a problem involving large-scale time-series datasets.

5.6 Conclusions

In this chapter, we demonstrated the impact of exploiting methods from time-series analysis to significantly improve the effectiveness of domain-specific prediction models. Together with a large team of researchers at Columbia and several other universities, we developed a system to tackle the ambitious problem of predicting the future impact of scientific concepts. Regarding our sole contributions in such large-scale effort, we had a prominent role in the design and development of the system in general (see Section 5.2), as well as of the time-series analysis component (see Section 5.3.2). In addition, along with two other students, we performed the large-scale experimental evaluation that appears in Sections 5.4 and 5.5.

We summarize the contributions of this chapter as follows: (i) we presented the architecture of the system we built to extract characteristics from articles in the scientific literature and to perform real-time predictions of the future impact of scientific concepts (Section 5.2); (ii) we described the large number of characteristics that our system extracts from the metadata and the full text of scientific articles (Section 5.3.1); (iii) we presented a principled methodology to capture patterns in short, sparse, and noisy time series that we implemented in the time-series analysis component, a critical component of our system (Section 5.3.2); (iv) we described a simple, yet effective prediction model of our system (Section 5.3.3); and (v) we performed a large-scale experimental evaluation to study the importance of considering features over the full-text of scientific articles and to measure the effectiveness of temporal features (Sections 5.4 and 5.5).

Our results show the clear benefit of full text features over metadata and that temporal features contribute the most in predictions. Specifically, when prediction is performed on a dataset including only full text articles, a system that exploits features drawn from full text performs significantly better than a system that only uses metadata features. The addition of all data in WOS yields an improved performance of metadata features, both in the metadata only performance and in the full feature performance. Nonetheless, across all metrics, the text features are so strong that even in this scenario, where metadata features are computed over all documents relevant to a term while text features are computed over only a subset of the relevant documents, the model based on metadata alone cannot outperform text features. The benefit of the analysis of the full text of scientific articles is well worth the increased performance cost of the natural language analysis. Importantly, across all experiments, combinations of full text features and metadata features with time-series relevant features yielded the greatest impact than any other indicator considered in our analysis.

This is of particular importance, considering the lightweight methodology used in our TSA component. In previous chapters of this dissertation, we developed effective methods for tasks that rely on standalone time-series collections. Instead, in this chapter we focused on a method to (i) combine features produced by different methods over different types of data; and (ii) construct more advanced features that can reason about the temporal relationships, trends, and patterns of simpler features. We believe such simple methodology is well-suited for applications where underlying operations produce thousands of features and, therefore, an exhaustive combination of the features would be prohibitively expensive.

In the next chapter, we use a similar methodology to capture temporal characteristics over information extracted from web search logs to predict early web searchers experiencing devastating diseases.

Chapter 6

Detecting Devastating Diseases in Search Logs

In Chapter 5, we developed a lightweight methodology to extract features from short, sparse, and noisy time series with minimum preprocessing and storage requirements. In addition, we deployed that methodology as a critical component of an end-to-end system for predicting the future impact of scientific concepts. Our results demonstrated the importance of exploiting methods from time-series analysis to significantly improve the effectiveness of prediction models for this task. In this chapter, we rely on a similar methodology to extract features from short, sparse, and noisy time series that often appear in the analysis of web search engine logs. Specifically, together with Eric Horvitz and Ryen White, our collaborators from Microsoft Research, we explore the promise of harnessing temporal patterns in behavioral signals extracted from web search engine logs for early detection of devastating diseases. First, we provide an overview of the problem and its challenges (Section 6.1). Second, we review the necessary background regarding the ontology of symptoms and the process to create the dataset utilized in this study (Section 6.2). Then, we describe the large number of characteristics that we extract from such dataset and present the methodology to track the temporal relationships and patterns in the content of queries over time (Section 6.3). Finally, we summarize the results of our large-scale experimental evaluation (Section

6.4) and conclude with the contributions of this chapter (Section 6.5).

6.1 Overview and Motivation

Web search is a primary resource for people concerned about the significance of health-related symptoms [Fox and Duggan, 2013]. Researchers have studied symptom and illness-related searches in pursuit of insights about how people search about health concerns, including patterns of querying and review of information in pursuit of diagnoses [White and Horvitz, 2012], healthcare utilization signals [White and Horvitz, 2014], traces of therapeutic decision making for challenging illnesses [Paul et al., 2014], and identification of new adverse effects of medications [White et al., 2013]. Prior studies have examined how population-level signals in social media can be used to detect the emergence of diseases [De Choudhury et al., 2013; Ginsberg et al., 2009].

We explore the prospect of harnessing anonymized long-term sequences of health-related search queries to yield information that could provide valuable signals for detection of illness in advance of traditional diagnosis. Leveraging online behavioral data to provide earlier detection of a disease or of the raised risk of illness on a large scale can make significant contributions to healthcare. Better outcomes can be achieved by earlier confirmation of illnesses and risks via gaining access to more timely diagnoses, treatments, and other proactive interventions. As an example, such capabilities might help to identify those at significant risk of suffering the onset of chronic disease processes, such as diabetes or heart disease, or the rise of acute processes such as atrial fibrillation or more severe cardiac arrhythmias. Interventional programs ranging from changes in diet or exercise to taking (or avoiding) certain medications can yield significant health benefits.

The diagnosis for certain medical conditions can be particularly devastating if the chances of survival are typically low at the time of diagnosis. Survival rates may be improved significantly via earlier detection and treatment. With many cancers, screening methods can be effective for early detection and therapy [Pepe et al., 2001], but they involve ex-

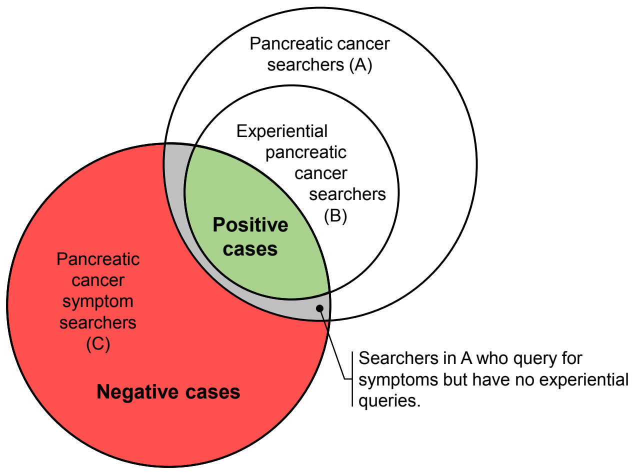

Pancreatic cancer searchers (A)
Experiential pancreatic cancer searchers (B)
Positive cases
Pancreatic cancer symptom searchers (C)
Negative cases
Searchers in A who query for symptoms but have no experiential queries.

Figure 6.1: Venn diagram depicting sets of users employed in analyses: pancreatic cancer searchers (A), pancreatic cancer searchers exhibiting experiential diagnostic queries (B), and those who search for the symptoms of pancreatic cancer (C). $|A \cup C|$ (i.e., the number of users in the original, pre-filtered dataset) is 9.2 million. Positives and negatives are sourced from $B \cap C$ and $C \setminus A$, respectively. Relative set sizes are not to scale.

plicit testing prompted by policies around risk factors such as family history [Lynch et al., 1996] or medical history [Everhart and Wright, 1995; Lowenfels et al., 1993]. Ideally, health screening systems would be able to observe people passively as they engage in their normal activities and alert them (per their preferences on vigilance) to potential health risks without requiring the investment of time and effort in special screening activities. Individual-level postings on social media have been mined for this purpose [De Choudhury et al., 2013; Sadilek et al., 2012], but may not have representative coverage of symptoms associated with social stigma [De Choudhury et al., 2014].

We present a study of the feasibility of doing early detection of devastating diseases based on large-scale logs of health-related web search activity. An important aspect of our analysis is to consider the content of queries over time, and the prospect that temporal relationships and patterns among queries over multiple sessions over several months provide subtle fingerprints of lurking illness. We rely on the methodology developed in Chapter 5 to analyze the evolution of features extracted from query logs over time using a variety of principled time-series analysis methods. We focus on the early detection of the presence of pancreatic cancer, a devastating diagnosis given the typical progression of the disease to an inoperable situation by the time it is found. Our work showcases the potential value of innovative approaches for speeding up the time to diagnosis of this deadly disease.

Pancreatic cancer is the fourth leading cause of cancer death in men and women in the United States and the sixth leading cause in Europe [Michaud, 2004]. The illness is widely known as being difficult to detect and is frequently diagnosed too late to be treated effectively [Hruban et al., 2000; Li et al., 2004]. A recent study found that the progression of pancreatic cancer from stage I to stage IV happens in just over one year [Yu et al., 2015]. Approximately 75% of pancreatic cancer patients die within a year of diagnosis, and only about 4% survive for five years post diagnosis. Exploration of the possibility that a patient has pancreatic cancer involves a careful and costly consideration of history, labwork, and imaging studies (in contrast to the passive screening methods described in this chapter) [Rulyak et al., 2003]. Screening is largely performed to detect the disease at an early phase (pre-invasive or early invasive) when it is still curable by surgical intervention and chemotherapy. Earlier diagnosis of pancreatic cancer improves the feasibility of discovering the illness at an earlier stage [Chari et al., 2015]. For patients diagnosed early who undergo curative surgery (e.g., a Whipple procedure), five-year survival rate is higher, but it remains less than 25% [Yeo et al., 1997].

We take as a proxy for ground truth of a diagnosis of pancreatic cancer the detection of *experiential diagnostic* queries issued by searchers. Experiential queries show strong evidence of being linked to the actual presence of symptomatology or conditions versus less directly involved, more distant *exploratory* queries seeking information about symptoms or diseases [Paul et al., 2014]. Experiential diagnostic queries for pancreatic cancer are identified via consideration of the structure of queries and of patterns of information gathering over multiple users in search logs. Experiential queries often include first-person assertions such as [*i was just diagnosed with pancreatic cancer*], which when associated with prior queries about symptoms identifies the positive cases.

We construct models to predict the future rise of experiential queries from longitudinal search data. Figure 6.1 shows the different subsets of users in our analysis, including people who search for pancreatic cancer (A), the subset of these searchers who issue experiential diagnostic queries (B), and those who search for a set of symptoms linked to pancreatic cancer (C). Those who *only* search for one or more related symptoms with no evidence of pancreatic cancer searching constitute the negative cases. We conduct a large-scale evaluation to address two questions: (i) whether we can detect cases where people show evidence of being diagnosed with pancreatic cancer many months in advance of their experiential diagnostic queries; and (ii) whether considering the temporal relationships and patterns of features extracted from query logs over time improves the accuracy of our prediction models. Our results affirmatively answer both questions. Specifically, we find that our methods can detect cases where people show evidence of being diagnosed with pancreatic cancer many months in advance of their experiential diagnostic queries. In addition, our experiments suggest that it is well worth the effort to track features over time and analyze their temporal relationships because such features are critically important for this prediction task.

With this research, we introduce the early detection of diseases as a promising new application of search log mining and machine learning that scales to millions of searchers. We continue as follows:

- We describe the process of generating a large-scale real-world dataset to study the feasibility of early detecting experiential pancreatic cancer cases from longitudinal individual search activity (Section 6.2).
- We review the large set of characteristics that we extract from query timelines (Section 6.3.1).
- We present a simple methodology to track the temporal relationships and patterns in the content of queries over time (Section 6.3.2).
- We summarize our prediction model to forecast with significant lead times that users will later input experiential queries for pancreatic cancer (Section 6.3.3).

- We explore the influence of different factors, such as the lead time or the presence of specific symptoms in the search activity, on the predictive performance of our learned models, including true positive rates when false positive rates are strictly controlled (Section 6.4).

Controlling false positives is especially important to reduce unnecessary costs and concerns given potential future applications such as providing early warnings and suggestions to searchers about undertaking more formal screenings. Finally, we conclude and discuss the implications of our work (Section 6.5). The material described in this chapter appears in [Paparrizos et al., 2016b; Paparrizos et al., 2016a].

We now describe the dataset creation process.

6.2 Dataset Creation to Study Pancreatic Cancer Searchers

We first provide an overview of the dataset used, starting with a description of the query logs (Section 6.2.1). We then discuss the creation of an ontology with symptoms commonly experienced by people with pancreatic cancer (Section 6.2.2) and provide details on extracting pancreatic cancer and symptom searchers (Section 6.2.3). We review the augmentation, tagging, and filtering steps for our dataset (Section 6.2.4). Finally, we summarize the creation of query timelines for the positive cases (i.e., experiential diagnostic searchers who also search for pancreatic cancer symptoms) and the negative cases (i.e., those who only search for the symptoms) (Section 6.2.5). Since reliable labels cannot be determined for the non-experiential pancreatic cancer searchers, we exclude them to create a cleaner dataset for training and testing. We show later (see Section 6.4.6) that predictive performance is largely unchanged if these searchers are included as negative examples during the application of the model in a realistic scenario.

Type	Name	Example synonyms
Symptom	back pain	pain in lower back, lowback pain
	blood clot	blood clots, thrombosis
	dark or tarry stool	dark poop, tarry feces
	dark urine	orange pee, brown urine
	enlarged gall bladder	swollen or inflamed gallbladder
	floating stool	floating stool, floaters
	greasy stool	greasy poop, oily feces
	high blood sugar	sudden diabetes
	itchy skin	skin itching, hands itchy
	yellow skin or eyes	jaundice, yellow eyes
	light stool	pale stool, white crap
	loss of appetite	poor appetite, not hungry
	nausea or vomiting	throwing up, nauseous
	smelly stool	stinky feces, smelly poop
	sudden weight loss	unexplained weight loss
	taste changes	changes in taste, dysgeusia
	loose stool	loose feces, loose stool, diarrhea
	constipation	constipated, backed up
	indigestion	acid reflux, heartburn
	abdominal swelling or pressure	swollen stomach
	abdominal pain	belly pain, stomach ache
Risk factor	alcoholism	heavy drinking, alcoholic
	hepatitis	hep b, hep c
	pancreatitis	–
	ulcers	ulcer
	obesity	obese, very fat, extremely fat
	smoking	smoker, cigarette, cigar
	chills or fever	chills, fever
	multiple endocrine neoplasia	men1
	nonpolyposis colorectal cancer	lynch syndrome, hnpcc
	von hippel-lindau syndrome	hippel-lindau syndrome
	intestinal polyposis syndrome	peutz-jeghers syndrome
	mole melanoma syndrome	fammm, b-k mole syndrome

Table 6.1: Ontology with symptoms, risk factors, and examples of associated synonyms for pancreatic cancer.

6.2.1 Anonymized Web Search Engine Logs

Search engines track various characteristics during their interaction with users so as to better capture information needs, improve their responses, and personalize the content. Every such interaction corresponds to a log entry that includes a unique, anonymized user identifier based on a web browser cookie. This enables the extraction of the search history comprising queries and clicks from an identifier for up to 18 months. Note that the identifier may comprise the search activity of multiple users on shared machines and does not consolidate activity from a user across multiple machines. We use the logs of a randomly-selected subset of Bing search engine users in the English-speaking United States locale from October 2013 to May 2015 inclusive.

6.2.2 Ontology of Symptoms and Risk Factors

Warning signs and symptoms for pancreatic cancer usually include generic, subtle signs and symptoms, such as abdominal and back pain, loss of appetite, and unexplained weight loss. We performed an extensive review of possible signs, symptoms, and risk factors associated with pancreatic cancer and developed an ontology with 21 categories of symptoms. This manually-curated ontology consists of two levels. The first level includes the names of the symptoms and the second level includes multiple names, synonyms, and expressions with which the corresponding symptom in the first level may appear in our data. We performed multiple iterations of refinements of this ontology to remove noise and to minimize erroneous query matches. Table 6.1 presents the 21 symptom categories with some representative examples of associated query expressions. Also shown are 12 risk factors and associated synonyms, derived from the literature (e.g., [Lowenfels and Maisonneuve, 2006]), describing attributes, characteristics, or exposures that may increase the likelihood of pancreatic cancer. The symptoms and the risk factors are featurized in predictive models, and they are also used in policies to determine when predictive models should be applied (see Section 6.4.5).

6.2.3 Extraction of Pancreatic Cancer Searchers and Symptom Searchers

In order to identify positive and negative cases for the generation of our learned model, we built a dataset comprising two groups of users (Figure 6.1). The *pancreatic cancer searchers* group, denoted as A in the figure, includes all searchers with at least one query explicitly on pancreatic cancer (i.e., a query matches this expression [(‘pancreas’ *OR* ‘pancreatic’) *AND* ‘cancer’]). The *symptom searchers* group, denoted as C, includes all users with at least one query related to symptoms linked to pancreatic cancer, as captured by the symptoms and synonyms described in Section 6.2.2.

Having unique identifiers for each user in the union of A and C (i.e., $A \cup C$) permits the extraction of the full query histories of 9.2 million searchers. We first sought to remove searchers who are likely healthcare professionals (HCPs). To do this, we employed a proprietary Bing classifier that identifies health-related queries to remove users from the study for whom 20% or more of queries are health related. This threshold was based on a prior analysis of identifying health professionals in search logs [White et al., 2014].

6.2.4 Dataset Augmentation, Tagging, and Filtering Steps

Age and gender are important factors associated with developing pancreatic cancer [Lowenfels and Maisonneuve, 2006; Michaud, 2004]. As such, we augmented the dataset with demographic information from proprietary search engine classifiers that estimate age (discretized as < 18, 18–24, 25–34, 35–50, or 50–85) and the gender for each user. The classifiers are trained on data where ground truth of demographic details are provided explicitly by users. The predictions are based on signals derived from searchers' long-term search activity, including their search queries and web domains of their clicked results. Since pancreatic cancer incidence rates vary by geographic location, we also annotated searchers with the U.S. state from which they searched most (based on reverse internet provider (IP) lookup data).

Beyond the demographic information, we are also interested in the subject matter of the queries and results that were visited over searchers' timelines. We augmented each query and corresponding clicked websites with their estimated Open Directory Project (ODP, dmoz.org) category. We used a text-based classifier, similar to [Bennett et al., 2010], that uses logistic regression to predict the ODP categories. When optimized for the score in each category, this classifier has a micro-averaged F-measure score of 0.60. For queries, the ODP category is that of the top-ranked search result. The remaining users after the augmentation and filtering steps total 7.4 million, from which $479,787$ are pancreatic cancer searchers.

6.2.5 Extraction of Positive and Negative Pancreatic Cancer Cases

We create *query timelines* for experiential pancreatic cancer searchers and experiential symptom searchers, which we then featurize for the early detection task. Figure 6.2 summarizes the strategies for identifying positives and negatives. To avoid including users with

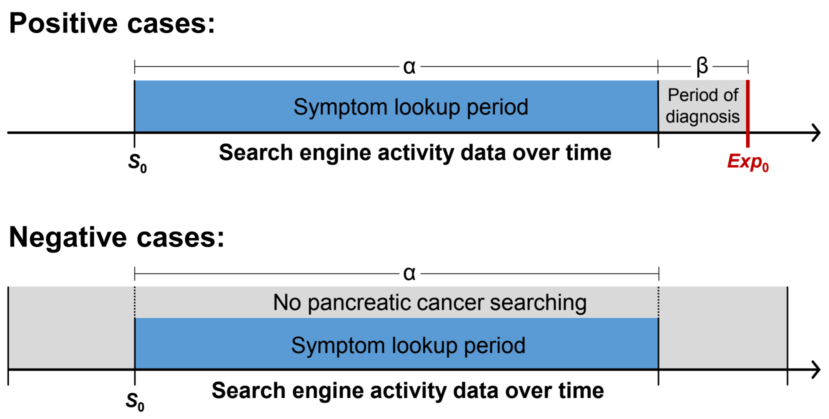

Positive cases:
α
β
Symptom lookup period
Period of diagnosis
Search engine activity data over time
S0
Exp0
Negative cases:
α
No pancreatic cancer searching
Symptom lookup period
Search engine activity data over time
S0

Figure 6.2: Schematic illustration of the query timelines used in the selection of positive and negative cases. S_0 refers to the first symptom query and Exp_0 is the first experiential diagnostic query. α is the duration of the symptom lookup period, which was approximately equal in the aggregate for the positives and negatives. β is the duration of the period of diagnosis, set to one week in this study.

very short histories, we filter out all users with fewer than five search sessions[1] spanning five different days. This reduced the population to 6.4 million users, with a mean total duration (time from first to last query for a user) of 210.32 days, standard deviation (SD) of 182.93 days, and interquartile range of 120 days.

Positive Cases: To identify experiential pancreatic cancer users, we created a set of first-person diagnostic queries for pancreatic cancer (denoted Exp_0). Some examples of such diagnostic queries are [*just diagnosed with pancreatic cancer*], [*why did i get cancer in pancreas*], and [*i was told i have pancreatic cancer what to expect*].

From the set of $479,787$ pancreatic cancer searchers, $3,203$ match the pattern of diagnostic queries. In order to consider them as experiential users, we require them to have searched at least for one symptom *prior to* the diagnosis query. This step generates a set of $1,072$ query timelines of experiential searchers that contain periods of *symptom lookup* followed by the diagnostic query. The symptom lookup period starts when the first symptom is detected as matching terms represented in the symptom ontology. For positive cases, the symptom lookup period completes at least one week before diagnosis (i.e., we consider the week before the diagnostic query as the period of diagnosis in real life and do not count

[1] *Session* is a query sequence with at most 30 minutes between queries [White and Drucker, 2007].

queries in that time since they may be polluted with ground truth signals). For reliability, especially given the need to compute *Temporal* features (Section 6.3.1), the minimum time period for which our features are computed is four weeks.

Negative Cases: To generate the negative set, we sample from the users who searched for pancreatic cancer symptoms but did not search for pancreatic cancer anywhere in their timeline (i.e., $C \setminus A$), either before or after the symptom lookup period. Performing this additional check on data from outside the lookup period is required to increase the likelihood that the negative cases are indeed negative. Users who searched for pancreatic cancer and its symptoms, but did not issue an experiential query (the gray subset in Figure 6.1 representing $(A \cap C) \setminus B$), were excluded since a label could not be reliably determined. In Section 6.4.6 we describe an additional experiment where pancreatic cancer searchers were included during model testing.

We were concerned that rudimentary behavioral differences that may reflect artifacts in the data could invalidate the learning task. For example, if our experiential users were just more active generally, then a feature that computed the total number of queries would have strong predictive value, yet would be uninteresting scientifically. We sought to address this by downsampling the negative cases to attain a similar distribution of symptom lookup periods in terms of the temporal duration and query volume as observed for the positive cases.[2] We did this by selecting users with a symptom lookup period duration within three standard deviations of the mean of the positive cases. This reduces the number of negative cases to 3,025,046. Table 6.2 presents the summary statistics on the symptom lookup periods in terms of the number of days and the number of queries in the two datasets. The table shows that the distributions for positive and negative cases (in terms of number of days and number of queries) are similar. The distributions are statistically indistinguishable using two-sample Kolmogorov-Smirnov tests for temporal duration ($D = 0.005$; $p = 0.7017$) and number of queries ($D = 0.003$; $p = 0.7681$), even though the latter

[2] We could also have addressed this by making all features relative percentages. Sampling provided more flexibility in feature construction. As an additional check, we included features such as the number of queries in the symptom lookup period; those were found to carry little evidential weight in the learned model.

Class	Duration (days)		Total # queries		N
	M	SD	M	SD	
Positives	109.34	49.66	380.66	150.83	1,072
Negatives	108.04	48.35	378.09	151.01	3,025,046

Table 6.2: Summary statistics, namely, mean (M), standard deviation (SD), and number of cases (N), of durations and numbers of queries in positive and negative datasets.

was not a filtering criterion. We note that query timelines are not aligned: the absolute point in time where people issue the experiential diagnostic query, and the accompanying symptom lookup period can differ between searchers.

6.3 Early Detection of Experiential Searchers

Our objective is to develop a methodology to extract characteristics from web search engine query logs and to analyze the evolution of these static characteristics over time, as we did in Chapter 5. To achieve that goal, we first describe the large set of characteristics that our system extracts from the query timelines (Section 6.3.1). Then, we present a variety of principled statistical measures capable of capturing patterns in time-varying characteristics (Section 6.3.2). Then, we describe the problem of focus and review our prediction model (Section 6.3.3)

6.3.1 Feature Extraction for Query Timelines

We group our static features into four major different categories: (i) demographic information about the user; (ii) characteristics about user sessions, query classes, and URL classes; (iii) characteristics about symptoms; and (iv) risk factors. Each category may contain multiple subcategories, as we explain next. We describe the temporal features in Section 6.3.2. Table 6.3 summarizes the features.

Category	**Name**	**Description**
Demographic	AgeIs_X	Age prediction based on queries and clicks
	AgeClassProbability_X	Probability of classifier for this age class
	GenderIs_X	Gender prediction based on queries and clicks
	GenderProbability_X	Probability of classifier for this gender class
	LocationIs_X	U.S. state from which the user most actively searches
SearchHistory	# Queries (Q)	Number of queries issued by the user
	# DistinctQ	Number of distinct queries issued by the user
	# DistinctSessions (S)	Number of sessions
	# Clicks (C)	Number of times a query led to a click
	# QWithCategories	Number of queries for which we have ODP category

	% QInCategories	Percent of queries for which we have ODP categories
	RatioOfCPerQ	Ratio of clicks to queries
QueryTopic	QCategoryIs__X	User has queries in ODP category
	# QInCategory__X	Number of queries in ODP category
	% QInCategory__X	Percent of queries in ODP category w.r.t all queries
	% ODPQInCategory__X	Percent of ODP queries w.r.t all ODP queries
	# QInCategoryW__X	Weighted version with classifier's probability as weight
	% QInCategoryW__X	Weighted version with classifier's probability as weight
	% ODPQInCategoryW__X	Weighted version with classifier's probability as weight
	# SInCategory__X	Number of sessions in ODP category
	% SInCategory__X	Percent of sessions in ODP category
	AvgTimeQCategory__X	Average time span between ODP queries
	StDevTimeQCategory__X	Standard deviation of that time span
	AvgTimeSequentialQ__X	Average number of sequential queries in ODP category
	StDevTimeSequentialQ__X	Standard deviation of that number of queries
	# Qfrom0to6InODP__X	Number of ODP queries during early morning
	# Qfrom6to12InODP__X	Number of ODP queries during late morning
	# Qfrom12to18InODP__X	Number of ODP queries during evening
	# Qfrom18to24InODP__X	Number of ODP queries during afternoon/night
URLTopic	CInURLCategory__X	User has clicked in URL of that category
	# CInURLCategory__X	Number of times user clicked in URL of that category
	RatioOfQinURLODP__X	Ratio of clicked urls in ODP w.r.t. number of queries
	% CInURLODP__X	% of clicked urls in ODP category w.r.t # of clicks
	# CInURLODPW__X	Weighted version with classifier's probability as weight
	RatioOfQInURLODPW__X	Weighted version with classifier's probability as weight
	% CInURLODPW__X	Weighted version with classifier's probability as weight
	# SInURLODP__X	Number of sessions with clicks in that category
	RatioOfSInURLODP__X	Ratio of sessions with clicks in ODP w.r.t all sessions
QueryURLTopic	# QWithSymODP__X__Y	Number of queries with symptom X in Y ODP
	% QWithSymODP__X__Y	Percent of queries with symptom X in Y ODP
	# QSymLedToC__X__Y	Number of queries with symptom X led to URL of Y
	% CSymLedToC__X__Y	Percent of queries with symptom X led to URL of Y
	# QODPLedToC__X__Y	Number of queries in query class X led to URL of Y
	% CODPLedToC__X__Y	Percent of queries in query class X led to URL of Y
SymptomGeneral	# SymptomQ	Number of queries matching our symptoms
	# DistinctSymptoms	Number of distinct symptoms
	# DistinctSymVariants	Number of distinct variants of symptoms
	% QForSymptoms	Percent of queries matching symptoms
	# QWithManySymptoms	Number of queries matching multiple symptoms
	% QWithManySymptoms	Percent of queries matching multiple symptoms
	% SWithSymptomQ	Percent of sessions matching a symptom
	% SWithManySymptomQ	Percent of sessions matching multiple symptoms
	AvgTimeSymptoms	Average time span between queries matching symptoms
	StDevTimeSymptoms	Standard deviation of such time
	AvgSymptomPerDay	Average number of queries matching symptoms per day
	StDevSymptomPerDay	Standard deviation of such number
	MaxSymptomPerDay	Max of such number
	% DaysWithSymptom	Percent of queries matching symptoms per day
SymptomSpecific	HasSymptom__X	User has query matching symptom category X
	# QForSymptom__X	Number of queries matching symptom category X
	% QForSymptom__X	Percent of queries matching symptom category X
	# SForSymptom__X	Number of sessions with queries in X symptom category
	% SForSymptom__X	Number of sessions with queries in X symptom category
	AvgTimeSymptom__X	Average time between queries matching that symptom
	StDevTimeSymptom__X	Standard deviation of that time
	# Qfrom0to6InSym__X	Number of X symptom-queries during early morning
	# Qfrom6to12InSym__X	Number of X symptom-queries during late morning
	# Qfrom12to18InSym__X	Number of X symptom-queries during during evening
	# Qfrom18to24InSym__X	Number of X symptom-queries during afternoon/night
Risk Factor	HasRiskFactor__X	User has query matching that risk factor
	# QForRF__X	Number of queries matching that risk factor
	% QForRF__X	Percent of queries matching that risk factor
	# DistinctRF	Number of distinct risk factors
	SymptomSequence__X__Y	If symptom X appeared before symptom Y

Temporal

	SymptomSequence_Y_X	If symptom Y appeared before symptom X
	SlopeOfQInODP_X	Slope of fitted functions in queries for X
	CumSlopeQInODP_X	Slope of fitted functions in cumulative queries for X

Table 6.3: Features used for early detection task, grouped by feature category.

Demographics: Cancer statistics from the U.S. National Cancer Institute[3] show that pancreatic cancer is more common with increasing age, is slightly more common in men than in women, and varies by geographic location. As such, we develop features related to the demographics of the users. In particular, we use the estimated age bucket and gender (see Section 6.2.4) along with the classifier's probabilities as confidence values. The dominant location (U.S. state) of a searcher is also included as a feature.

Search Characteristics: People express their information needs and preferences through queries and click behavior (i.e., the website visits). We extract various features to capture these search and retrieval activities. As we discussed previously, the queries, as well as the visited websites, were tagged with their ODP category in an attempt to identify domains of interest (see Section 6.2.4). A first set of features *SearchHistory* contains several generic statistics, such as counts, ratios, and percentages, which are characteristics of the global behavior of the user. For example, we compute the number of queries, sessions, and clicks, as well as ratios of clicks per query for each user. Then, we compute a large number of features with respect to the ODP categories of queries *QueryTopic*, clicked search results *URLTopic*, and the combination *QueryURLTopic*. These include compute counts and percentage of queries and sessions in each ODP category, the average time until queries appear in the same category, as well as the time of the day that queries appear in each category. Similar features are also computed for each category of the visited websites (e.g., counts, ratios, and percentages of visited websites that belong to each category). We additionally compute features to characterize the user sessions, including features that capture the click behavior of users associated with queries. For example, we compute counts and percentages of all the combinations of query categories that led to visits in website categories.

Symptoms: The features described above attempt to capture generic characteristics from

[3]http://seer.cancer.gov/statfacts/html/pancreas.html

user sessions. However, for the problem of interest, we seek to also leverage features from queries containing terms captured in the symptom ontology for pancreatic cancer (Section 6.2.2). The symptom features are divided into two classes: (i) *SymptomGeneric* and (ii) *SymptomSpecific*. Generic symptom features contain counts and percentages for the queries and sessions matching symptoms in our ontology, the average time between symptom queries, as well as the average number of symptom queries that are issued daily. Specific symptom features are generated per symptom category. For example, for each symptom, we compute counts and percentages of appearance, the time between distinct symptoms, and the time of day such symptom queries are issued. As with the user session features, we combine symptoms to capture the click behavior and, hence, we compute counts and percentages of each symptom query leading to a visit on a website belonging to particular ODP categories.

Risk Factors: This class contains features related to the presence of terms representing risk factors in the symptom lookup period. For each risk factor, we note its presence or absence, and also the number of queries containing that risk factor and the fraction of all queries from that user that these risk factor queries represents. Total number of distinct risk factors in the symptom lookup period is also a feature.

6.3.2 Feature Aggregation and Temporal Pattern Extraction

All previous features produce aggregated statistics over the full time window under consideration. We include a subset of the features we exploited in Chapter 5, to capture the temporal variation of these statistics over misaligned query timelines with noise and missing values. For every feature extracted in Section 6.3.1, we generate a time series with points that represent aggregated values for intervals of the time window. For example, each feature can be computed per month, per week, or per day, depending on the level of granularity we seek to capture. Since the occurrence of specific features can be very sparse, we set the time window to four weeks for temporal features. For features that are not percentages or ratios, we also compute the cumulative time series. For each time series, we use the first coefficient of the linear least-square estimates to devise features that capture the trend (i.e., increasing, decreasing, and unchanged) and the rate of change (i.e., slope). We omit features capturing global characteristics (see Section 5.3.2) that in our preliminary analysis did not provide additional benefit. Finally, we define domain-specific temporal features to capture the *sequence* in which symptoms appear in query timelines.

6.3.3 Problem Description and Prediction Model

We address the problem of early detection of experiential searchers for pancreatic cancer via anonymized web search engine logs. We cast this as a binary classification task, where the model is trained on features extracted from search log query timelines of experiential pancreatic cancer searchers and symptom-only searchers. We focus on maintaining very low false-positive rates (i.e., 1 misprediction in $100,000$ correctly identified cases) while keeping high the imbalance ratio of positive and negative cases (i.e., one thousand positives vs. millions of negative cases); these properties are important for potential future large-scale real-world applications such as an alerting mechanism in search engines.

The prediction model uses the features outlined in the previous section, computed for each searcher, to make predictions about the future occurrence of experiential diagnostic searches in each searcher‘s query timeline. We use gradient boosted trees [Friedman, 2001], which employ an ensemble of decision trees to construct a better learned model. Advantages include the ability to capture non-linear relationships, model interpretability (e.g., a ranked list of important features is generated), facility for rapid training and testing, and robustness against noisy labels and missing values. We experimented with different learning algorithms, but gradient boosted trees yielded superior accuracy.

6.4 Experimental Results

We now present the findings of our experiments. We report the overall performance in Section 6.4.1 and the performance as we increase the lead time before the first experiential

Weeks before Exp_0	TPR (as %) at FPRs ranging from 0.00001–0.1					AUROC
	0.00001	0.0001	0.001	0.01	0.1	
1 week	7.122	10.386	20.772	36.202	71.810	0.9112
5 weeks	7.122	10.979	20.178	34.421	70.620	0.9047
9 weeks	7.122	10.683	18.991*	33.234*	70.023	0.8854*
13 weeks	7.122	9.792	17.804*	32.937*	67.359*	0.8700*
17 weeks	6.825	9.199*	17.209*	32.640**	64.688**	0.8539**
21 weeks	6.528*	9.199*	16.319**	32.345**	61.424***	0.8315**

Table 6.4: Performance at four-week intervals for users where features can be computed from Exp_0 – 1 week to Exp_0 – 21 weeks. Values averaged across the ten folds of cross-validation. The significance of differences in AUROC and TPR using paired t-tests for each week versus $Exp_0 - 1$ indicated * $p < 0.01$, ** $p < 0.001$, and *** $p < 0.0001$. Weeks denote lead time before Exp_0 (β in Figure 6.2).

diagnostic query (Section 6.4.2). We then inspect the model to understand the contributions that each feature makes towards early detection (Section 6.4.3) and the performance of different feature *classes* (Section 6.4.4). Then, we examine the effect on model performance of conditioning on symptoms and risk factors (Section 6.4.5) and consider a realistic deployment scenario (Section 6.4.6). Finally, we discuss limitations of our approach (Section 6.4.7). Throughout the experimental evaluation, we use the area under the receiver operating characteristic curve (AUROC) and recall (TPR, true positive rate) at fixed, extremely low false positive rates (FPRs) as our primary evaluation metrics. Additionally, we apply 10-fold cross validation, stratified by user to evaluate the generalizability of the model when applied to new users. Significance level is $p < 0.05$ unless otherwise stated.

6.4.1 Overall Performance of Prediction Model

The overall performance of the classifier in making predictions based on data up to the beginning of the period of diagnosis (i.e., Exp_0 – 1 week) in AUROC is 0.9003. Given that low error rates would be vital in practice to avoid unnecessary patient alarm, we focus on the true positive rate (fraction of all positives that are recalled by the model) at low false positive rates (FPR). Focusing on FPR in the range 0.00001–0.01, the model is able to recall 5–30% of the positive cases, depending on the specific FPR. We see this performance as promising given the limited information (primarily search-related activity) available to the model.

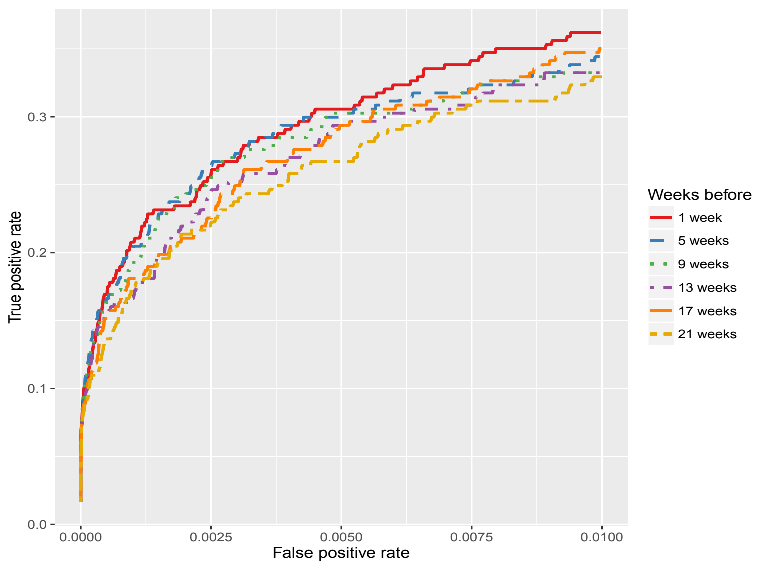


Figure 6.3: Average partial ROC curves in the FPR range 0–0.01, for models learned using data up to 21 weeks before the first experiential diagnostic query (error bars excluded for clarity). Variance in FPR and TPR is minor.

6.4.2 Evaluation of Model with Different Lead Times

A key part of early detection is being able to predict the emergence of the disease well in advance. To understand how the prediction performance changes as we move further back in time before the first experiential diagnostic query, we selected the set of 337 positive searchers and 945,394 negative searchers who were still observed in the logs many weeks prior to the experiential diagnostic query. We report results from one week before the experiential diagnostic query, all the way up to 21 weeks before the diagnostic query. To count as being present at $Exp_0 - 21$ weeks, a searcher needs to have symptom queries extending back at least four weeks before that point (i.e., to $Exp_0 - 25$ weeks, or approximately six months before the first experiential diagnostic query).

We trained a model for these users in the same way as we did in Section 6.4.1. The ratio between positive and negative searchers remains similar to that for all users (i.e., approximately 1:3000). Table 6.4 reports the TPR at different false positive rates for this same set of users at different four-week increments, as well as the AUROC. The general trend is that the performance drops fairly consistently as we increase the lead time, but

Feature	Weight	Direction	Class
NumOfDistinctSymptoms	1.0000	Positive	SymptomGeneral
NumOfQueriesInHealthCategory	0.8253	Positive	QueryTopic
NumOfDistinctSymptomsVariants	0.6899	Positive	SymptomGeneral
AgeClassProbability_5085	0.6889	Positive	Demographic
HasBackPain	0.6622	Negative	SymptomSpecific
HasIndigestion	0.6432	Negative	SymptomSpecific
HasIndigestionThenAbdominalPain	0.6349	Positive	Temporal
SlopeNumOfDistinctSymptoms	0.6154	Positive	Temporal
HasBackPainThenYellowSkinOrEyes	0.6004	Positive	Temporal
AgeClassProbability_LT18	0.5869	Negative	Demographic

Table 6.5: Top 10 features by evidential weight relative to the top feature. “Positive” or “Negative” direction means that a feature correlates positively or negatively, respectively, with the rise of experiential queries.

even 20 or so weeks before the first experiential diagnostic query the predictive performance is still quite strong (AUROC=0.8315, TPR=6.528% at FPR=0.00001). Assuming that pancreatic cancer progresses steadily from stage I to stage IV in just over one year (as has been previously reported [Yu et al., 2015]), accurate predictions 20 weeks in advance of the diagnostic period could lead to a sizable increase in the five-year survival rate (e.g., moving the point of diagnosis from Stage III to Stage II could increase the survival rate from 3% to 5–7% [American Cancer Society, 2014]).

Focusing on the FPR region from 0 to 0.01 (i.e., false positives occur fewer than 1 in 100 times) and visualizing that part of the ROC curve (Figure 6.3) we observe some clear differences in the performance of the models in this important region. The average normalized partial AUROC ranges from 0.292 ($Exp_0 - 1$ week) to 0.231 ($Exp_0 - 21$ weeks). All differences in AUROC for $Exp_0 - 5$ or more weeks versus $Exp_0 - 1$ week are significant ($p < 0.01$ using paired t-tests).

6.4.3 Contribution of Individual Features

In addition to understanding the overall performance, we are also interested in understanding the features that are most important in the learned model for predicting the future issuance of experiential queries. Table 6.5 shows the top 10 features with the highest weight, along with their weight relative to the top-ranked feature (*NumOfDistinctSymptoms*) and the feature class. The direction is based on the correlation between the feature value and the labels in the training data, using Pearson biserial correlation or the phi coefficient, depending on whether or not the feature data is binary. Table 6.5 shows that there is a broad range of features. The number of distinct pancreatic cancer symptoms was the most important feature. Temporal features representing changes over time and sequence ordering of symptom pairs are also important. Specifically, the trend and the slope of features capturing the appearance of symptom searches often correlate positively with experiential searchers in comparison to the trend and the slope of features capturing general purpose web search activity. Similarly, features capturing the order in which symptom searches appear often correlate positively with experiential searchers in comparison to features capturing individual symptom searches. For example, individual symptom features related to back pain and indigestion are important but have a negative influence on predicting future experiential queries, likely because (i) there are many explanations for why these symptoms appear in a query timeline, and (ii) they are positive for many negative cases (16.7% of negatives search for back pain, 7.4% search for indigestion). Finally, age is also important, and it is positively correlated if the searcher is older and is negatively correlated if they are younger.

6.4.4 Contribution of Feature Categories

Beyond the individual features, we can also consider the accuracy of the models based on feature classes. This can be particularly important when some classes of features are easy to obtain in practice. For example, the demographic features may be available for all searchers without the need to perform temporal modeling of query patterns. Table 6.6 presents the AUROC for models trained on each of the feature classes. The findings show that the *Temporal* class is particularly important, capturing the key role of temporal dynamics for this prediction task. The model is still accurate solely with access to demographics and basic features about general searching. However, performance improves substantially if we consider the specifics of the symptoms searched (*SymptomSpecific*) or the topics of the queries and results clicked (*QueryTopic*).

Feature Class	AUROC	TPR (as a %) at FPR=0.00001
Demographic	0.6565***	0.280%***
RiskFactor	0.6988***	0.653%**
SearchHistory	0.7202***	1.399%**
QueryURLTopic	0.7597**	2.052%**
SymptomGeneral	0.7672**	2.146%**
URLTopic	0.7753**	2.332%*
SymptomSpecific	0.8176*	2.800%*
QueryTopic	0.8137*	2.892%*
Temporal	0.8391*	2.985%*
Overall	0.9003	4.851%

Table 6.6: Performance of individual feature classes (as AUROC and TPR at a FPR of 0.00001) averaged across the 10 experimental folds. The performance differences against the *Overall* performance are statistically significant using paired t-tests at * $p < 0.01$, ** $p < 0.001$, and *** $p < 0.0001$.

6.4.5 Model Performance Conditioned on Symptoms and Risk Factors

We also considered the impact of the presence of symptoms and risk factors on the performance of the model.

- *Symptoms*: We filtered the positive and negative cases to those where a symptom was present in query timelines.

- *Risk factors*: These are risk factors corresponding to the presence of factors such as pancreatitis, smoking, and obesity, as well as cancer syndromes such as hereditary intestinal polyposis syndrome or familial atypical multiple mole melanoma syndrome (i.e., genetic disorders that predispose individuals to develop pancreatic cancer), all of which have been shown to lead to increased likelihood of developing pancreatic cancer [Fuchs et al., 1996; Goldstein et al., 1995; Lowenfels et al., 1993; Talamini et al., 1999].

Recall that our cross-validation was stratified by user. During cross validation, we learned a model on the users in the training folds and then for testing we limited to users with evidence of the specific symptoms or risk factors in their search history prior to the experiential diagnostic query. In each case the number of positives and negatives is less than the full set.[4] Table 6.7 presents statistics on the performance for each model where the number of

[4]An alternative would be to train a separate model for symptom or risk factor. An issue with doing that is there are insufficient positive examples about each dataset with which to train a robust model.

positive examples was at least 10 (to help ensure that AUROC calculations were meaningful). The table also presents TPRs at different false positive rates, as well as the percentage of positive or negative cases that have the symptom or risk factor searches. Finally, the last three columns shows the estimated number of true positives (capture) and false positives (cost) that would be observed, assuming a FPR of 0.00001, and the associated capture-cost ratio. Ideal targets for rates of capture versus cost in a deployed service can be derived via a decision analysis that considers the net expected value of the early detection and the expected costs of unnecessary anxiety. Such an optimization would leverage a careful characterization of the value of early intervention and details of designs of methods for engaging people.

Table 6.7 shows that focusing on users who search for risk factors such as smoking, hepatitis, and obesity leads to better overall performance. There were fewer than ten users searching for each of the cancer syndromes (e.g., hereditary nonpolyposis colorectal cancer) and, hence, they were excluded from Table 6.7. Focusing on the percentage of positives and negatives that contain each of the symptoms or risk factors, we observe that there are some that are much more likely to occur in positives (e.g., pancreatitis and smoking are 5.9 and 3.6 times as likely, respectively). Focusing on the utility, we find that if we set the FPR to 0.00001, overall we would find 52 positives in the union of positives and negatives at the expense of 30 negatives, who would be altered mistakenly. There are some symptoms and risk factors for which the capture-cost is more favorable. For example, in the case of alcoholism or obesity, we would find 20–30 times as many TPs as FPs. There are others symptoms such as nausea or vomiting, or chills or fever, where the costs in mistakenly alerting users equal or outweigh the benefits. Presence of symptoms or risk factors could help decide whether to apply early detection models for a searcher.

Symptom or Risk Factor	TPR at FPRs ranging from 0.00001–0.1					AUROC	# pos (%)	# neg (%)	FPR = 0.00001		
	0.00001	0.0001	0.001	0.01	0.1				Capture	Cost	Capture/Cost
Dark or tarry stool (S)	7.692	7.692	23.077	38.462	46.154	0.7173***	13 (1.2%)	58,597 (1.9%)	1	0.5860	1.7066
Abdominal swelling (S)	4.167	8.333	16.667	20.833	45.833	0.7735***	24 (2.2%)	45083 (1.5%)	1	0.4508	2.2183
Ulcers (RF)	0.000	0.000	0.000	7.895	50.000	0.7894***	38 (3.5%)	16,081 (0.5%)	0	0.1608	0.0000
Dark urine (S)	0.000	5.556	16.667	27.778	50.000	0.8129**	18 (1.7%)	51,236 (1.7%)	0	0.5124	0.0000
Pancreatitis (RF)	6.061	9.091	12.121	24.242	54.546	0.8220**	33 (3.1%)	34,184 (1.1%)	2	0.3418	5.8514
Abdominal pain (S)	5.385	10.000	16.923	32.308	60.000	0.8343**	130 (12.1%)	311,266 (10.3%)	7	3.1127	2.2489
Enlarged gallbladder (S)	0.885	2.655	9.735	25.664	53.982	0.8358**	113 (10.5%)	98,454 (3.3%)	1	0.9845	1.0157
Constipation (S)	3.529	7.059	9.412	22.353	57.647	0.8469**	85 (7.9%)	317,300 (10.5%)	3	3.1730	0.9455
Smoking (RF)	3.846	3.846	7.692	15.385	53.846	0.8585	26 (2.4%)	27,817 (0.9%)	1	0.2782	3.5945
Blood clot (S)	4.494	10.112	14.607	31.461	61.798	0.8589	89 (8.3%)	351,385 (11.6%)	4	3.5139	1.1383
High blood sugar (S)	6.135	8.896	16.564	31.595	60.429	0.8611	326 (30.4%)	429,543 (14.2%)	20	4.2954	4.6561
Nausea or vomiting (S)	3.200	8.800	17.600	30.400	63.200	0.8706	125 (11.7%)	639,502 (21.1%)	4	6.3950	0.6255
Chills or fever (RF)	3.636	7.273	20.909	30.909	65.455	0.8727	110 (10.3%)	357,536 (11.8%)	4	3.5754	1.1188
Loose stool (S)	4.615	7.692	18.462	35.385	72.308	0.8756	65 (6%)	74,720 (2.5%)	3	0.7472	4.0150
Indigestion (S)	7.547	12.264	20.755	38.679	68.868	0.8932	106 (9.9%)	504,462 (16.7%)	8	5.0446	1.5859
Itchy skin (S)	18.750	25.000	25.000	25.000	75.000	0.8982	16 (1.5%)	79,448 (2.6%)	3	0.7945	3.7760
Back pain (S)	7.801	14.184	19.858	34.752	69.504	0.9047	141 (13.2%)	223,586 (7.4%)	11	2.2359	4.9197
Yellow skin or eyes (S)	2.174	5.439	19.565	38.044	73.913	0.9217	92 (8.6%)	85,805 (2.8%)	2	0.8581	2.3307
Hepatitis (RF)	7.692	10.256	20.513	38.462	71.795	0.9275	39 (3.6%)	25,158 (0.8%)	3	0.2516	11.9237
Alcoholism (RF)	12.500	16.667	27.083	41.667	89.583	0.9494**	48 (4.5%)	32,333 (1.1%)	6	0.3233	18.5586
Obesity (RF)	20.690	20.690	37.931	62.069	82.7590	0.9572**	29 (2.7%)	22,153 (0.7%)	6	0.2215	27.0880
Overall	4.851	8.302	17.258	36.474	72.015	0.9003	1,072 (100%)	3,025,046 (100%)	52	30.2505	1.7190

Table 6.7: Performance of the models conditioned on a variety of symptom (S) and risk factors (RF). Values below the dashed line have a higher AUROC than *Overall*. Capture represents the number of TP cases in the cohort of positives $\cup$ negatives at FPR=0.00001. Cost is computed as the target FPR (.00001) multiplied by the size of the negative set in each subgroup. Since this exact FPR may not be attainable in each subgroup, cost may not be an integer. A capture-cost ratio of > 1.0 means that more people would benefit from an alert than would be mistakenly alerted. Statistically significant differences with *Overall* model (DeLong's test [DeLong et al., 1988]) are marked ** $p < 0.001$ and *** $p < 0.0001$ (where α following a Bonferroni correction is 0.002).

6.4.6 Deploying Learned Model in Practice

Up to now, our model considers experiential diagnostic users as positives and symptom-only users as negatives. This is a clean dataset for algorithm training and testing but it ignores the symptom searchers who issue non-experiential pancreatic cancer searches (gray region in Figure 6.1). These users may have been diagnosed or may simply be exploring. Regardless, they should be considered in practice.

We perform an additional experiment on a separate set of symptom searchers that included non-experiential pancreatic cancer searchers as negatives. We trained a model on all data described thus far and applied it to identify (i) experiential and (ii) experiential+treatment users in a new held out dataset in advance of their first experiential diagnostic query. We generated the test set from logs of a separate randomly selected subset of Bing users, over an 18-month period from August 2014 to January 2016 inclusive. There was no overlap in users with the set used for training. We identified positive cases as earlier and expanded the definition of negatives to include pancreatic cancer searchers. This resulted in 2.9 million negatives, including 48,221 non-experiential pancreatic cancer searchers, and 945 experiential searchers with preceding symptom searches. To help target the identification of cases where experiential queries are issued, we created a subset of the positives who issued treatment-related queries, such as *whipple procedure*, a surgical operation to remove tumours from pancreas, and *cluorouracil (5-FU)*, a medication to treat cancer, following Exp_0; in total, 494 users (52%) met this requirement. The symptom lookup durations for positives and negatives were similar to Section 6.2.5. We randomly split the test data into ten equally-sized subsets for significance testing. Table 6.8 reports the predictive performance at different lead times.

Table 6.8 shows that the performance of the model remains strong on this held-out set and is comparable to that reported in the earlier sections. The performance decreases with increased lead time as noted previously (see Table 6.4). Interestingly, the performance in identifying the subset of experiential diagnostic users who subsequently searched for treatments is higher than for the experiential-only set. This is promising confirmatory

Weeks before	Experiential		Experiential+Treatment	
Exp_0	AUROC	TPR (%)	AUROC	TPR (%)
1 week	0.9012	8.677	0.9225	12.145
5 weeks	0.8892	8.571	0.9089*	11.943
9 weeks	0.8754**	8.278*	0.8902**	11.343*
13 weeks	0.8611**	8.018**	0.8795**	10.738**
17 weeks	0.8456***	7.666**	0.8683***	10.400***
21 weeks	0.8330***	7.438***	0.8508***	9.645***

Table 6.8: Average AUROC and average TPR (as %) at FPR=0.00001 for identifying experiential users and experiential+treatment users from held out dataset. Differences in AUROC and TPR between $Exp_0 - 1$ week and other weeks noted (*** $p < 0.001$, ** $p < 0.001$, and * $p < 0.01$).

evidence as these users are assumed to have experienced a cancer diagnosis, per definitions of experiential queries [Paul et al., 2014].[5]

6.4.7 Limitations and Discussion

Despite the promising findings, we note several important limitations. First, we lack explicit ground truth about diagnoses per the anonymity of our logs. We rely on models of self-reporting in queries. We have found that streams of queries following the experiential queries can provide confirmatory evidence of pancreatic cancer diagnoses. Indeed, in the weeks immediately following the experiential diagnostic query, over 40% of searchers queried for treatment options, with many using sophisticated terminology (e.g., Whipple procedure, pancreaticoduodenectomy, neoadjuvant therapy) and over 20% of searchers searched for pancreatic cancer medications (e.g., gemcitabine, 5-fu). In contrast, only 0.5% and 0.02% of searchers in our negative set searched for treatments and medications, respectively, at any point in their query timeline.

We need to work with diagnosed patients to understand (i) the relationship between experiential searching and diagnosis; and (ii) the model performance with the use of traditional data as ground truth about diagnoses (e.g., medical records). We also need to understand the role of factors such as race [Coughlin et al., 2000], family history [Lynch et al., 1996],

[5]For completeness, we also trained a model on all data from earlier in the chapter, including the non-experiential pancreatic cancer searchers as negative cases, and tested it on this held-out set. The performance is around 5% lower than reported in Table 7, for both AUROC and TPR, across all weeks. Including the non-experiential pancreatic cancer searchers may add noise to model training.

medical histories [Lowenfels et al., 1993], diabetes [Everhart and Wright, 1995], and other factors (e.g., smoking [Fuchs et al., 1996]). Some of these can be crudely estimated from geographic and census data (race), whereas others (family and medical histories) are best sought from searchers directly.

To reflect anticipated performance in a natural setting, we focused on our imbalanced dataset. We re-ran the analysis with a balanced set, with highly similar results. Finally, we note that this is a retrospective analysis for model training and testing, and we need to consider its representativeness for real-time screening, e.g., identifying negative cases in the retrospective study relies on symptom lookup durations. Potential additional analyses include explorations of predicting on fixed dates versus at the end of the observation period.

6.5 Conclusions

In this chapter, we explored the promise of harnessing temporal patterns in behavioral signals extracted from web search engine logs for early detection of devastating diseases. Specifically, together with collaborators from Microsoft Research, we leveraged billions of queries from the Bing search engine and studied the feasibility of doing early detection of the presence of pancreatic cancer in experiential user queries. We summarize the contributions of this chapter as follows: (i) we introduced the early detection of diseases as a promising new application of search log mining and machine learning that scales to millions of searchers; (ii) we described the process of generating a large-scale real-world dataset to study the feasibility of early detecting experiential pancreatic cancer cases from longitudinal individual search activity (Section 6.2); (iii) we reviewed the large set of characteristics that we extract from query timelines (Section 6.3.1); (iv) we presented a simple methodology to track the temporal relationships and patterns in the content of queries over time (Section 6.3.2); (v) we summarized our prediction model to forecast with significant lead times that users will later input experiential queries for pancreatic cancer (Section 6.3.3); and (vi) we explored the influence of different factors, such as the lead time or the presence of specific symptoms in the search activity, on the predictive performance of our learned models, including true positive rates when false positive rates are strictly controlled (Section 6.4).

Our study provides insights on the feasibility of learning from search engine logs to predict future issuance of experiential queries about pancreatic cancer at a low error rate. The success of these methods has implications for online methods that would provide passive screening of searchers, to offer early warning about potential signs of pancreatic cancer and other devastating diseases. Our evaluation suggests that the combination of static features extracted from query timelines with our temporal features yielded the greatest impact than any other feature category considered in our analysis. Additionally, our analysis revealed a number of important domain-specific findings. Specifically, we discovered that conditioning on different symptoms and risk factors can enhance predictive power. We found capture-cost tradeoffs associated with different symptoms and risk factors in terms of the total number of truly positive cases identified versus the number of searchers who would be mistakenly alerted. We characterized model performance as we increase lead time and we found that we can attain a TPR in region of 5–30%, while controlling the FPR to 0.00001–0.01 months before a diagnostic query is observed. Looking forward, we seek to understand the costs and clinical significance of these methods, including how they might offer early warning of devastating disease onset to enhance outcomes (e.g., quality of life).

Chapter 7

Related Work

In Chapter 2, we provided a thorough overview of data mining approaches for time-series analysis and presented necessary related work. This chapter now reviews additional literature that is relevant to each chapter of this dissertation. Specifically, Section 7.1 outlines related work on time-series clustering, the problem that we addressed in Chapter 3. Section 7.2 summarizes related work on unsupervised feature learning for time series, the problem that we addressed in Chapter 4. Section 7.3 explores related work on methods that measure and predict impact in the scientific literature, which is the problem that we addressed in Chapter 5. Finally, Section 7.4 covers related research on large-scale analyses of web search behavior and on methods for early detection of diseases, with emphasis on pancreatic cancer, which is the problem that we addressed in Chapter 6.

7.1 Efficient and Accurate Time-Series Clustering

Chapter 3 focused on efficient, accurate, and domain-independent time-series clustering. Section 3.2 provided an in-depth discussion of the state of the art for time-series clustering, which we will not repeat for brevity. As argued throughout the chapter, we focused on shape-based clustering of time series. Beyond shape-based clustering algorithms, statistical-based approaches use measures that summarize characteristics [Wang et al., 2006], coefficients of models [Kalpakis et al., 2001], or portions of time series (i.e., shapelets) [Zakaria et al., 2012] as descriptive features to cluster time series. Unfortunately, statistical-based approaches frequently require the non-trivial tuning of multiple parameters, which often leads to ad-hoc decisions, or their effectiveness has been established only for isolated domains and not in a domain-independent manner. Instead, shape-based approaches are general and leverage the vast literature on distance measures. In Section 3.6.6, we showed that both k-Shape and k-AVG+ED significantly outperform the U-Shapelets method [Zakaria et al., 2012].

The best performing shape-based approaches from the literature are partitional methods combined with scale- and shift-invariant distance measures. Among partitional methods, k-medoids [Kaufman and Rousseeuw, 2009] is the most popular method as it enables the easy adoption of any shape-based distance measure [Ding et al., 2008; Wang et al., 2013]. However, k-medoids requires the computation of the full dissimilarity matrix among all time series, which makes it particularly slow and unable to scale. Recent alternative approaches [Gupta et al., 1996; Meesrikamolkul et al., 2012; Niennattrakul and Ratanamahatana, 2009; Petitjean et al., 2011; Yang and Leskovec, 2011] have focused on k-means [MacQueen, 1967], which is scalable but requires the modification of the centroid computation method when the distance measure is altered, in order to support the same properties (e.g., scaling, translation, and shifting). Because DTW is the most prominent shape-based distance measure [Ding et al., 2008; Wang et al., 2013], the majority of the k-means approaches have proposed new centroid computation methods to be used in conjunction with DTW [Gupta et al., 1996; Meesrikamolkul et al., 2012; Niennattrakul and Ratanamahatana, 2009; Petitjean et al., 2011]. k-DBA has been shown to be the most robust of these approaches [Petitjean et al., 2011]. Another approach worth mentioning is KSC [Yang and Leskovec, 2011], which focuses on a different shape-based distance measure that offers simultaneously pairwise scaling and shifting of time series. Unfortunately, the effectiveness of such pairwise scaling and shifting, and, hence, KSC, has not been established beyond a limited number of datasets. In Sections 3.6.3 and 3.6.4, we evaluated k-Shape against shape-based approaches, including combinations of k-means and k-medoids with the most competitive distance measures, as well as k-DBA and KSC. Apart from shape-based approaches, we also compared k-Shape against hierarchical and spectral methods (see Section 3.6.4), density-based methods (see Section 3.6.5), and shapelet-based methods (see Section 3.6.6).

For completeness, we note that [Golay et al., 1998] used the arithmetic mean property for centroid computation and cross-correlation in distance measures with parameters to regulate the fuzziness of fuzzy clustering of fMRI data. (In Section 3.6, we showed that k-AVG+SBD is not competitive for our non-fuzzy setting.) [Goutte et al., 1999] used cross-correlation to transform fMRI data into features for clustering. [Baragona, 2000] evaluated several weighting schemes of cross-correlation along with a genetic algorithm to avoid inappropriate initialization for the k-min clustering criterion [Sahni and Gonzalez, 1976]. [Hőppner and Klawonn, 2009] proposed a fuzzy c-means [Bezdek, 2013] clustering method that uses a weighting scheme to penalize alignment of time series in larger lags when cross-correlation is used, a weighted arithmetic mean for centroid computation, and a mechanism to handle outliers given a user-defined threshold. Finally, cross-correlation was used to speed up subsequence matching [Mueen et al., 2014] and the computation of shapelets [Mueen et al., 2011], as well as for stream mining, pattern extraction, and time-series monitoring [Papadimitriou et al., 2006; Sakurai et al., 2005a; Wu et al., 2010; Zhu and Shasha, 2002].

7.2 Efficient Time-Series Representation Learning

Chapter 4 focused on efficient feature representation learning for time series. Section 4.2 provided an in-depth discussion of the state of the art for time-series dimensionality reduction and representation methods, kernel methods, and approximation mechanisms to accelerate kernel methods. As argued throughout this chapter, we focused on and reviewed unsupervised approaches to learn low-dimensional feature representations. Beyond these approaches, traditional methods assume a model, which is often expressed in the form of analytical equations with parameters, to describe time series and use the estimated parameters of such model as features in time-series mining tasks [Längkvist et al., 2014]. There is a plethora of model-based approaches in the literature [Hyndman and Athanasopoulos, 2014; Box et al., 2015; Fulcher, 2017] that rely on different models to serve different application needs. The most prominent approaches favor models based on Hidden Markov Models [Rabiner and Juang, 1986], Gaussian Process models [Brahim-Belhouari and Bermak, 2004], autoregressive models [Lütkepohl, 2005], such as autoregressive integrated moving average (ARIMA) models and generalized autoregressive conditional heteroscedastic (GARCH) models, exponential smoothing models [Gardner, 2006], and models from Linear Dynamic Systems [Luenberger, 1979]. Unfortunately, unrealistic assumptions of such models, combined with their limited power to model highly complex, noisy, and high-dimensional time series with analytical equations, impact the effectiveness of model-based approaches as standalone feature extraction methods for real-world problems [Längkvist et al., 2014].

As a consequence, there have been significant efforts in the time-series literature to extract generic features to represent time series using combinations of statistical measures that summarize different time-series properties, including their distribution, correlation structure, stationarity, entropy, and fitting to a range of different time-series models. For example, initial efforts focused on a small number of statistical measures to extract features from time series for subsequent analysis [Nanopoulos et al., 2001; Wang et al., 2006; Deng et al., 2013]. Recent notable efforts collected more than $1,000$ conceptually different methods to extract more than $9,000$ features [Fulcher et al., 2013; Fulcher and Jones, 2014] proposed across scientific disciplines, which indicates the difficulty to learn robust feature representations for time series. Despite the effectiveness of such methods for classification and forecasting tasks where the supervised selection of features reduces the dimensionality of feature vectors and increases accuracy, these approaches are not competitive for unsupervised tasks.

Similarly, many time-series classification methods involve operations for feature extraction. For example, dictionary-based classifiers use a sliding window to first extract subsequences from time series and, then, to map such subsequences into a set of patterns with the goal to transform time series into distributions of patterns [Schäfer, 2016; Kate, 2016; Lin et al., 2012]. Shapelet-based classifiers extract subsequences from time series that are discriminatory of their class membership [Grabocka et al., 2014; Hills et al., 2014]. Finally, interval-based classifiers extract features from intervals of time series [Baydogan et al., 2013; Deng et al., 2013]. As before, such methods select features in a supervised manner to improve classification accuracy [Bagnall et al., 2017]. Unfortunately, such methods are not competitive for unsupervised settings, as we have shown for clustering methods relying in unsupervised selection of shapelets in Chapter 3.

Alternative approaches to learn feature representations in an unsupervised manner rely on neural networks [Zhang et al., 1998] and deep learning methods [Bengio et al., 2007b; Bengio et al., 2007a; Bengio et al., 2012]. An advantage of these methods in comparison to our approach is that they can learn multiple layers of feature representations to facilitate the modeling of complex structures in time series. For example, Restricted Boltzmann Machines (RBM) [Hinton et al., 2006; Hinton and Salakhutdinov, 2006] learn a weight matrix between visible observations and layers of hidden units. To model time series, Temporal RBM [Sutskever and Hinton, 2007] follow a methodology similar to the autoregressive models discussed above and connect current hidden units with previously visible units. Auto-encoder methods learn weight matrices between the visible observations, the hidden units, and the reconstructions of visible units [Bengio et al., 2007a; Bengio et al., 2009]. Recurrent neural network (RNN) [Hüsken and Stagge, 2003] model the time dependencies by connecting the hidden layers of each visible observation. Finally, convolutional RBM (convRBM) [Lee et al., 2009a; Lee et al., 2009b] and auto-encoders (convAE) [Masci et al., 2011] model time series by dividing the input visible observations into segments that are only partially connected to hidden units. (We refer the reader to [Längkvist et al., 2014] for a thorough review of neural networks and deep learning approaches for time series.) Recent efforts relying on deep learning methods [Cui et al., 2016; Wang et al., 2017; Fiterau et al., 2017] show competitive performance in terms of classification accuracy to state-of-the-art time-series classifiers [Bagnall et al., 2017] but without substantial improvements in terms of efficiency. As before, such methods have yet to prove competitive results in unsupervised settings like those considered in our work. In contrast, our approach to learn feature representations achieves remarkable accuracy and scalability for both supervised and unsupervised tasks.

7.3 An End-to-End System for Predicting the Impact of Scientific Concepts

Chapter 5 explored the promise of tracking features over time for improving prediction accuracy of the prominence of scientific concepts in the future. In this section, we cover additional relevant work.

Studying science is a science in and of itself. For example, the National Science Foundation has two programs designed to fund this type of research: Science, Technology, and Society (STS), which is primarily oriented toward qualitative research, and Science of Science and Innovation Policy (SciSIP), which is primarily oriented toward quantitative research. STS as a field of study has a long history. As the name indicates, STS utilizes social science and humanities approaches to understand the relationships among science, technology, and society. There is a wide range of STS approaches. For example, laboratory ethnography [Knorr-Cetina, 1999; Traweek, 1992] involves extended fieldwork within science and technology settings, observing and interviewing scientists and engineers in their native habitats. Actor-network theory [Latour, 1988] involves tracing the relationships among human actors and non-human actants. As such, technologies are seen as having some agency, or ability to shape the world. Another approach commonly used within the domain of science and technology policy is an expert panel, such as the Delphi method [Bornmann and Daniel, 2008]. Such qualitative approaches are useful for learning about specific labs or subfields in rich detail, however they are not typically scalable. Thus, to automatically track scientific innovation in real time, quantative approaches are far more appropriate.

Scientometrics, or the measurement of science, has long been used to understand science at the macro scale as well as to make policy recommendations [Edge, 1979; Bornmann and Daniel, 2009; Schreiber, 2013]. Since ranking algorithms based on scientometric data have demonstrated real potential to influence the direction of scientific progress [Beel and Gipp, 2009], it is important for these algorithms to take into account as much information as possible to inform the resource allocation decisions of nations, institutions, and individual researchers [Lane and Bertuzzi, 2011; Lane, 2010].

Study of scientific impact spans almost a century, during which time expanding data sets and sophisticated tools have allowed for increasingly powerful results. Following several decades of small, expensive studies conducted for journal evaluation and acquisition, major citation indexing projects enabled the application of quantitative methods to the problems of research evaluation [Narin, 1976] and scientific prestige [Cole and Cole, 1967; Bayer and Folger, 1966]. Since then, metrics such as the Journal Impact Factor [Garfield, 2006], which is primarily used to evaluate the impact of a journal, and, more recently, the h-Index [Hirsch, 2005], which is primarily used to evaluate the impact of a scientist, have been used. Scientometrics builds in part on the type of qualitative research described above, such as study of the function of citation [Moravcsik and Murugesan, 1975; Chubin and Moitra, 1975; Spiegel-Rösing, 1977] or the motivations for citation, often bringing these to bear in critique of the use of citations in research evaluation [Bornmann and Daniel, 2008]. For example, citation counts include not only works that build on previous work but also works that negate the previous work or cite it perfunctorily [Ziman, 1968; Bonzi, 1982]. Together these research streams comprise a large part of the quantitative science of science within the social sciences. Machine learning has introduced new horizons in the study of science [Losiewicz et al., 2000] that continue to expand with increasing computational power and the availability of full text databases [Arbesman and Christakis, 2011].

An early paper speculated on the relationship between citation data and future author performance [Garfield and Malin, 1968], and a few recent studies have attempted to predict future citations received by an author based on features of past work. These include studies of the predictive value of the h-index, which have played a role in the debates over that metric [Hirsch, 2007; Hönekopp and Khan, 2012], as well as attempts to predict changes in an author's h-index over time [Acuna et al., 2012; Penner et al., 2013; Dong et al., 2015]. [Zhu et al., 2015] present a variant of h-index called the hip-index (influence primed h-index) based on datasets of papers and references that were influential for a paper and use it to predict fellows of an organization. All of these studies have tended to use simple feature sets, most often including citation-based indicators of past performance, although social factors [Laurance et al., 2013], social network properties [McCarty et al., 2013; Sarigöl et al., 2014], and structural variation models representing impact on state of the art [Chen, 2012] have also been examined. Others [Ding et al., 2009] have experimented with weighted Pagerank algorithms to rank authors in author co-citation networks and a HITS framework [Wang et al., 2014b] for simultaneous ranking of future impact of papers and authors.

Network-based approaches, building on research in social network analysis, have proven effective in helping to understand the structure of science [Birnholtz et al., 2013; Velden et al., 2010; Velden and Lagoze, 2013]. While our research builds on these approaches, the goal of Chapter 5 is to go beyond the typical network-based analyses that focus on nodes and edges and instead consider the content of the edges via natural language processing of full text. Importantly, our work goes one step further and captures the temporal variation of characteristics produced by network-based and text-based approaches.

Previous work has applied bibliometrics at the level of entities discovered in full text [Ding et al., 2013] as well as based on productivity, collaboration and influence [Havemann and Larsen, 2015]. Additionally, topic proportions from Latent Dirichlet Allocation have been used to study the history of scientific ideas [Hall et al., 2008]. To the best of our knowledge, our work is the first to predict term frequencies as proxies for the emergence of scientific concepts, and is novel in the sophistication of the full text features we bring to bear on the problem. While previous work that has used full text in prediction has relied on bags of words (e.g., [Yogatama et al., 2011; Yan et al., 2011; Boyack et al., 2011]), we base some of our analysis on larger units of texts (phrases), on more linguistically motivated features such as rhetorical analysis, and on the temporal patterns and relations among such features.

Recently, there has been more work on the analysis of scientific articles that could ultimately be helpful for prediction of scientific impact [Tsai et al., 2013; Louis and Nenkova, 2013; Tan and Lee, 2014]. For example, the 2003 KDD Cup [Gehrke et al., 2003] included a citation prediction track. Since then, approaches to prediction have matured, and despite varying research designs, several classes of predictive variables have been established. These include citation data [Manjunatha et al., 2003], journal characteristics [Callaham et al., 2002; Lokker et al., 2008; Kulkarni et al., 2007], author characteristics [Castillo et al., 2007], n-gram features drawn from abstracts and index terms [Ibáñez et al., 2009; Fu and Aliferis, 2008], download statistics [Brody et al., 2006], and social media mentions [Eysenbach, 2011]. Fu and Aliferis unified much of the early work in this area, reporting evidence that author metrics improved the scores obtained by modeling journal characteristics alone, and that adding metadata features improved scores still further [Fu and Aliferis, 2008]. More recent natural language processing research has yielded mixed results on n-gram and topic features drawn from full text [Yogatama et al., 2011; Yan et al., 2011], and the usefulness of full text in citation prediction for papers remains an open question.

Much research has focused on particular disciplines and sub-disciplines of science. For example, scientometric approaches have been applied to computer science [Guha et al., 2013] as well as its subfields, such as human-computer interaction [Bartneck and Hu, 2009] and computer-supported cooperative work [Horn et al., 2004]. Since the goal of Chapter 5 is to help predict innovation across various fields of science and engineering, we build on this earlier work, but cannot rely solely upon metrics that have proven to be effective within any one field.

7.4 Detecting Devastating Diseases in Search Logs

Chapter 6 explored the promise of harnessing temporal patterns in behavioral signals extracted from web search logs to early detect devastating diseases. Related research in a number of areas are relevant to our work. Specifically, we review relevant work for: (i) health-related web searching; (ii) large-scale analyses of search behavior; and (iii) methods for the early detection of diseases, with a focus on pancreatic cancer.

The web is an important source of health-related information for many people. To better understand how people pursue health information, studies have examined online health search using a variety of methods, including interviews [Peterson et al., 2003], surveys [Trotter and Morgan, 2008], and analyses of large-scale search log data [Ayers and Kronenfeld, 2007; Bernstam et al., 2009]. According to a 2013 survey, 59% of American adults had used the Web to find health information in the year preceding the survey, 35% of those adults engaged in self-diagnosis, and over half of these self-diagnosing searchers then discussed the matter with a clinician following the search [Fox and Duggan, 2013]. Despite the potential benefits, concerns have been raised about the quality of online health information [Cline and Haynes, 2001]. In a large-scale survey of the use of search for self-diagnosis, White and Horvitz [White and Horvitz, 2009] found that almost 40% of participants experienced increased anxiety from searching health information online. Studies have characterized problems with symptom search, including the influence of poor accounting for base rates of diseases and people's bias to focus on results covering serious illnesses versus more likely benign explanations. Such biases can lead to inappropriate anxiety [Lauckner and Hsieh, 2013; White and Horvitz, 2009] and highlight the criticality of studying how patients use the Web, including the nature and dynamics of queries, and content delivered in response.

There has been a large amount of research on the analysis of search behavior from search engine logs. Log analysis provides insights to understand how people engage in information seeking in online settings [White and Drucker, 2007], while also having applications for tasks such as result ranking [Agichtein et al., 2006; Joachims, 2002], query suggestion [Jones et al., 2006], prediction of future search actions and interests [Downey et al., 2007; Lau and Horvitz, 1999], and detection of real-world events and activities [Richardson, 2009]. Given access to population-scale data on how people search for health information, this can be applied for important tasks such as the detection of influenza [Ginsberg et al., 2009], the detection of adverse drug reactions [White et al., 2013], population-scale studies of nutrition [West et al., 2013], epidemiology [Ginsberg et al., 2009], and studies of chronic medical conditions such as pregnancy [Fourney et al., 2015]. Related to this research, but focused on activity post-diagnosis, are studies of cancer-related searching [Bader and Theofanos, 2003; Castleton et al., 2011; Helft, 2011], some of which have revealed strong similarities between temporal patterns in search logs and those in practice [Ofran et al., 2012; Paul et al., 2014]. Studies have leveraged online behavioral signals for early disease detection at the population level [Ginsberg et al., 2009], and individually [De Choudhury et al., 2014; Sadilek et al., 2012].

Screening high-risk individuals for pancreatic cancer is the only practical approach to detect precancerous or cancerous changes in the pancreas at the phase in which surgical intervention will have a high chance of cure [Klapman and Malafa, 2008]. Risk level can be determined by factors such as race [Coughlin et al., 2000], family history [Lynch et al., 1996], and a history of pancreatitis [Lowenfels et al., 1993]. Imaging studies via methods such as endoscopic ultrasound, computer tomography scans, and magnetic resonance imaging [Mertz et al., 2000; **?**] have been useful to diagnose pancreatic cancer once the tumor is large enough to cause unusual, salient symptoms that induce people to seek medical attention (e.g., yellow eyes, changes in stool), but at this point the disease is more likely to be at an advanced and unresectable stage (i.e., locally advanced or metastatic, when it cannot be removed by surgery) [Legmann et al., 1998]. Common, seemingly innocuous symptoms such as back pain, abdominal pain, itchy skin, unexplained weight loss and nausea (and combinations and temporal patterns of these and other symptoms) may also be observed in the query stream. Such symptom searches can provide patterns of symptoms that might one day be employed in new kinds of health surveillance systems. Such systems could be used to alert people who would otherwise not feel moved to see a healthcare professional.

Active, explicit screening for early signs of pancreatic cancer is not cost effective unless there is a reasonable probability of detecting invasive or pre-invasive disease (at least 16% according to one study [Rulyak et al., 2003]). A log-based methodology provides scale that is not achievable with more traditional epidemiological studies, which tend to be on the order of tens or hundreds of participants [Huxley et al., 2005; Renehan et al., 2008].

Chapter 8

Conclusions

In this dissertation, we studied and developed computational methods for efficient unsupervised learning of robust feature representations from time series. We focused on two types of applications: (i) applications that operate solely over time-series collections; and (ii) applications where the analysis of time series complements the analysis of other types of data. Next, we summarize our primary contributions:

- **Efficient and Accurate Time-Series Clustering:** In Chapter 3, we addressed the problem of domain-independent, accurate, and scalable clustering of time series. We characterized the drawbacks of existing methods for this task and we introduced two novel time-series clustering methods. Specifically, we showed how to derive a distance measure from cross-correlation that offers important invariances to time-series distortions in a principled manner. We also showed how to efficiently compute this measure. We presented two novel methods to compute a single centroid or multiple centroids per cluster when that distance measure is used. We developed two novel algorithms for time-series clustering that rely on the aforementioned distance measure: one algorithm computes a single centroid per cluster based on the entire set of time series in each cluster, whereas the other (outlier-aware) algorithm computes multiple centroids per cluster in order to consider the proximity and spatial distribution of time series in each cluster. Then, we presented a one-nearest-neighbor classification algorithm that relies on time-series clustering as a subroutine to reduce the search space and accelerate one-nearest-neighbor algorithms. Finally, we performed an extensive experimental evaluation of our methods against state-of-the-art time-series distance measures, clustering algorithms, and classification algorithms. Our findings showed that our efficient and parameter-free distance measure leads to similar results to the most accurate but computationally expensive distance measures that require parameter tuning. In addition, our clustering methods outperform all state-of-the-art scalable and non-scalable partitional, hierarchical, spectral, density-based, and shapelet-based clustering approaches, with only one method achieving similar performance, but this method requires tuning of parameters and cannot scale to large volumes of data. Finally, our evaluation suggested that we can rely on time-series clustering methods as subroutines to effectively reduce the search space for one-nearest-neighbor time-series classification algorithms. Consequently, we believe that our methods emerge as domain-independent, accurate, and scalable approaches for time-series comparison, clustering, and classification.

- **Efficient Time-Series Representation Learning:** In Chapter 4, we introduced and addressed the problem of efficiently learning data-aware, low-dimensional representations of time series that preserve the invariances offered by a given similarity function. Our approach represents a departure from classic approaches in the time-series literature where representation methods are agnostic to the similarity function used in subsequent learning processes. Our objective was to simplify and unify the design of scalable time-series mining methods. To achieve this ambitious goal, we relied on kernel methods. Specifically, kernel methods interact with data through pairwise comparisons using a kernel function and, therefore, require constructing a data-to-data similarity matrix. To alleviate the burdens associated with the high memory and runtime costs of (i) computing the data-to-data matrix and (ii) performing dimensionality reduction using such matrix, we proposed GRAIL, a generic representation learning framework. GRAIL uses the Nyström method to reduce the data-to-data matrix to a data-to-landmark points matrix and relies on matrix sketching to approximate the eigendecomposition of the data-to-data matrix and, therefore, learn low-dimensional representations that preserve invariances offered by a user-specified kernel function. The effectiveness of GRAIL critically relies on (i) the choice of kernel function; (ii) the choice of landmark time series; and (iii) the unsupervised estimation of parameters that affects the compactness and the reconstruction quality of the learned representations. We addressed these issues as follows: (i) we developed an efficient kernel function to compare time series under important invariances for time-series distortions; (ii) we constructed landmark time series using the effective time-series clustering methods from Chapter 3; (iii) we presented a method to estimate kernel function parameters and compactness of representation; (iv) we described how to efficiently learn GRAIL representations by approximating the eigendecomposition of the data-to-data matrix using matrix sketching; (v) we showed how GRAIL representations accelerate existing methods for major time-series mining tasks, such as querying and indexing, clustering, classification, sampling, and visualization; and (vi) we evaluated our ideas by conducting an extensive experimental evaluation. Our findings showed that kernel classifiers using our kernel function significantly outperform one-nearest-neighbor classifiers with state-of-the-art distance measures, which debunks a long-standing perception in the time-series literature that such classifiers are difficult to beat. GRAIL representations are more compact and have significantly better pruning power than current time-series representations. Finally, GRAIL representations, combined with kernel methods, significantly outperform, in terms of efficiency and accuracy, state-of-the-art methods that operate over the full length of time series in all aforementioned tasks. We believe that GRAIL rises as a new primitive for highly accurate, yet scalable, time-series analysis that significantly simplifies the design of new algorithms while requiring minimal user input.

- **An End-to-End System for Predicting the Impact of Scientific Concepts:** In Chapter 5, we studied the problem of predicting the future impact of scientific concepts and described an end-to-end system developed for this task by a team of researchers at Columbia and several other universities. Our objective was to (i) develop generic, simple, and lightweight methodologies to effectively extract features from time-varying measurements that seamlessly integrate with techniques developed for other types of data; and (ii) demonstrate the impact of exploiting methods from time-series analysis to significantly improve the accuracy of prediction models used in such large-scale, real-world application. Specifically, we reviewed the challenges of this ambitious, yet important, problem of identifying research concepts that hold the most promise as early as possible. We described the architecture of the system we built to facilitate the extraction of characteristics from articles in the scientific literature and real-time prediction of the future impact of scientific concepts. We outlined the large number of characteristics that our system extracts from the metadata and the full text of scientific articles. Then, we described a variety of principled statistical measures capable of capturing patterns in short, sparse, and noisy time-varying characteristics extracted from scientific articles. Finally, we performed a large-scale experimental evaluation. Our results showed the clear benefit of full text features over metadata features and that temporal features contribute the most in predictions. Across all our experiments, combinations of full text features and metadata features with time series relevant features yielded the greatest impact than any other indicator considered in our analysis. We believe such simple methodology is well-suited for applications where underlying operations produce thousands of user-defined characteristics that we need to track over time and transform into actionable knowledge.

- **Detecting Devastating Diseases in Search Logs:** In Chapter 6, we explored the promise of harnessing temporal patterns in behavioral signals extracted from web search engine logs for early detection of devastating diseases. Together with collaborators from Microsoft Research, we leveraged billions of queries from the Bing search engine and studied the feasibility of doing early detection of the presence of pancreatic cancer in experiential user queries. We cast this as a binary classification task, where the model is trained on features extracted from search log query timelines of experiential pancreatic cancer searchers and symptom-only searchers. We relied on a similar methodology to the one we developed in Chapter 5 to extract features from short, sparse, and noisy time series that often appear in the analysis of web search engine logs. We summarize the contributions of this chapter as follows: (i) we introduced the early detection of diseases as a promising new application of search log mining and machine learning that scales to millions of searchers; (ii) we described the process of generating a large-scale real-world dataset to study the feasibility of early detecting experiential pancreatic cancer cases from longitudinal individual search activity; (iii) we reviewed the large set of characteristics that we extract from query timelines; (iv) we presented a simple methodology to track the temporal relationships and patterns in the content of queries over time; (v) we described our prediction model to forecast with significant lead times that users will later input experiential queries for pancreatic cancer; (vi) we explored the influence of different factors, such as the lead time or the presence of specific symptoms in the search activity, on the predictive performance of our learned models, including true positive rates when false positive rates are strictly controlled. Our study provided insights on the feasibility of learning from search engine logs to predict future issuance of experiential queries about pancreatic cancer at a low error rate. The success of these methods has implications for online methods that would provide passive screening of searchers, to offer early warning about potential signs of pancreatic cancer and other devastating diseases. Our evaluation suggested that the combination of static features extracted from query timelines with our temporal features yielded the greatest impact than any other feature category considered in our analysis. Additionally, our analysis revealed a number of important domain-specific findings. For example, we discovered that conditioning on different symptoms and risk factors can enhance predictive power. We found capture-cost tradeoffs associated with different symptoms and risk factors in terms of the total number of truly positive cases identified versus the number of searchers who would be mistakenly alerted. We characterized model performance as we increase lead time and we found that we can attain a true positive rate of 5-30% while controlling the false positive rate to 0.00001-0.01 many months before a diagnostic query is observed. We believe that our simple methodology to reason about the evolution of time-varying characteristics in such important, large-scale, and real-world application, along with the interesting domain-specific findings that our work revealed, will motivate new studies across different scientific disciplines and industrial settings.

In summary, in this dissertation we investigated the design of computational methods to automatically learn effective feature representations from time series. Our objective was to simplify and unify the design of scalable and accurate time-series mining algorithms by constructing time-series representations that require minimal input from scientists or practitioners working over time-series collections. Specifically, we advocated that given a method to compare or model time series, the extraction of compact feature representations in time series can be fully automated. Importantly, such learned feature representations seamlessly integrate with and, therefore, accelerate existing methods to perform major time-series mining tasks. We hope that the contributions of this thesis will prove useful to the research community and result in the development of faster, more scalable, and more accurate algorithms for time-series analysis than those presented in this thesis.

Chapter 9

Future Work

Despite the significant progress made in this dissertation, there are still many open and exciting challenges in large-scale time-series analysis. Below, we focus on promising directions to address two important and largely unexplored challenges in time-series analysis, which are immediate extensions of our work in this dissertation.

- **Online Clustering of Massive Time-Series Collections:** In Chapter 3, we addressed the problem of domain-independent, accurate, and scalable clustering of time series. Specifically, we developed k-Shape and k-MS, two novel algorithms for time-series clustering that rely on a scalable iterative refinement procedure similar to the one used by the cornerstone k-means algorithm, but with significant differences. k-Shape and k-MS rely on the SBD distance measure to compare time series; k-Shape relies on the SE method to compute a single centroid per cluster based on the entire set of time series in each cluster, whereas k-MS relies on MSE to compute multiple centroids per cluster in order to consider the proximity and spatial distribution of time series in each cluster. We showed that k-Shape and k-MS scale linearly to the number of time series and through extensive experimental evaluation we demonstrated significant efficiency gains for our approaches in contrast to rival methods. However, our algorithms currently operate in the standard offline setting where the set of time series is known in advance and there is no restriction in the number of passes the algorithms can perform over the data. The high dimensionality of time series, combined with the rapid growth of time-series collections, poses great challenges to cluster time series with high performance and facilitate interactive exploration of time series in practice. An interesting direction for future work would be to modify our algorithms to operate in online settings, where algorithms often perform one pass over the data and retain only small amounts of information. To facilitate time-series clustering under such scenario, we need to develop new algorithmic ideas to address challenges associated with the high dimensionality and the large volume of time series. To address challenges associated with the high dimensionality of time series, a promising direction is to rely on a time-series representation method to reduce the dimensionality of time series and perform all operations, including time-series comparison and centroid computation, under the new low-dimensional representation. Considering that SBD relies on Fourier transform for its efficient computation, one interesting direction is to represent time series using only the leading Fourier coefficients. Several approaches in the literature explore similar ideas to approximate distance measures relying on correlations or lag correlations [Li et al., 1996; Zhu and Shasha, 2002; Sakurai et al., 2005a; Mueen et al., 2010] for pairs of time series. However, the computation of cluster centroids under such low-dimensional representations requires to approximate the entire dataset and not just pairs of time series, a problem that currently remains largely unexplored. To address challenges associated with the large volume of time series, we can rely on several paradigms developed for k-means-like algorithms to operate in online settings [Bottou and Bengio, 1995; Guha et al., 2003; Ailon et al., 2009; Sculley, 2010; Shindler et al., 2011]. However, the majority of such paradigms assume efficient methods to update cluster centroids when new data arrives. SE and MSE, our approaches to compute cluster centroids, rely on expensive eigendecomposition methods to extract centroids and, therefore, it becomes prohibitively expensive to re-calculate centroids for every new time series that arrives. A promising direction to resolve this issue is to rely on the methodology of matrix sketching [Liberty, 2013] that enables eigendecomposition of matrices in online settings.

- **Learning Alternative Time-Series Representations:** In Chapter 4, we introduced and addressed the problem of efficiently learning data-aware, low-dimensional representations of time series that preserve the invariances offered by a given similarity function. To achieve this ambitious goal, we relied on kernel methods and proposed GRAIL, a generic representation learning framework. GRAIL uses the Nyström method to reduce the data-to-data matrix to a data-to-landmark points matrix and relies on matrix sketching [Liberty, 2013] to approximate the eigendecomposition of the data-to-data matrix. Through these two steps, GRAIL learns low-dimensional representations that preserve invariances offered by a user-specified kernel function. The dimensionality of the representations learned through Nyström corresponds to the number of landmark time series extracted for the entire dataset and, therefore, is significantly smaller than the size of the entire dataset. Unfortunately, for very large datasets, Nyström might require thousands of landmark time series to accurately approximate the data-to-data matrix. In such cases, it is very likely that the dimensionality of the learned representation surpasses the dimensionality of the original time series, which defies the original purpose of producing low-dimensional representations. To eliminate this issue, we proposed to use matrix sketching to estimate the eigendecomposition of the entire data-to-data matrix and, therefore, we reduce even more the dimensionality of the representations by retaining only the top eigenvalues and eigenvectors of the data-to-data matrix. However, depending on the dataset, this step might impact the approximation quality of the new representation. A promising direction is to rely on sparse coding mechanisms [Olshausen and Field, 1997; Mallat, 1999; Lee et al., 2007; Mairal et al., 2009] to learn sparse representations for time series instead of the compact representations that we currently learn with GRAIL. Specifically, given a set of atoms that compose a dictionary, sparse coding aims to represent data as a linear combination of only *a few* basic atoms. The difference to Nyström is that sparse coding enforces a sparsity constrain in the learning of representations to avoid construction of representations that express time series as a linear combination of *all* atoms in a dictionary. Having shown that k-Shape can serve as a generic method for dictionary learning from time series, we can rely on the building blocks of SINK and WINK kernels that we used for GRAIL, to develop sparse time-series approximations. An alternative approach is to learn low-dimensional compact representations using neural networks [Zhang et al., 1998] and deep learning methods [Bengio et al., 2007b; Bengio et al., 2007a; Bengio et al., 2012]. These approaches construct multiple layers of representations to facilitate the modeling of complex structures in data. In contrast to kernel methods that require a user-defined kernel function to assess the similarity between pairs of time series, neural methods require the selection of the network architecture that better captures the desired properties in data. For example, Restricted Boltzmann Machines [Hinton et al., 2006; Hinton and Salakhutdinov, 2006], Recurrent Neural Networks [Hüsken and Stagge, 2003], and convolutional Restricted Boltzmann Machines [Lee et al., 2009a; Lee et al., 2009b] are some classes of neural methods that are suitable to model time series. For applications where large volumes of time series are available, we foresee a great promise in using neural methods to learn time-series representations.

This dissertation develops several methods, tools, and building blocks to help us address these open problems in large-scale time-series analysis, as we discussed above.

Bibliography

[Acuna et al., 2012] Acuna, D., Allesina, S., and Kording, K. (2012). Future impact: Predicting scientific success. *Nature*, 489:201–202.

[Affandi et al., 2013] Affandi, R., Kulesza, A., Fox, E., and Taskar, B. (2013). Nystrom approximation for large-scale determinantal processes. In *AISTATS*, pages 85–98.

[Agichtein et al., 2006] Agichtein, E., Brill, E., and Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *SIGIR*, pages 19–26.

[Agrawal et al., 1993] Agrawal, R., Faloutsos, C., and Swami, A. (1993). Efficient similarity search in sequence databases. In *FODO*, pages 69–84.

[Ailon et al., 2009] Ailon, N., Jaiswal, R., and Monteleoni, C. (2009). Streaming k-means approximation. In *NIPS*, pages 10–18.

[Aizerman et al., 1964] Aizerman, A., Braverman, E., and Rozoner, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

[Akaike, 1974] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723.

[Alam et al., 2015] Alam, S., Albareti, F., Prieto, C., Anders, F., Anderson, S., Anderton, T., Andrews, B., Armengaud, E., Aubourg, É., Bailey, S., et al. (2015). The eleventh and twelfth data releases of the Sloan Digital Sky Survey: Final data from SDSS-III. *The Astrophysical Journal Supplement Series*, 219:12.

[Aloise et al., 2009] Aloise, D., Deshpande, A., Hansen, P., and Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75:245–248.

[Alon et al., 2003] Alon, J., Sclaroff, S., Kollios, G., and Pavlovic, V. (2003). Discovering clusters in motion time-series data. In *CVPR*, pages 375–381.

[American Cancer Society, 2014] American Cancer Society (2014). Staging of pancreatic cancer. Available at `http://www.cancer.org/cancer/pancreaticcancer/detailedguide/pancreatic-cancer-pdf`.

[Arbesman and Christakis, 2011] Arbesman, S. and Christakis, N. A. (2011). Eurekometrics: Analyzing the nature of discovery. *PLoS Computational Biology*, 7:e1002072.

[Argyros and Ermopoulos, 2003] Argyros, T. and Ermopoulos, C. (2003). Efficient subsequence matching in time series databases under time and amplitude transformations. In *ICDM*, pages 481–484.

[Athar, 2011] Athar, A. (2011). Sentiment analysis of citations using sentence structure-based features. In *ACL*, pages 81–87.

[Avron et al., 2014] Avron, H., Nguyen, H., and Woodruff, D. (2014). Subspace embeddings for the polynomial kernel. In *NIPS*, pages 2258–2266.

[Ayers and Kronenfeld, 2007] Ayers, S. and Kronenfeld, J. (2007). Chronic illness and health-seeking information on the internet. *Health*, 11:327–347.

[Bach and Jordan, 2005] Bach, F. and Jordan, M. (2005). Predictive low-rank decomposition for kernel methods. In *ICML*, pages 33–40.

[Bach and Badaskar, 2007] Bach, N. and Badaskar, S. (2007). A survey of relation extraction. Available at `http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf`.

[Bachem et al., 2016] Bachem, O., Lucic, M., Hassani, H., and Krause, A. (2016). Fast and provably good seedings for k-means. In *NIPS*, pages 55–63.

[Bader and Theofanos, 2003] Bader, J. and Theofanos, M. (2003). Searching for cancer information on the internet: Analyzing natural language search queries. *Journal of Medical Internet Research*, 5.

[Bagnall et al., 2016] Bagnall, A., Bostrom, A., Large, J., and Lines, J. (2016). The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version. *arXiv preprint arXiv:1602.01711*.

[Bagnall and Lines, 2014] Bagnall, A. and Lines, J. (2014). An experimental evaluation of nearest neighbour time series classification. *arXiv preprint arXiv:1406.4757*.

[Bagnall et al., 2017] Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31:606–660.

[Bagnall and Janacek, 2004] Bagnall, A. J. and Janacek, G. J. (2004). Clustering time series from ARMA models with clipped data. In *KDD*, pages 49–58.

[Bahlmann et al., 2002] Bahlmann, C., Haasdonk, B., and Burkhardt, H. (2002). Online handwriting recognition with support vector machines-a kernel approach. In *IWFHR*, pages 49–54.

[Bar-Joseph et al., 2002] Bar-Joseph, Z., Gerber, G., Gifford, D. K., Jaakkola, T. S., and Simon, I. (2002). A new approach to analyzing gene expression time series data. In *RECOMB*, pages 39–48.

[Baragona, 2000] Baragona, R. (2000). Genetic algorithms and cross-correlation clustering of time series. Available at `https://pdfs.semanticscholar.org/dfa4/0e52c9f30cf1f84e769541104d545bbf10d3.pdf`.

[Bartneck and Hu, 2009] Bartneck, C. and Hu, J. (2009). Scientometric analysis of the CHI proceedings. In *SIGCHI*, pages 699–708.

[Batagelj and Cerinšek, 2013] Batagelj, V. and Cerinšek, M. (2013). On bibliographic networks. *Scientometrics*, 96:845–864.

[Batista et al., 2014] Batista, G. E., Keogh, E. J., Tataw, O. M., and De Souza, V. M. (2014). CID: An efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28:634–669.

[Baydogan et al., 2013] Baydogan, M. G., Runger, G., and Tuv, E. (2013). A bag-of-features framework to classify time series. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 35:2796–2802.

[Bayer and Folger, 1966] Bayer, A. E. and Folger, J. (1966). Some correlates of a citation measure of productivity in science. *Sociology of Education*, 39:381–390.

[Beel and Gipp, 2009] Beel, J. and Gipp, B. (2009). Google Scholar's ranking algorithm: The impact of citation counts (An empirical study). In *RCIS*, pages 439–446.

[Begum et al., 2015] Begum, N., Ulanova, L., Wang, J., and Keogh, E. (2015). Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In *SIGKDD*, pages 49–58.

[Bengio et al., 2012] Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538.*

[Bengio et al., 2009] Bengio, Y. et al. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2:1–127.

[Bengio et al., 2007a] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007a). Greedy layer-wise training of deep networks. In *NIPS*, pages 153–160.

[Bengio et al., 2007b] Bengio, Y., LeCun, Y., et al. (2007b). Scaling learning algorithms towards AI. *Large-scale kernel machines*, 34:1–41.

[Bennett et al., 2010] Bennett, P. N., Svore, K., and Dumais, S. T. (2010). Classification-enhanced ranking. In *WWW*, pages 111–120.

[Berndt and Clifford, 1994] Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *AAAI Workshop on KDD*, pages 359–370.

[Bernstam et al., 2009] Bernstam, E., Herskovic, J., and Hersh, W. (2009). Query log analysis in biomedicine. *Handbook of Research on Web Log Analysis*, pages 359–377.

[Bezdek, 2013] Bezdek, J. C. (2013). *Pattern recognition with fuzzy objective function algorithms*. Springer Science.

[Birnholtz et al., 2013] Birnholtz, J., Guha, S., Yuan, Y., Gay, G., and Heller, C. (2013). Cross-campus collaboration: A scientometric and network case study of publication activity across two campuses of a single insitution. *Journal of the American Society for Information Science and Technology*, 64:162–172.

[Biswal et al., 2010] Biswal, B. B., Mennes, M., Zuo, X.-N., Gohel, S., Kelly, C., Smith, S. M., Beckmann, C. F., Adelstein, J. S., Buckner, R. L., Colcombe, S., et al. (2010). Toward discovery science of human brain function. *Proceedings of the National Academy of Sciences*, 107:4734–4739.

[Bonzi, 1982] Bonzi, S. (1982). Characteristics of a literature as predictors of relatedness between cited and citing works. *Journal of the American Society for Information Science*, 33:208–216.

[Bornmann and Daniel, 2008] Bornmann, L. and Daniel, H.-D. (2008). What do citation counts measure? A review of studies on citing behavior. *Journal of Documentation*, 64:45–80.

[Bornmann and Daniel, 2009] Bornmann, L. and Daniel, H.-D. (2009). The state of h index research. Is the h index the ideal way to measure research performance? *EMBO reports*, 10:2.

[Bottou and Bengio, 1995] Bottou, L. and Bengio, Y. (1995). Convergence properties of the k-means algorithms. In *NIPS*, pages 585–592.

[Box and Cox, 1964] Box, G. E. and Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society Series B (Methodological)*, 26:211–252.

[Box et al., 2015] Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: Forecasting and control*. John Wiley & Sons.

[Boyack et al., 2011] Boyack, K. W., Newman, D., Duhon, R. J., Klavans, R., Patek, M., Biberstine, J. R., Schijvenaars, B., Skupin, A., Ma, N., and Börner, K. (2011). Clustering more than two million biomedical publications: Comparing the accuracies of nine text-based similarity approaches. *PLoS ONE*, 6:e18029.

[Brahim-Belhouari and Bermak, 2004] Brahim-Belhouari, S. and Bermak, A. (2004). Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47:705–712.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

[Breiman et al., 1984] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

[Brody et al., 2006] Brody, T., Harnad, S., and Carr, L. (2006). Earlier web usage statistics as predictors of later citation impact. *Journal of the American Society for Information Science and Technology*, 57:1060–1072.

[Bui et al., 2011] Bui, Q.-C., Katrenko, S., and Sloot, P. M. (2011). A hybrid approach to extract protein–protein interactions. *Bioinformatics*, 27:259–265.

[Cai and Ng, 2004] Cai, Y. and Ng, R. (2004). Indexing spatio-temporal trajectories with Chebyshev polynomials. In *SIGMOD*, pages 599–610.

[Callaham et al., 2002] Callaham, M., Wears, R. L., and Weber, E. (2002). Journal prestige, publication bias, and other characteristics associated with citation of published studies in peer-reviewed journals. *Journal of the American Medical Association*, 287:2847–2850.

[Castillo et al., 2007] Castillo, C., Donato, D., and Gionis, A. (2007). Estimating number of citations using author reputation. In *SPIRE*, pages 107–117.

[Castleton et al., 2011] Castleton, K., Fong, T., Wang-Gillam, A., Waqar, M., Jeffe, D., Kehlenbrink, L., Gao, F., and Govindan, R. (2011). A survey of internet utilization among patients with cancer. *Supportive Care in Cancer*, 19:1183–1190.

[Chan et al., 2003] Chan, F., Fu, A., and Yu, C. (2003). Haar wavelets for efficient similarity search of time-series: with and without time warping. *IEEE Transactions on Knowledge and Data Engineering*, 15:686–705.

[Chan and Fu, 1999] Chan, K.-P. and Fu, A. W.-C. (1999). Efficient time series matching by wavelets. In *ICDE*, pages 126–133.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27.

[Chari et al., 2015] Chari, S. T., Kelly, K., Hollingsworth, M. A., Thayer, S. P., Ahlquist, D. A., Andersen, D. K., Batra, S. K., Brentnall, T. A., Canto, M., Cleeter, D. F., et al. (2015). Early detection of sporadic pancreatic cancer: Summative review. *Pancreas*, 44:693.

[Chatfield, 2000] Chatfield, C. (2000). *Time-series forecasting*. CRC Press.

[Chaudhuri, 1996] Chaudhuri, B. (1996). A new definition of neighborhood of a point in multi-dimensional space. *Pattern Recognition Letters*, 17:11–17.

[Chen, 2012] Chen, C. (2012). Predictive effects of structural variation on citation counts. *Journal of the American Society for Information Science and Technology*, 63:431–449.

[Chen and Ng, 2004] Chen, L. and Ng, R. (2004). On the marriage of Lp-norms and edit distance. In *VLDB*, pages 792–803.

[Chen et al., 2005] Chen, L., Özsu, M. T., and Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502.

[Chen et al., 2007a] Chen, Q., Chen, L., Lian, X., Liu, Y., and Yu, J. X. (2007a). Indexable PLA for efficient similarity search. In *VLDB*, pages 435–446.

[Chen et al., 2007b] Chen, Y., Nascimento, M. A., Ooi, B. C., and Tung, A. K. (2007b). Spade: On shape-based pattern detection in streaming time series. In *ICDE*, pages 786–795.

[Chiu et al., 2003] Chiu, B., Keogh, E., and Lonardi, S. (2003). Probabilistic discovery of time series motifs. In *SIGKDD*, pages 493–498.

[Chubin and Moitra, 1975] Chubin, D. E. and Moitra, S. D. (1975). Content analysis of references: Adjunct or alternative to citation counting? *Social Studies of Science*, 5:423–441.

[Cline and Haynes, 2001] Cline, R. and Haynes, K. (2001). Consumer health information seeking on the internet: the state of the art. *Health Education Research*, 16:671–692.

[Cole and Cole, 1967] Cole, S. and Cole, J. R. (1967). Scientific output and recognition: A study in the operation of the reward system in science. *American Sociological Review*, 32:391–403.

[Cooley and Tukey, 1965] Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273–297.

[Coughlin et al., 2000] Coughlin, S., Calle, E., Patel, A., and Thun, M. (2000). Predictors of pancreatic cancer mortality among a large cohort of united states adults. *Cancer Causes & Control*, 11:915–923.

[Cox and Cox, 2000] Cox, T. F. and Cox, M. A. (2000). *Multidimensional scaling*. CRC press.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

[Cui et al., 2016] Cui, Z., Chen, W., and Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*.

[Cunningham and Ghahramani, 2015] Cunningham, J. P. and Ghahramani, Z. (2015). Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16:2859–2900.

[Cuturi, 2011] Cuturi, M. (2011). Fast global alignment kernels. In *ICML*, pages 929–936.

[Cuturi et al., 2007] Cuturi, M., Vert, J.-P., Birkenes, O., and Matsui, T. (2007). A kernel for time series based on global alignments. In *ICASSP*, volume 2, pages II–413.

[Das et al., 1998] Das, G., Lin, K.-I., Mannila, H., Renganathan, G., and Smyth, P. (1998). Rule discovery from time series. In *KDD*, pages 16–22.

[De Choudhury et al., 2013] De Choudhury, M., Counts, S., and Horvitz, E. (2013). Predicting postpartum changes in emotion and behavior via social media. In *SIGCHI*, pages 3267–3276.

[De Choudhury et al., 2014] De Choudhury, M., Morris, M. R., and White, R. (2014). Seeking and sharing health information online: comparing search engines and social media. In *SIGCHI*, pages 1365–1376.

[DeLong et al., 1988] DeLong, E. R., DeLong, D. M., and Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845.

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

[Deng et al., 2013] Deng, H., Runger, G., Tuv, E., and Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153.

[Dietterich, 2002] Dietterich, T. (2002). Machine learning for sequential data: A review. *Structural, syntactic, and statistical pattern recognition*, pages 227–246.

[Dimitriadou et al., 2002] Dimitriadou, E., Weingessel, A., and Hornik, K. (2002). A combination scheme for fuzzy clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 16:901–912.

[Ding et al., 2008] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E. (2008). Querying and mining of time series data: Experimental comparison of representations and distance measures. *PVLDB*, 1:1542–1552.

[Ding et al., 2015] Ding, R., Wang, Q., Dang, Y., Fu, Q., Zhang, H., and Zhang, D. (2015). Yading: Fast clustering of large-scale time series data. *PVLDB*, 8:473–484.

[Ding et al., 2013] Ding, Y., Song, M., Han, J., Yu, Q., Yan, E., Lin, L., and Chambers, T. (2013). Entitymetrics: Measuring the impact of entities. *PloS ONE*, 8:e71416.

[Ding et al., 2009] Ding, Y., Yan, E., Frazho, A., and Caverlee, J. (2009). PageRank for ranking authors in co-citation networks. *Journal of the American Society for Information Science and Technology*, 60:2229–2243.

[Dong et al., 2015] Dong, Y., Johnson, R. A., and Chawla, N. V. (2015). Will this paper increase your h-index?: Scientific impact prediction. In *WSDM*, pages 149–158.

[Downey et al., 2007] Downey, D., Dumais, S., and Horvitz, E. (2007). Models of searching and browsing: Languages, studies, and application. In *IJCAI*, pages 2740–2747.

[Drineas and Mahoney, 2005] Drineas, P. and Mahoney, M. W. (2005). On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175.

[Edge, 1979] Edge, D. (1979). Quantitative measures of communication in science: A critical review. *History of Science*, 17:102–134.

[Esling and Agon, 2012] Esling, P. and Agon, C. (2012). Time-series data mining. *ACM Computing Surveys*, 45:12.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *SIGKDD*, pages 226–231.

[Everhart and Wright, 1995] Everhart, J. and Wright, D. (1995). Diabetes mellitus as a risk factor for pancreatic cancer: a meta-analysis. *Journal of the American Medical Association*, 273:1605–1609.

[Eysenbach, 2011] Eysenbach, G. (2011). Can tweets predict citations? Metrics of social impact based on Twitter and correlation with traditional metrics of scientific impact. *Journal of Medical Internet Research*, 13:e123.

[Faloutsos et al., 1994] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429.

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

[Filippone et al., 2008] Filippone, M., Camastra, F., Masulli, F., and Rovetta, S. (2008). A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41:176–190.

[Fiterau et al., 2017] Fiterau, M., Bhooshan, S., Fries, J., Bournhonesque, C., Hicks, J., Halilaj, E., Ré, C., and Delp, S. (2017). Shortfuse: Biomedical time series representations in the presence of structured information. *arXiv preprint arXiv:1705.04790*.

[Fourney et al., 2015] Fourney, A., White, R. W., and Horvitz, E. (2015). Exploring time-dependent concerns about pregnancy and childbirth from search logs. In *SIGCHI*, pages 737–746.

[Fox and Duggan, 2013] Fox, S. and Duggan, M. (2013). Health online 2013.

[Freeman, 1977] Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41.

[Freeman, 1978] Freeman, L. C. (1978). Centrality in social networks: Conceptual clarification. *Social Networks*, 3:215–239.

[Frentzos et al., 2007] Frentzos, E., Gratsias, K., and Theodoridis, Y. (2007). Index-based most similar trajectory search. In *ICDE*, pages 816–825.

[Friedman, 2001] Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

[Friedman, 1937] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701.

[Frigo and Johnson, 2005] Frigo, M. and Johnson, S. G. (2005). The design and implementation of FFTW3. *Proceedings of the IEEE*, 93:216–231.

[Fu and Aliferis, 2008] Fu, L. D. and Aliferis, C. (2008). Models for predicting and explaining citation count of biomedical articles. In *AMIA*, pages 222–226.

[Fu et al., 2014] Fu, T. Z., Song, Q., and Chiu, D. M. (2014). The academic social network. *Scientometrics*, 101:203–239.

[Fuchs et al., 1996] Fuchs, C., Colditz, G., Stampfer, M., Giovannucci, E., Hunter, D., Rimm, E., Willett, W., and Speizer, F. (1996). A prospective study of cigarette smoking and the risk of pancreatic cancer. *Archives of Internal Medicine*, 156:2255–2260.

[Fulcher, 2017] Fulcher, B. D. (2017). Feature-based time-series analysis. *arXiv preprint arXiv:1709.08055*.

[Fulcher and Jones, 2014] Fulcher, B. D. and Jones, N. S. (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26:3026–3037.

[Fulcher et al., 2013] Fulcher, B. D., Little, M. A., and Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of The Royal Society Interface*, 10:20130048.

[Funk and Owen-Smith, 2012] Funk, R. and Owen-Smith, J. (2012). A dynamic network approach to breakthrough innovation. *arXiv preprint arXiv:1212.3559*.

[Gardner, 2006] Gardner, E. (2006). Exponential smoothing: The state of the art, Part II. *International Journal of Forecasting*, 22:637–666.

[Garfield, 2006] Garfield, E. (2006). Citation indexes for science: A new dimension in documentation through association of ideas. *International Journal of Epidemiology*, 35:1123–1127.

[Garfield and Malin, 1968] Garfield, E. and Malin, M. V. (1968). Can Nobel Prize winners be predicted? In *AAAS*.

[Gavrilov et al., 2000] Gavrilov, M., Anguelov, D., Indyk, P., and Motwani, R. (2000). Mining the stock market: Which measure is best? In *KDD*, pages 487–496.

[Ge and Smyth, 2000] Ge, X. and Smyth, P. (2000). Deformable markov model templates for time-series pattern matching. In *SIGKDD*, pages 81–90.

[Gehrke et al., 2003] Gehrke, J., Ginsparg, P., and Kleinberg, J. (2003). Overview of the 2003 KDD Cup. *ACM SIGKDD Explorations Newsletter*, 5:149–151.

[Ginsberg et al., 2009] Ginsberg, J., Mohebbi, M., Patel, R., Brammer, L., Smolinski, M., and Brilliant, L. (2009). Detecting influenza epidemics using search engine query data. *Nature*, 457:1012–1014.

[Gionis et al., 1999] Gionis, A., Indyk, P., Motwani, R., et al. (1999). Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529.

[Gittens and Mahoney, 2013] Gittens, A. and Mahoney, M. W. (2013). Revisiting the nyström method for improved large-scale machine learning. *The Journal of Machine Learning Research*, 28:567–575.

[Giusti and Batista, 2013] Giusti, R. and Batista, G. E. (2013). An empirical comparison of dissimilarity measures for time series classification. In *BRACIS*, pages 82–88.

[Goddard et al., 2003] Goddard, S., Harms, S. K., Reichenbach, S. E., Tadesse, T., and Waltman, W. J. (2003). Geospatial decision support for drought risk management. *Communications of the ACM*, 46:35–37.

[Golay et al., 1998] Golay, X., Kollias, S., Stoll, G., Meier, D., Valavanis, A., and Boesiger, P. (1998). A new correlation-based fuzzy logic clustering algorithm for fMRI. *Magnetic Resonance in Medicine*, 40:249–260.

[Goldin and Kanellakis, 1995] Goldin, D. Q. and Kanellakis, P. C. (1995). On similarity queries for time-series data: Constraint specification and implementation. In *CP*, pages 137–153.

[Goldstein et al., 1995] Goldstein, A. M., Fraser, M. C., Struewing, J. P., Hussussian, C. J., Ranade, K., Zametkin, D. P., Fontaine, L. S., Organic, S. M., Dracopoli, N. C., Clark Jr, W. H., et al. (1995). Increased risk of pancreatic cancer in melanoma-prone kindreds with p16 ink4 mutations. *The New England Journal of Medicine*, 333:970–975.

[Golub and Van Loan, 2012] Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*. JHU Press.

[Gou et al., 2012] Gou, J., Yi, Z., Du, L., and Xiong, T. (2012). A local mean-based k-nearest centroid neighbor classifier. *The Computer Journal*, 55:1058–1071.

[Goutte et al., 1999] Goutte, C., Toft, P., Rostrup, E., Nielsen, F., and Hansen, L. (1999). On clustering fMRI time series. *NeuroImage*, 9:298–310.

[Grabocka et al., 2014] Grabocka, J., Schilling, N., Wistuba, M., and Schmidt-Thieme, L. (2014). Learning time-series shapelets. In *SIGKDD*, pages 392–401.

[Grueber and Studt, 2012] Grueber, M. and Studt, T. (2012). Global R&D funding forecast. *R&D Magazine*, 16:3–35.

[Guha et al., 2003] Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. (2003). Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15:515–528.

[Guha et al., 2013] Guha, S., Steinhardt, S., Ahmed, S. I., and Lagoze, C. (2013). Following bibliometric footprints: The ACM digital library and the evolution of computer science. In *JCDL*, pages 139–142.

[Gupta et al., 1996] Gupta, L., Molfese, D. L., Tammana, R., and Simos, P. G. (1996). Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Transactions on Biomedical Engineering*, 43:348–356.

[Gupta and Manning, 2010] Gupta, S. and Manning, C. D. (2010). Identifying focus, techniques and domain of scientific papers. In *CSSWC*.

[Guyon et al., 1993] Guyon, I., Boser, B., and Vapnik, V. (1993). Automatic capacity tuning of very large vc-dimension classifiers. In *NIPS*, pages 147–155.

[Haasdonk and Bahlmann, 2004] Haasdonk, B. and Bahlmann, C. (2004). Learning with distance substitution kernels. In *DAGM*, pages 220–227.

[Halkidi et al., 2001] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145.

[Hall et al., 2008] Hall, D., Jurafsky, D., and Manning, C. D. (2008). Studying the history of ideas using topic models. In *EMNLP*, pages 363–371.

[Hamid et al., 2014] Hamid, R., Xiao, Y., Gittens, A., and DeCoste, D. (2014). Compact random feature maps. In *ICML*, pages 19–27.

[Hamilton, 1994] Hamilton, J. D. (1994). *Time series analysis*. Princeton University Press.

[Han et al., 2011] Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[Hansen and Jaumard, 1997] Hansen, P. and Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical Programming*, 79:191–215.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics.

[Haussler, 1999] Haussler, D. (1999). Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz.

[Havemann and Larsen, 2015] Havemann, F. and Larsen, B. (2015). Bibliometric indicators of young authors in astrophysics: Can later stars be predicted? *Scientometrics*, 102:1413–1434.

[Helft, 2011] Helft, P. (2011). Patients with cancer, internet information, and the clinical encounter: a taxonomy of patient users. In *American Society of Clinical Oncology Educational Book*, pages e89–92.

[Hilborn, 2000] Hilborn, R. C. (2000). *Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers*. Oxford University Press.

[Hills et al., 2014] Hills, J., Lines, J., Baranauskas, E., Mapp, J., and Bagnall, A. (2014). Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28:851–881.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18:1527–1554.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313:504–507.

[Hirsch, 2005] Hirsch, J. E. (2005). An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102:16569.

[Hirsch, 2007] Hirsch, J. E. (2007). Does the h index have predictive power? *Proceedings of the National Academy of Sciences*, 104:19193–19198.

[Hofmann et al., 2008] Hofmann, T., Schölkopf, B., and Smola, A. J. (2008). Kernel methods in machine learning. *The Annals of Statistics*, pages 1171–1220.

[Honda et al., 2002] Honda, R., Wang, S., Kikuchi, T., and Konishi, O. (2002). Mining of moving objects from time-series images and its application to satellite weather imagery. *Journal of Intelligent Information Systems*, 19:79–93.

[Hönekopp and Khan, 2012] Hönekopp, J. and Khan, J. (2012). Future publication success in science is better predicted by traditional measures than by the h index. *Scientometrics*, 90:843–853.

[Hőppner and Klawonn, 2009] Hőppner, F. and Klawonn, F. (2009). Compensation of translational displacement in time series clustering using cross correlation. In *Advances in Intelligent Data Analysis VIII*, pages 71–82.

[Horn et al., 2004] Horn, D., Finholt, T., Birnholtz, J., Motwani, D., and Jayaraman, S. (2004). Six degrees of jonathan grudin: A social network analysis of the evolution and impact of cscw research. In *CSCW*.

[Hotelling, 1933] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417.

[Hotelling, 1936] Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28:321–377.

[Hruban et al., 2000] Hruban, R., Goggins, M., Parsons, J., and Kern, S. (2000). Progression model for pancreatic cancer. *Clinical Cancer Research*, 6:2969–2972.

[Hu et al., 2013] Hu, B., Chen, Y., and Keogh, E. (2013). Time series classification under more realistic assumptions. In *SDM*, pages 578–586.

[Hüsken and Stagge, 2003] Hüsken, M. and Stagge, P. (2003). Recurrent neural networks for time series classification. *Neurocomputing*, 50:223–235.

[Huxley et al., 2005] Huxley, R., Ansary-Moghaddam, A., De González, A. B., Barzi, F., and Woodward, M. (2005). Type-II diabetes and pancreatic cancer: A meta-analysis of 36 studies. *British Journal of Cancer*, 92:2076–2083.

[Hyndman and Athanasopoulos, 2014] Hyndman, R. J. and Athanasopoulos, G. (2014). *Forecasting: principles and practice.* OTexts.

[Ibáñez et al., 2009] Ibáñez, A., Larrañaga, P., and Bielza, C. (2009). Predicting citation count of bioinformatics papers within four years of publication. *Bioinformatics*, 25:3303–3309.

[Jain, 1989] Jain, A. K. (1989). *Fundamentals of digital image processing.* Prentice-Hall, Inc.

[Jasta, 2016] Jasta, A. (2016). Observability at Twitter: Technical overview, part I. Available at `https://blog.twitter.com/engineering/en_us/a/2016/observability-at-twitter-technical-overview-part-i.html`.

[Joachims, 1998] Joachims, T. (1998). *Text Categorization with Support Vector Machines: Learning with Many Relevant Features.* Springer.

[Joachims, 2002] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *SIGKDD*, pages 133–142.

[Jones et al., 2006] Jones, R., Rey, B., Madani, O., and Greiner, W. (2006). Generating query substitutions. In *WWW*, pages 387–396.

[Kalpakis et al., 2001] Kalpakis, K., Gada, D., and Puttagunta, V. (2001). Distance measures for effective clustering of ARIMA time-series. In *ICDM*, pages 273–280.

[Kar and Karnick, 2012] Kar, P. and Karnick, H. (2012). Random feature maps for dot product kernels. In *AISTATS*, pages 583–591.

[Kate, 2016] Kate, R. J. (2016). Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 30:283–312.

[Katznelson, 2004] Katznelson, Y. (2004). *An introduction to harmonic analysis.* Cambridge University Press.

[Kaufman and Rousseeuw, 2009] Kaufman, L. and Rousseeuw, P. (2009). *Finding groups in data: An introduction to cluster analysis.* John Wiley & Sons.

[Keogh, 2006] Keogh, E. (2006). A decade of progress in indexing and mining large time series databases. In *VLDB*, pages 1268–1268.

[Keogh et al., 2001a] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001a). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3:263–286.

[Keogh et al., 2001b] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001b). Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30:151–162.

[Keogh and Kasetty, 2003] Keogh, E. and Kasetty, S. (2003). On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7:349–371.

[Keogh and Lin, 2005] Keogh, E. and Lin, J. (2005). Clustering of time-series subsequences is meaningless: Implications for previous and future research. *Knowledge and Information Systems*, 8:154–177.

[Keogh et al., 2005] Keogh, E., Lin, J., and Fu, A. (2005). HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *ICDM*, pages 226–233.

[Keogh et al., 2004] Keogh, E., Palpanas, T., Zordan, V. B., Gunopulos, D., and Cardle, M. (2004). Indexing large human-motion databases. In *VLDB*, pages 780–791.

[Keogh and Ratanamahatana, 2005] Keogh, E. and Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7:358–386.

[Keogh et al., 2015] Keogh, E., Xi, X., Wei, L., and Ratanamahatana, C. A. (2015). The UCR Time Series Classification/Clustering Homepage. Available at `www.cs.ucr.edu/~eamonn/time_series_data`.

[Kessler, 1965] Kessler, M. M. (1965). Comparison of the results of bibliographic coupling and analytic subject indexing. *Journal of the Association for Information Science and Technology*, 16:223–233.

[Kim et al., 2001] Kim, S.-W., Park, S., and Chu, W. W. (2001). An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*, pages 607–614.

[Kin-pong and Ada, 1999] Kin-pong, C. and Ada, F. (1999). Efficient time series matching by wavelets. In *ICDE*, pages 126–133.

[Klapman and Malafa, 2008] Klapman, J. and Malafa, M. P. (2008). Early detection of pancreatic cancer: Why, who, and how to screen. *Cancer Control*, 15:280–287.

[Knorr-Cetina, 1999] Knorr-Cetina, K. (1999). *Epistemic Cultures: How the Sciences Make Knowledge*. Harvard University Press.

[Korn et al., 1997] Korn, F., Jagadish, H. V., and Faloutsos, C. (1997). Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD*, pages 289–300.

[Kulesza and Taskar, 2010] Kulesza, A. and Taskar, B. (2010). Structured determinantal point processes. In *NIPS*, pages 1171–1179.

[Kulis and Grauman, 2012] Kulis, B. and Grauman, K. (2012). Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:1092–1104.

[Kulkarni et al., 2007] Kulkarni, A. V., Busse, J. W., and Shams, I. (2007). Characteristics associated with citation rate of the medical literature. *PloS ONE*, 2:e403.

[Kumar et al., 2012] Kumar, S., Mohri, M., and Talwalkar, A. (2012). Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 13:981–1006.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

[Lane, 2010] Lane, J. (2010). Let's make science metrics more scientific. *Nature*, 464:488–489.

[Lane and Bertuzzi, 2011] Lane, J. and Bertuzzi, S. (2011). Measuring the results of science investments. *Science*, 331:678–680.

[Längkvist et al., 2014] Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24.

[Latour, 1988] Latour, B. (1988). *Science in Action: How to Follow Scientists and Engineers Through Society.* Harvard University Press.

[Lau and Horvitz, 1999] Lau, T. and Horvitz, E. (1999). *Patterns of search: analyzing and modeling web query refinement.* Springer.

[Lauckner and Hsieh, 2013] Lauckner, C. and Hsieh, G. (2013). The presentation of health-related search results and its impact on negative emotional outcomes. In *SIGCHI*, pages 333–342.

[Laurance et al., 2013] Laurance, W. F., Useche, D. C., Laurance, S. G., and Bradshaw, C. J. (2013). Predicting publication success for biologists. *BioScience*, 63:817–823.

[Lecun and Bengio, 1995] Lecun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In *The handbook of brain theory and neural networks*. MIT Press.

[Lee et al., 2007] Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient sparse coding algorithms. In *NIPS*, pages 801–808.

[Lee et al., 2009a] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009a). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pages 609–616.

[Lee et al., 2009b] Lee, H., Pham, P., Largman, Y., and Ng, A. Y. (2009b). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, pages 1096–1104.

[Lee and Verleysen, 2007] Lee, J. A. and Verleysen, M. (2007). *Nonlinear dimensionality reduction*. Springer Science & Business Media.

[Legmann et al., 1998] Legmann, P., Vignaux, O., Dousset, B., Baraza, A., Palazzo, L., Dumontier, I., Coste, J., Louvel, A., Roseau, G., Couturier, D., et al. (1998). Pancreatic tumors: comparison of dual-phase helical ct and endoscopic sonography. *American Journal of Roentgenology*, 170:1315–1322.

[Li et al., 2016] Li, C., Jegelka, S., and Sra, S. (2016). Fast dpp sampling for nystrom with application to kernel methods. *arXiv preprint arXiv:1603.06052*.

[Li et al., 1996] Li, C.-S., Yu, P. S., and Castelli, V. (1996). Hierarchyscan: A hierarchical similarity search algorithm for databases of long sequences. In *ICDE*, pages 546–553.

[Li et al., 1998] Li, C.-S., Yu, P. S., and Castelli, V. (1998). MALM: A framework for mining sequence database at multiple abstraction levels. In *CIKM*, pages 267–272.

[Li et al., 2004] Li, D., Xie, K., Wolff, R., and Abbruzzese, J. L. (2004). Pancreatic cancer. *The Lancet*, 363:1049–1057.

[Lian et al., 2007] Lian, X., Chen, L., Yu, J. X., Wang, G., and Yu, G. (2007). Similarity match over high speed time-series streams. In *ICDE*, pages 1086–1095.

[Liao, 2005] Liao, W. (2005). Clustering of time series dataÑa survey. *Pattern Recognition*, 38:1857–1874.

[Liberty, 2013] Liberty, E. (2013). Simple and deterministic matrix sketching. In *SIGKDD*, pages 581–588.

[Lin et al., 2003] Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *DMKD*, pages 2–11.

[Lin et al., 2012] Lin, J., Khade, R., and Li, Y. (2012). Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315.

[Lin et al., 2004] Lin, J., Vlachos, M., Keogh, E., and Gunopulos, D. (2004). Iterative incremental clustering of time series. In *EDBT*, pages 106–122.

[Lines and Bagnall, 2014] Lines, J. and Bagnall, A. (2014). Ensembles of elastic distance measures for time series classification. In *SDM*, pages 524–532.

[Lines and Bagnall, 2015] Lines, J. and Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29:565–592.

[Loboz et al., 2010] Loboz, C., Smyl, S., and Nath, S. (2010). Datagarage: Warehousing massive performance data on commodity servers. *Proceedings of the VLDB Endowment*, 3:1447–1458.

[Lokker et al., 2008] Lokker, C., McKibbon, K., McKinlay, R. J., Wilczynski, N. L., and Haynes, R. B. (2008). Prediction of citation counts for clinical articles at two years using data available within three weeks of publication: Retrospective cohort study. *British Medical Journal*, 336:655–657.

[Losiewicz et al., 2000] Losiewicz, P., Oard, D. W., and Kostoff, R. N. (2000). Textual data mining to support science and technology management. *Journal of Intelligent Information Systems*, 15:99–119.

[Louis and Nenkova, 2013] Louis, A. and Nenkova, A. (2013). What Makes Writing Great? First Experiments on Article Quality Prediction in the Science Journalism Domain. *Transactions of the Association for Computational Linguistics*, 1.

[Lowenfels and Maisonneuve, 2006] Lowenfels, A. B. and Maisonneuve, P. (2006). Epidemiology and risk factors for pancreatic cancer. *Best Practice & Research Clinical Gastroenterology*, 20:197–209.

[Lowenfels et al., 1993] Lowenfels, A. B., Maisonneuve, P., Cavallini, G., Ammann, R. W., Lankisch, P. G., Andersen, J. R., Dimagno, E. P., Andren-Sandberg, A., and Domellof, L. (1993). Pancreatitis and the risk of pancreatic cancer. *The New England Journal of Medicine*, 328:1433–1437.

[Luenberger, 1979] Luenberger, D. G. (1979). *Introduction to dynamic systems: theory, models, and applications*, volume 1. Wiley New York.

[Lütkepohl, 2005] Lütkepohl, H. (2005). *New introduction to multiple time series analysis.* Springer Science & Business Media.

[Lynch et al., 1996] Lynch, H. T., Smyrk, T., Kern, S. E., Hruban, R. H., Lightdale, C. J., Lemon, S. J., Lynch, J. F., Fusaro, L. R., Fusaro, R. M., and Ghadirian, P. (1996). Familial pancreatic cancer: a review. In *Seminars in Oncology*, volume 23, pages 251–275.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *BSMSP*, pages 281–297.

[MacRoberts and MacRoberts, 1984] MacRoberts, M. H. and MacRoberts, B. R. (1984). The negational reference: Or the art of dissembling. *Social Studies of Science*, 14:91–94.

[Mahajan et al., 2009] Mahajan, M., Nimbhorkar, P., and Varadarajan, K. (2009). The planar k-means problem is NP-hard. In *WALCOM*, pages 274–285.

[Mahdavinejad et al., 2017] Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., and Sheth, A. P. (2017). Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*.

[Mairal et al., 2009] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In *ICML*, pages 689–696.

[Mairal et al., 2010] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60.

[Maji and Berg, 2009] Maji, S. and Berg, A. C. (2009). Max-margin additive classifiers for detection. In *ICCV*, pages 40–47.

[Mallat, 1999] Mallat, S. (1999). *A wavelet tour of signal processing.* Academic press.

[Manjunatha et al., 2003] Manjunatha, J. N., Sivaramakrishnan, K. R., Pandey, R. K., and Murthy, M. N. (2003). Citation prediction using time series approach KDD Cup 2003 (task 1). *ACM SIGKDD Explorations Newsletter*, 5:152–153.

[Mantegna, 1999] Mantegna, R. N. (1999). Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems*, 11:193–197.

[Masci et al., 2011] Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. *ICANN*, pages 52–59.

[McCarty et al., 2013] McCarty, C., Jawitz, J. W., Hopkins, A., and Goldman, A. (2013). Predicting author h-index using characteristics of the co-author network. *Scientometrics*, 96:1–17.

[McKeown et al., 2016] McKeown, K., Hal, D., Snigdha, C., John, P., Kapil, T., Pablo, B., Or, B., Suvarna, B., Michael, C., R., F. K., Luis, G., Rahul, J., Ben, K., Kevin, M., Taesun, M., Arvind, N., Diarmuid, O., Dragomir, R., Clay, T., and Simone, T. (2016). Predicting the impact of scientific concepts using full-text features. *Journal of the Association for Information Science and Technology*, 67:2684–2696.

[Meesrikamolkul et al., 2012] Meesrikamolkul, W., Niennattrakul, V., and Ratanamahatana, C. A. (2012). Shape-based clustering for time series data. In *PAKDD*, pages 530–541.

[Megalooikonomou et al., 2004] Megalooikonomou, V., Li, G., and Wang, Q. (2004). A dimensionality reduction technique for efficient similarity analysis of time series databases. In *CIKM*, pages 160–161.

[Megalooikonomou et al., 2005] Megalooikonomou, V., Wang, Q., Li, G., and Faloutsos, C. (2005). A multiresolution symbolic representation of time series. In *ICDE*, pages 668–679.

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446.

[Mertz et al., 2000] Mertz, H. R., Sechopoulos, P., Delbeke, D., and Leach, S. D. (2000). Eus, pet, and ct scanning for evaluation of pancreatic adenocarcinoma. *Gastrointestinal Endoscopy*, 52:367–371.

[Michaud, 2004] Michaud, D. (2004). Epidemiology of pancreatic cancer. *Minerva Chirurgica*, 59:99–111.

[Mitani and Hamamoto, 2000] Mitani, Y. and Hamamoto, Y. (2000). Classifier design based on the use of nearest neighbor samples. In *ICPR*, pages 769–772.

[Mitani and Hamamoto, 2006] Mitani, Y. and Hamamoto, Y. (2006). A local mean-based nonparametric classifier. *Pattern Recognition Letters*, 27:1151–1159.

[Moravcsik and Murugesan, 1975] Moravcsik, M. J. and Murugesan, P. (1975). Some results on the function and quality of citations. *Social Studies of Science*, 5:86–92.

[Morse and Patel, 2007] Morse, M. D. and Patel, J. M. (2007). An efficient and accurate method for evaluating time series similarity. In *SIGMOD*, pages 569–580.

[Mueen et al., 2014] Mueen, A., Hamooni, H., and Estrada, T. (2014). Time series join on subsequence correlation. In *ICDM*, pages 450–459.

[Mueen et al., 2011] Mueen, A., Keogh, E., and Young, N. (2011). Logical-shapelets: an expressive primitive for time series classification. In *KDD*, pages 1154–1162.

[Mueen et al., 2010] Mueen, A., Nath, S., and Liu, J. (2010). Fast approximate correlation for massive time-series data. In *SIGMOD*, pages 171–182.

[Müller et al., 1997] Müller, K.-R., Smola, A. J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., and Vapnik, V. (1997). Predicting time series with support vector machines. In *ICANN*, pages 999–1004.

[Myers, 1992] Myers, G. (1992). Ôin this paper we reportÉ'Õ: Speech acts and scientific facts. *Journal of Pragmatics*, 17:295–313.

[Nanopoulos et al., 2001] Nanopoulos, A., Alcock, R., and Manolopoulos, Y. (2001). Feature-based classification of time-series data. *International Journal of Computer Research*, 10:49–61.

[Narin, 1976] Narin, F. (1976). *Evaluative Bibliometrics: The Use of Publication and Citation Analysis in the Evaluation of Scientific Activity.* National Science Foundation.

[Neelakantan and Collins, 2014] Neelakantan, A. and Collins, M. (2014). Learning dictionaries for named entity recognition using minimal supervision. In *EACL*, pages 452–461.

[Nemenyi, 1963] Nemenyi, P. (1963). *Distribution-free Multiple Comparisons.* PhD thesis, Princeton University.

[Newman, 2003] Newman, M. (2003). Mixing patterns in networks. *Physical Review E*, 67:026126.

[Newman, 2010] Newman, M. (2010). *Networks: An Introduction.* Oxford University Press.

[Ng et al., 2002] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856.

[Niennattrakul and Ratanamahatana, 2009] Niennattrakul, V. and Ratanamahatana, C. A. (2009). Shape averaging under time warping. In *ECTI-CON*, pages 626–629.

[Nyström, 1930] Nyström, E. J. (1930). Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54:185–204.

[Oates, 1999] Oates, T. (1999). Identifying distinctive subsequences in multivariate time series by clustering. In *KDD*, pages 322–326.

[Ofran et al., 2012] Ofran, Y., Paltiel, O., Pelleg, D., Rowe, J. M., and Yom-Tov, E. (2012). Patterns of information-seeking for cancer on the internet: an analysis of real world data. *PloS ONE*, 7:e45921.

[Oglic and Gärtner, 2017] Oglic, D. and Gärtner, T. (2017). Nyström method with kernel k-means++ samples as landmarks. In *ICML*, pages 2652–2660.

[Olshausen and Field, 1997] Olshausen, B. and Field, D. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37:3311–3325.

[Oppenheim and Schafer, 2009] Oppenheim, A. V. and Schafer, R. W. (2009). *Discrete-Time Signal Processing*. Prentice Hall Press, 3rd edition.

[Orfanidis, 1985] Orfanidis, S. (1985). *Optimum signal processing: An introduction*. Macmillan New York.

[Palpanas, 2015] Palpanas, T. (2015). Data series management: the road to big sequence analytics. *ACM SIGMOD Record*, 44:47–52.

[Pan et al., 2012] Pan, R. K., Kaski, K., and Fortunato, S. (2012). World citation and collaboration networks: Uncovering the role of geography in science. *Scientific Reports*, 2:902.

[Papadimitriou et al., 2003] Papadimitriou, S., Brockwell, A., and Faloutsos, C. (2003). Adaptive, hands-off stream mining. In *VLDB*, pages 560–571.

[Papadimitriou et al., 2005] Papadimitriou, S., Sun, J., and Faloutsos, C. (2005). Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708.

[Papadimitriou et al., 2006] Papadimitriou, S., Sun, J., and Yu, P. S. (2006). Local correlation tracking in time series. In *ICDM*, pages 456–465.

[Papapetrou et al., 2011] Papapetrou, P., Athitsos, V., Potamias, M., Kollios, G., and Gunopulos, D. (2011). Embedding-based subsequence matching in time-series databases. *ACM Transactions on Database Systems*, 36:17.

[Paparrizos and Gravano, 2015] Paparrizos, J. and Gravano, L. (2015). k-Shape: Efficient and accurate clustering of time series. In *SIGMOD*, pages 1855–1870.

[Paparrizos and Gravano, 2016] Paparrizos, J. and Gravano, L. (2016). k-shape: Efficient and accurate clustering of time series. *ACM SIGMOD Record*, 45:69–76.

[Paparrizos and Gravano, 2017] Paparrizos, J. and Gravano, L. (2017). Fast and accurate time-series clustering. *ACM Transactions on Database Systems*, 42:8.

[Paparrizos et al., 2016a] Paparrizos, J., White, R. W., and Horvitz, E. (2016a). Detecting devastating diseases in search logs. In *SIGKDD*, pages 559–568.

[Paparrizos et al., 2016b] Paparrizos, J., White, R. W., and Horvitz, E. (2016b). Screening for pancreatic adenocarcinoma using signals from web search logs: Feasibility study and results. *Journal of Oncology Practice*, 12:737–744.

[Paul et al., 2014] Paul, M. J., White, R., and Horvitz, E. (2014). Search and breast cancer: On disruptive shifts of attention over life histories of an illness. Technical report, MSR-TR-2014-144.

[Pearson, 1901] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2:559–572.

[Pelkonen et al., 2015] Pelkonen, T., Franklin, S., Teller, J., Cavallaro, P., Huang, Q., Meza, J., and Veeraraghavan, K. (2015). Gorilla: a fast, scalable, in-memory time series database. *PVLDB*, 8:1816–1827.

[Penner et al., 2013] Penner, O., Petersen, A. M., Pan, R. K., and Fortunato, S. (2013). The case for caution in predicting scientists' future impact. *Physics Today*, 66:8–9.

[Pepe et al., 2001] Pepe, M. S., Etzioni, R., Feng, Z., Potter, J. D., Thompson, M. L., Thornquist, M., Winget, M., and Yasui, Y. (2001). Phases of biomarker development for early detection of cancer. *Journal of the National Cancer Institute*, 93:1054–1061.

[Percival and Walden, 2006] Percival, D. B. and Walden, A. T. (2006). *Wavelet methods for time series analysis*, volume 4. Cambridge university press.

[Peterson et al., 2003] Peterson, G., Aslani, P., and Williams, K. A. (2003). How do consumers search for and appraise information on medicines on the internet? a qualitative study using focus groups. *Journal of Medical Internet Research*, 5.

[Petitjean et al., 2014] Petitjean, F., Forestier, G., Webb, G. I., Nicholson, A. E., Chen, Y., and Keogh, E. (2014). Dynamic time warping averaging of time series allows faster and more accurate classification. In *ICDM*, pages 470–479.

[Petitjean et al., 2016] Petitjean, F., Forestier, G., Webb, G. I., Nicholson, A. E., Chen, Y., and Keogh, E. (2016). Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowledge and Information Systems*, 47:1–26.

[Petitjean and Gançarski, 2012] Petitjean, F. and Gançarski, P. (2012). Summarizing a set of time series by averaging: From Steiner sequence to compact multiple alignment. *Theoretical Computer Science*, 414:76–91.

[Petitjean et al., 2011] Petitjean, F., Ketterlin, A., and Gançarski, P. (2011). A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44:678–693.

[Popivanov and Miller, 2002] Popivanov, I. and Miller, R. J. (2002). Similarity search over time-series data using wavelets. In *ICDE*, pages 212–221.

[Rabiner and Juang, 1986] Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *IEEE Assp Magazine*, 3:4–16.

[Rahimi and Recht, 2008] Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184.

[Rakthanmanon et al., 2012] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., and Keogh, E. (2012). Searching and mining trillions of time series subsequences under dynamic time warping. In *SIGKDD*, pages 262–270.

[Rakthanmanon et al., 2011] Rakthanmanon, T., Keogh, E. J., Lonardi, S., and Evans, S. (2011). Time series epenthesis: Clustering time series streams requires ignoring some data. In *ICDM*, pages 547–556.

[Rand, 1971] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850.

[Ratanamahatana and Keogh, 2004] Ratanamahatana, C. A. and Keogh, E. (2004). Making time-series classification more accurate using learned constraints. In *SDM*, pages 11–22.

[Ravi Kanth et al., 1998] Ravi Kanth, K. V., Agrawal, D., and Singh, A. (1998). Dimensionality reduction for similarity searching in dynamic databases. In *SIGMOD*, pages 166–176.

[Renehan et al., 2008] Renehan, A. G., Tyson, M., Egger, M., Heller, R. F., and Zwahlen, M. (2008). Body-mass index and incidence of cancer: a systematic review and meta-analysis of prospective observational studies. *The Lancet*, 371:569–578.

[Rice, 2006] Rice, J. (2006). *Mathematical statistics and data analysis*. Cengage Learning.

[Richardson, 2009] Richardson, M. (2009). Learning about the world from long-term query logs. *ACM Transactions on the Web*, 2:21.

[Rodriguez and Laio, 2014] Rodriguez, A. and Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, 344:1492–1496.

[Ruiz et al., 2012] Ruiz, E. J., Hristidis, V., Castillo, C., Gionis, A., and Jaimes, A. (2012). Correlating financial time series with micro-blogging activity. In *WSDM*, pages 513–522.

[Rulyak et al., 2003] Rulyak, S. J., Kimmey, M. B., Veenstra, D. L., and Brentnall, T. A. (2003). Cost-effectiveness of pancreatic cancer screening in familial pancreatic cancer kindreds. *Gastrointestinal Endoscopy*, 57:23–29.

[Sadilek et al., 2012] Sadilek, A., Kautz, H., and Silenzio, V. (2012). Modeling spread of disease from social interactions. In *ICWSM*, pages 322–329.

[Sahni and Gonzalez, 1976] Sahni, S. and Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM*, 23:555–565.

[Saito, 1994] Saito, N. (1994). *Local Feature Extraction and Its Applications Using a Library of Bases*. PhD thesis, Yale University.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26:43–49.

[Sakurai et al., 2005a] Sakurai, Y., Papadimitriou, S., and Faloutsos, C. (2005a). Braid: Stream mining through group lag correlations. In *SIGMOD*, pages 599–610.

[Sakurai et al., 2005b] Sakurai, Y., Yoshikawa, M., and Faloutsos, C. (2005b). Ftw: fast similarity search under the time warping distance. In *PODS*, pages 326–337.

[Salvador and Chan, 2004] Salvador, S. and Chan, P. (2004). Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *ICTAI*, pages 576–584.

[Sánchez et al., 1997] Sánchez, J. S., Pla, F., and Ferri, F. J. (1997). On the use of neighbourhood-based non-parametric classifiers. *Pattern Recognition Letters*, 18:1179–1186.

[Sarigöl et al., 2014] Sarigöl, E., Pfitzner, R., Scholtes, I., Garas, A., and Schweitzer, F. (2014). Predicting scientific success based on coauthorship networks. *EPJ Data Science*, 3:9.

[Schäfer, 2016] Schäfer, P. (2016). Scalable time series classification. *Data Mining and Knowledge Discovery*, 30:1273–1298.

[Schäfer and Högqvist, 2012] Schäfer, P. and Högqvist, M. (2012). Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *EDBT*, pages 516–527.

[Schölkopf, 2001] Schölkopf, B. (2001). The kernel trick for distances. In *NIPS*, pages 301–307.

[Schölkopf et al., 1997] Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In *ICANN*, pages 583–588.

[Schölkopf et al., 1998] Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10:1299–1319.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press.

[Schreiber, 2013] Schreiber, M. (2013). How relevant is the predictive power of the h index? a case study of the time-dependent Hirsch index. *Journal of Informetrics*, 7:325–329.

[Sculley, 2010] Sculley, D. (2010). Web-scale k-means clustering. In *WWW*, pages 1177–1178.

[Shasha and Zhu, 2004] Shasha, D. and Zhu, Y. (2004). *High Performance Discovery In Time Series: Techniques And Case Studies (Monographs in Computer Science)*. SpringerVerlag.

[Shatkay and Zdonik, 1996] Shatkay, H. and Zdonik, S. B. (1996). Approximate queries and representations for large data sequences. In *ICDE*, pages 536–545.

[Shimodaira et al., 2002] Shimodaira, H., Noma, K.-i., Nakai, M., and Sagayama, S. (2002). Dynamic time-alignment kernel in support vector machine. In *NIPS*, pages 921–928.

[Shin and Kuboyama, 2008] Shin, K. and Kuboyama, T. (2008). A generalization of haussler's convolution kernel: mapping kernel. In *ICML*, pages 944–951.

[Shindler et al., 2011] Shindler, M., Wong, A., and Meyerson, A. W. (2011). Fast and accurate k-means for large datasets. In *NIPS*, pages 2375–2383.

[Shou et al., 2005] Shou, Y., Mamoulis, N., and Cheung, D. (2005). Fast and exact warping of time series using adaptive segmental approximations. *Machine Learning*, 58:231–267.

[Silva et al., 2016] Silva, D., Batista, G. E., and Keogh, E. (2016). Prefix and suffix invariant dynamic time warping. In *ICDM*, pages 1209–1214.

[Small, 1973] Small, H. (1973). Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the Association for Information Science and Technology*, 24:265–269.

[Small, 2011] Small, H. (2011). Interpreting maps of science using citation context sentiments: a preliminary investigation. *Scientometrics*, 87:373–388.

[Smith, 2007] Smith, J. O. (2007). *Mathematics of the discrete Fourier transform (DFT): with audio applications*. Julius Smith.

[Smola and Schölkopf, 2004] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14:199–222.

[Spiegel-Rösing, 1977] Spiegel-Rösing, I. (1977). Science studies: Bibliometric and content analysis. *Social Studies of Science*, 7:97–113.

[Stefan et al., 2013] Stefan, A., Athitsos, V., and Das, G. (2013). The move-split-merge metric for time series. *Knowledge and Data Engineering, IEEE Transactions on*, 25:1425–1438.

[Struzik and Siebes, 1999] Struzik, Z. R. and Siebes, A. (1999). Measuring time series similarity through large singular features revealed with wavelet transformation. In *DEXA*, pages 162–166.

[Sun et al., 2013] Sun, X., Kaur, J., Milojević, S., Flammini, A., and Menczer, F. (2013). Social dynamics of science. *Scientific Reports*, 3:1069.

[Sutskever and Hinton, 2007] Sutskever, I. and Hinton, G. (2007). Learning multilevel distributed representations for high-dimensional sequences. In *AISTATS*, pages 548–555.

[Suykens et al., 1999] Suykens, J., Lukas, L., Van Dooren, P., De Moor, B., Vandewalle, J., et al. (1999). Least squares support vector machine classifiers: a large scale algorithm. In *ECCTD*, pages 839–842.

[Swales, 1990] Swales, J. (1990). *Genre Analysis: English in Academic and Research*, chapter 7: Research articles in English. Cambridge University Press.

[Talamini et al., 1999] Talamini, G., Bassi, C., Falconi, M., Sartori, N., Salvia, R., Rigo, L., Castagnini, A., Di Francesco, V., Frulloni, L., Bovo, P., et al. (1999). Alcohol and smoking as risk factors in chronic pancreatitis and pancreatic cancer. *Digestive Diseases and Sciences*, 44:1303–1311.

[Tan and Lee, 2014] Tan, C. and Lee, L. (2014). A corpus of sentence-level revisions in academic writing: A step towards understanding statement strength in communication. *arXiv preprint arXiv:1405.1439*.

[Teräsvirta et al., 1993] Teräsvirta, T., Lin, C.-F., and Granger, C. W. (1993). Power of the neural network linearity test. *Journal of Time Series Analysis*, 14:209–220.

[Teufel, 2010] Teufel, S. (2010). *The Structure of Scientific Articles: Applications to Citation Indexing and Summarization*. CSLI Publications.

[Torgerson, 1952] Torgerson, W. S. (1952). Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419.

[Traweek, 1992] Traweek, S. (1992). *Beamtimes and Lifetimes: The World of High Energy Physicists*. Harvard University Press.

[Trotter and Morgan, 2008] Trotter, M. I. and Morgan, D. W. (2008). Patients' use of the internet for health related matters: a study of internet usage in 2000 and 2006. *Health Informatics Journal*, 14:175–181.

[Tsai et al., 2013] Tsai, C.-T., Kundu, G., and Roth, D. (2013). Concept-based analysis of scientific literature. In *CIKM*, pages 1733–1738.

[Uehara and Shimada, 2002] Uehara, K. and Shimada, M. (2002). Extraction of primitive motion and discovery of association rules from human motion data. In *Progress in Discovery Science*, pages 338–348.

[Velden et al., 2010] Velden, T., Haque, A., and Lagoze, C. (2010). A new approach to analyzing patterns of collaboration in co-authorship networks: Mesoscopic analysis and interpretation. *Scientometrics*, 85:219–242.

[Velden and Lagoze, 2013] Velden, T. and Lagoze, C. (2013). The extraction of community structures from publication networks to support ethnographic observations of field differences in scientific communication. *Journal of the American Society for Information Science and Technology*, 64:2405–2427.

[Viana et al., 2013] Viana, M. P., Amancio, D. R., and Costa, L. d. F. (2013). On time-varying collaboration networks. *Journal of Informetrics*, 7:371–378.

[Vlachos et al., 2004] Vlachos, M., Gunopulos, D., and Das, G. (2004). Indexing time-series under conditions of noise. *Data mining in time series databases*, 57:67–100.

[Vlachos et al., 2006] Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., and Keogh, E. (2006). Indexing multidimensional time-series. *The VLDB Journal*, 15:1–20.

[Vlachos et al., 2002] Vlachos, M., Kollios, G., and Gunopulos, D. (2002). Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684.

[Wachman et al., 2009] Wachman, G., Khardon, R., Protopapas, P., and Alcock, C. R. (2009). Kernels for periodic time series arising in astronomy. In *PKDD*, pages 489–505.

[Wang et al., 2014a] Wang, H., Cai, Y.-f., Yang, Y., Zhang, S., and Mamoulis, N. (2014a). Durable queries over historical time series. *IEEE Transactions on Knowledge and Data Engineering*, 26:595–607.

[Wang and Jiang, 1994] Wang, L. and Jiang, T. (1994). On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348.

[Wang et al., 2014b] Wang, S., Xie, S., Zhang, X., Li, Z., Yu, P. S., and Shu, X. (2014b). Future influence ranking of scientific literature. In *SDM*, pages 749–757.

[Wang et al., 2013] Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. (2013). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26:275–309.

[Wang et al., 2006] Wang, X., Smith, K., and Hyndman, R. (2006). Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, 13:335–364.

[Wang et al., 2017] Wang, Z., Yan, W., and Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *IJCNN*, pages 1578–1585.

[Warren Liao, 2005] Warren Liao, T. (2005). Clustering of time series data - a survey. *Pattern Recognition*, 38:1857–1874.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393:409–10.

[West et al., 2013] West, R., White, R. W., and Horvitz, E. (2013). From cookies to cooks: Insights on dietary patterns via analysis of web usage logs. In *WWW*, pages 1399–1410.

[White and Drucker, 2007] White, R. W. and Drucker, S. M. (2007). Investigating behavioral variability in web search. In *WWW*, pages 21–30.

[White et al., 2014] White, R. W., Harpaz, R., Shah, N. H., DuMouchel, W., and Horvitz, E. (2014). Toward enhanced pharmacovigilance using patient-generated data on the internet. *Clinical Pharmacology & Therapeutics*, 96:239–246.

[White and Horvitz, 2009] White, R. W. and Horvitz, E. (2009). Cyberchondria: studies of the escalation of medical concerns in web search. *ACM Transactions on Information Systems*, 27:23.

[White and Horvitz, 2012] White, R. W. and Horvitz, E. (2012). Studies of the onset and persistence of medical concerns in search logs. In *SIGIR*, pages 265–274.

[White and Horvitz, 2014] White, R. W. and Horvitz, E. (2014). From health search to healthcare: explorations of intention and utilization via query logs and user surveys. *Journal of the American Medical Informatics Association*, 21:49–55.

[White et al., 2013] White, R. W., Tatonetti, N. P., Shah, N. H., Altman, R. B., and Horvitz, E. (2013). Web-scale pharmacovigilance: listening to signals from the crowd. *Journal of the American Medical Informatics Association*, 20:404–408.

[Wick et al., 2013] Wick, M. L., Kobren, A., and McCallum, A. (2013). Large-scale author coreference via hierarchical entity representations. In *PRPM*.

[Wilcoxon, 1945] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, pages 80–83.

[Williams and Seeger, 2001] Williams, C. K. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *NIPS*, pages 682–688.

[Willinger et al., 1998] Willinger, W., Paxson, V., and Taqqu, M. S. (1998). Self-similarity and heavy tails: Structural modeling of network traffic. In Adler, R. J., Feldman, R. E., and Taqqu, M. S., editors, *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, pages 27–53.

[Wilson and Martinez, 1997] Wilson, D. R. and Martinez, T. R. (1997). Instance pruning techniques. In *ICML*, pages 403–411.

[Wilson and Martinez, 2000] Wilson, D. R. and Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine learning*, 38:257–286.

[Wu et al., 2010] Wu, D., Ke, Y., Yu, J. X., Philip, S. Y., and Chen, L. (2010). Detecting leaders from correlated time series. In *DASFAA*, pages 352–367.

[Wu et al., 2008] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14:1–37.

[Xi et al., 2006] Xi, X., Keogh, E., Shelton, C., Wei, L., and Ratanamahatana, C. A. (2006). Fast time series classification using numerosity reduction. In *ICML*, pages 1033–1040.

[Xiong and Yeung, 2002] Xiong, Y. and Yeung, D.-Y. (2002). Mixtures of ARMA models for model-based time series clustering. In *ICDM*, pages 717–720.

[Yan et al., 2011] Yan, R., Tang, J., Liu, X., Shan, D., and Li, X. (2011). Citation count prediction: learning to estimate future citations for literature. In *CIKM*, pages 1247–1252.

[Yang and Leskovec, 2011] Yang, J. and Leskovec, J. (2011). Patterns of temporal variation in online media. In *WSDM*, pages 177–186.

[Yang and Wu, 2006] Yang, Q. and Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5:597–604.

[Yang et al., 2012] Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. (2012). Nyström method vs random fourier features: A theoretical and empirical comparison. In *NIPS*, pages 476–484.

[Ye and Keogh, 2009] Ye, L. and Keogh, E. (2009). Time series shapelets: A new primitive for data mining. In *KDD*, pages 947–956.

[Yeo et al., 1997] Yeo, C. J., Abrams, R. A., Grochow, L. B., Sohn, T. A., Ord, S. E., Hruban, R. H., Zahurak, M. L., Dooley, W. C., Coleman, J., Sauter, P. K., et al. (1997). Pancreaticoduodenectomy for pancreatic adenocarcinoma: postoperative adjuvant chemoradiation improves survival. a prospective, single-institution experience. *Annals of surgery*, 225:621.

[Yi and Faloutsos, 2000] Yi, B.-K. and Faloutsos, C. (2000). Fast time sequence indexing for arbitrary lp norms. In *VLDB*, pages 385–394.

[Yi et al., 1998] Yi, B.-K., Jagadish, H., and Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208.

[Yogatama et al., 2011] Yogatama, D., Heilman, M., O'Connor, B., Dyer, C., Routledge, B. R., and Smith, N. A. (2011). Predicting a scientific community's response to an article. In *EMNLP*, pages 594–604.

[Yu et al., 2015] Yu, J., Blackford, A. L., dal Molin, M., Wolfgang, C. L., and Goggins, M. (2015). Time to progression of pancreatic ductal adenocarcinoma from low-to-high tumour stages. *Gut*, 64:1783–1789.

[Zakaria et al., 2012] Zakaria, J., Mueen, A., and Keogh, E. (2012). Clustering time series using unsupervised-shapelets. In *ICDM*, pages 785–794.

[Zhang et al., 1998] Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14:35–62.

[Zhang et al., 2008] Zhang, K., Tsang, I. W., and Kwok, J. T. (2008). Improved nyström low-rank approximation and error analysis. In *ICML*, pages 1232–1239.

[Zhu et al., 2015] Zhu, X., Turney, P., Lemire, D., and Vellino, A. (2015). Measuring academic influence: Not all citations are equal. *Journal of the Association for Information Science and Technology*, 66:408–427.

[Zhu and Shasha, 2002] Zhu, Y. and Shasha, D. (2002). Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369.

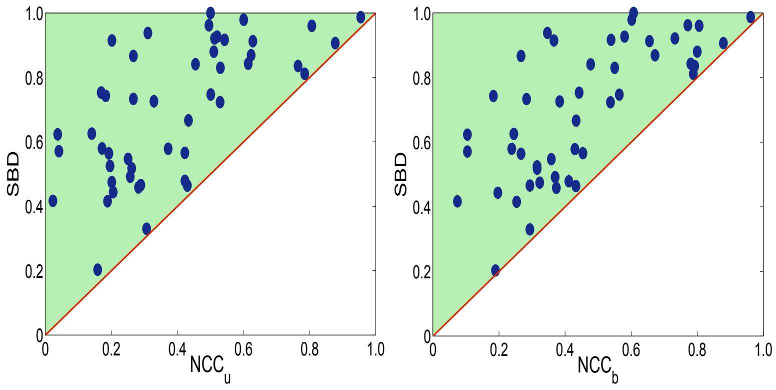
[Ziman, 1968] Ziman, J. M. (1968). *Public Knowledge: An Essay Concerning the Social Dimension of Science*. Cambridge University Press.

Appendix A

Cross-Correlation Variants under Time-Series Normalizations

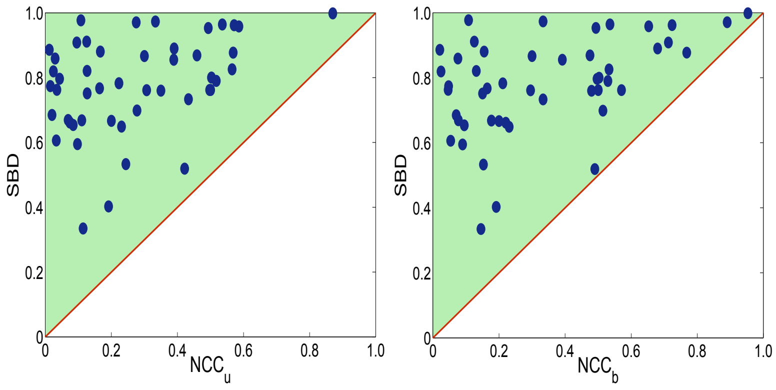
As discussed in Chapter 3 and more specifically in Section 3.3.1, the choice of normalizations for the data and the cross-correlation measure can significantly impact the cross-correlation sequence produced. To understand the sensitivity of cross-correlation variants to data normalizations we evaluate their performance using 48 datasets from the UCR time-series collection [Keogh et al., 2015]. (These 48 datasets are a subset of the 85 datasets used in Chapter 3.) This collection contains z-normalized datasets but the unnormalized versions are not available. Therefore, to experiment with unnormalized versions of these datasets and ensure that the initial sequences differ in amplitude, we first multiply each sequence with a random number chosen individually for that sequence. Then, we perform and study three common time-series normalizations: (1) *OptimalScaling*: to match two time series $\vec{x}$ and $\vec{y}$, we compute their optimal scaling coefficient $c = \frac{\vec{x} \cdot \vec{y}^T}{\vec{y} \cdot \vec{y}^T}$, which is used for every pairwise comparison (e.g., SBD($\vec{x}$,$\vec{y}$) is computed as SBD($\vec{x}$,$c \cdot \vec{y}$)); (2) *ValuesBetween0-1*: we normalize each sequence $\vec{x}$ such that its values fall between 0 and 1, by transforming it into $\vec{x}' = \frac{\vec{x} - min(\vec{x})}{max(\vec{x}) - min(\vec{x})}$; and (3) *z-normalization*: we normalize each sequence $\vec{x}$ such that its mean is 0 and its standard deviation is 1, by transforming it into $\vec{x}' = \frac{\vec{x} - mean(\vec{x})}{std(\vec{x})}$.

We evaluate the 1-NN classification accuracy of each cross-correlation variant, namely,



(a) SBD vs. NCC$_u$ (b) SBD vs. NCC$_b$

Figure A.1: Comparison of SBD, NCC$_b$, and NCC$_u$ over 48 datasets under the *OptimalScaling* normalization. Circles above the diagonal indicate datasets over which SBD has better accuracy than the compared method.



(a) SBD vs. NCC$_u$ (b) SBD vs. NCC$_b$

Figure A.2: Comparison of SBD, NCC$_b$, and NCC$_u$ over 48 datasets under the *ValuesBetween0-1* normalization. Circles above the diagonal indicate datasets over which SBD has better accuracy than the compared method.

SBD, NCC_u, and NCC_b, on each dataset for the three common time-series normalization scenarios. Figure A.1 presents the results for the *OptimalScaling* normalization. In particular, SBD significantly outperforms both NCC_u (Figure A.1a) and NCC_b (Figure A.1b) in all 48 datasets and, in turn, NCC_b outperforms NCC_u in 40 datasets, with statistical significance.

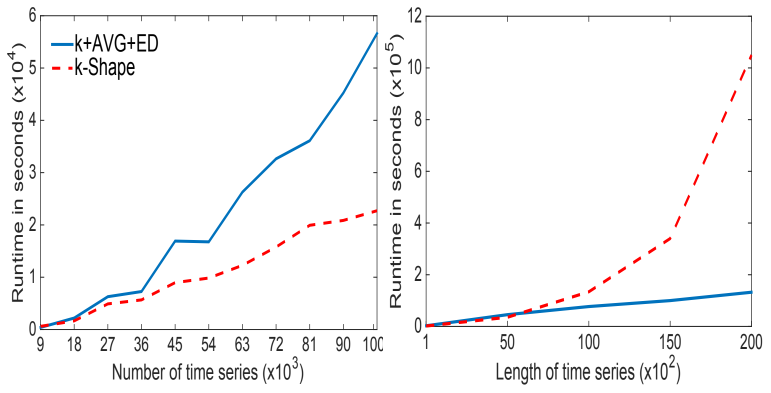
Similarly, for *ValuesBetween0-1* data normalization, SBD significantly outperforms NCC_u (Figure A.2a) and NCC_b (Figure A.2b) in all 48 datasets. NCC_b performs similarly to or better than NCC_u in 41 datasets and the Wilcoxon test suggests this difference in accuracy is statistically significant. Finally, for *z-normalization*, SBD and NCC_b achieve similar performance and both significantly outperform NCC_u.

Therefore, we can conclude that SBD is the most robust cross-correlation variant as it achieves better performance across multiple different time series normalizations. On average, SBD achieves accuracy values of 0.699, 0.779, and 0.795, for *OptimalScaling*, *ValuesBetween0-1*, and *z-normalization* normalizations, respectively.

Appendix B

Scalability of k-Shape

As discussed in Chapter 3 and more specifically in Section 3.3.3, k-Shape requires $\mathcal{O}(\max\{n \cdot k \cdot m \cdot \log(m), n \cdot m^2, k \cdot m^3\})$ time per iteration to cluster n time series of length m into k clusters. The majority of the computational cost of our algorithm depends on the time-series length m, whereas its dependence on the number n of time series is linear. To better understand the scalability of k-Shape, we use the synthetic CBF dataset [Saito, 1994] because it enables experiments with varying values of both n and m without changing any of its general properties. Figure B.1 reports the results of our scalability experiments where both n and m are up to one order of magnitude larger than the biggest dataset in the UCR archive [Keogh et al., 2015]. We report the average CPU runtime of 5 runs. Specifically, in Figure B.1a, we vary n while we set $m = 128$ as in the original CBF dataset: k-Shape, similarly to k+AVG+ED, scales linearly with the number of time series without any loss in accuracy for both methods. Importantly, k-Shape, remains significantly faster than k-AVG+ED with the increasing number of time series because k-Shape requires 45% fewer iterations to converge than k+AVG+ED does. Then, in Figure B.1b, we vary m while we set $n = 18K$: k-Shape still outperforms k-AVG+ED for $m \ll n$ but k-Shape becomes slower, as expected, when the time-series length is large and approaches the total number of time series in the dataset. Similarly to the previous experiment, there is no loss in accuracy for either technique.



(a) Varying n for m=128 (b) Varying m for n=18K

Figure B.1: Runtime of k-AVG+ED and k-Shape as a function of (a) the number of time series n and (b) the time series length m.

Therefore, we can conclude that k-Shape is an efficient and scalable method to cluster large time-series collections in the common case when the time series length is lower than the number of time series (i.e., when $m \ll n$).

Appendix C

Description of the UCR Time-Series Archive

Throughout this thesis, we evaluate our ideas and algorithms using the world's largest public collection of labeled time-series datasets, namely, the UCR time-series datasets archive [Keogh et al., 2015]. As of the writing of this dissertation, the UCR archive consists of 85 real and synthetic datasets that span several different domains. These datasets are annotated such that each time series belongs to only one class. Additionally, each dataset is split into training and test sets. In Table C.1, we provide the name of each dataset in the UCR archive along with the number of time series in the training and test sets, the number of classes, and the lengths of the time series. We perform extensive experimental evaluations using these datasets in Chapters 3 and 4.

Dataset Name	Training Instances	Test Instances	Classes	Length
50words	450	455	50	270
Adiac	390	391	37	176
ArrowHead	36	175	3	251
Beef	30	30	5	470
BeetleFly	20	20	2	512
BirdChicken	20	20	2	512
CBF	30	900	3	128
Car	60	60	4	577
ChlorineConcentration	467	3840	3	166
CinC_ECG_torso	40	1380	4	1639
Coffee	28	28	2	286
Computers	250	250	2	720
Cricket_X	390	390	12	300

Cricket_Y	390	390	12	300
Cricket_Z	390	390	12	300
DiatomSizeReduction	16	306	4	345
DistalPhalanxOutlineAgeGroup	139	400	3	80
DistalPhalanxOutlineCorrect	276	600	2	80
DistalPhalanxTW	139	400	6	80
ECG200	100	100	2	96
ECG5000	500	4500	5	140
ECGFiveDays	23	861	2	136
Earthquakes	139	322	2	512
ElectricDevices	8926	7711	7	96
FaceAll	560	1690	14	131
FaceFour	24	88	4	350
FacesUCR	200	2050	14	131
Fish	175	175	7	463
FordA	1320	3601	2	500
FordB	810	3636	2	500
Gun_Point	50	150	2	150
Ham	109	105	2	431
HandOutlines	370	1000	2	2709
Haptics	155	308	5	1092
Herring	64	64	2	512
InlineSkate	100	550	7	1882
Insect	220	1980	11	256
ItalyPowerDemand	67	1029	2	24
LargeKitchenAppliances	375	375	3	720
Lighting2	60	61	2	637
Lighting7	70	73	7	319
MALLAT	55	2345	8	1024
Meat	60	60	3	448
MedicalImages	381	760	10	99
MiddlePhalanxOutlineAgeGroup	154	400	3	80
MiddlePhalanxOutlineCorrect	291	600	2	80
MiddlePhalanxTW	154	399	6	80
Motes	20	1252	2	84
NonInvasiveFatalECG_Thorax1	1800	1965	42	750
NonInvasiveFatalECG_Thorax2	1800	1965	42	750
OSULeaf	200	242	6	427
OliveOil	30	30	4	570
PhalangesOutlinesCorrect	1800	858	2	80
Phoneme	214	1896	39	1024
Plane	105	105	7	144
ProximalPhalanxOutlineAgeGroup	400	205	3	80
ProximalPhalanxOutlineCorrect	600	291	2	80
ProximalPhalanxTW	205	400	6	80
RefrigerationDevices	375	375	3	720
ScreenType	375	375	3	720
ShapeletSim	20	180	2	500
ShapesAll	600	600	60	512
SmallKitchenAppliances	375	375	3	720
SonyAIBORobotSurface	20	601	2	70
SonyAIBORobotSurfaceII	27	953	2	65
StarLightCurves	1000	8236	3	1024
Strawberry	370	613	2	235
SwedishLeaf	500	625	15	128
Symbols	25	995	6	398
Synthetic_control	300	300	6	60
ToeSegmentation1	40	228	2	277
ToeSegmentation2	36	130	2	343
Trace	100	100	4	275
TwoLeadECG	23	1139	2	82

Two__Patterns	1000	4000	4	128
UWaveGestureLibraryAll	896	3582	8	945
Wafer	1000	6164	2	152
Wine	57	54	2	234
WordsSynonyms	267	638	25	270
Worms	77	181	5	900
WormsTwoClass	77	181	2	900
Yoga	300	3000	2	426
uWaveGestureLibrary__X	896	3582	8	315
uWaveGestureLibrary__Y	896	3582	8	315
uWaveGestureLibrary__Z	896	3582	8	315

Table C.1: Description of the time-series datasets in the UCR archive [Keogh et al., 2015].