Efficient computational methods for understanding human behaviour from mobile phone data

Jordan Frank

Reasoning and Learning Lab
School of Computer Science
Faculty of Science
McGill University
Montréal, Quebec

Submitted: November 2012

A thesis submitted in partial fulfillment for
the degree of Doctor of Philosophy

Declaration of Authorship

“Behavior is a mirror in which every one displays his own image”

Johann Wolfgang von Goethe

Abstract

Smartphones and similar mobile devices present an unprecedented opportunity to collect data about human behaviour. Integrated into daily life, devices are privy to information about who we communicate with, where we are, and how we spend our time. However, without the proper tools for analysis and the ability to scale up to large amounts of data and large numbers of users, this data is of limited value. This thesis presents tools for extracting information about activities and location from sensor data available on commodity smartphones.

We present an algorithm for comparing time series and demonstrate on a large number of benchmark problems that it is competitive with, and often exceeds, the performance of existing algorithms, while being efficient enough to run in real time on a standard smartphone. Activity and gait recognition are presented as applications of our approach, and the performance on these tasks is shown to exceed that of state-of-the-art approaches.

For location detection, we present a thorough study of wifi signals in dynamic, non-stationary environments, and conclude that a new multinomial model is preferable to the standard Gaussian model used to model wifi measurements on a mobile terminal. We present a clustering algorithm for wifi signals that uses an underlying multinomial model, and demonstrate on a large dataset that this approach performs favourably on the task of identifying physical locations from wifi data. By basing our approach on the hierarchical Dirichlet process, the number of identifiable locations scales based on the data, and does not need to be prespecified. This allows for accurate, real-time localization of a mobile device both indoors and outdoors, in a manner that, contrary to existing approaches, protects the privacy of the user.

Even detailed information is of limited value without the ability to convey it to the user in a meaningful way. We present a framework for translating from sensor data into human-readable descriptions of the events taking place. This framework is evaluated on a corpora of 150 GB of sensor data recorded from 38 users over 14 months, and provides for convenient visualization of large amounts of data and accurate reporting, in natural language, of events transpiring in the data.

Abrégé

Les téléphones intelligents et les appareils mobiles semblables nous fournissent une occasion sans précédent de recueillir des données à propos du comportement humain. Puisqu'ils font partie intégrante de notre vie quotidienne, ces appareils ont accès à des informations privilégiées concernant les personnes avec qui nous communiquons, les endroits où nous nous trouvons, et les façons dont nous passons notre temps. Cependant, sans outils d'analyse adéquats et la capacité de traiter de grandes quantités de données et de grands nombres d'usagers, ces données ont une valeur limitée. Cette thèse présente des outils capables d'extraire des informations concernant les activités et l'emplacement d'une personne à partir des données amassées par les capteurs intégrés dans les téléphones intelligents de consommation courante.

Nous présentons un algorithme de comparaison de séries temporelles et nous démontrons que sur un grand nombre de problèmes de référence, la performance de cet algorithme est comparable, et souvent supérieure, à celle des approches de pointe alors que l'algorithme est suffisamment efficace pour être éxécuté en temps réel sur un téléphone intelligent moyen. Nous présentons la reconnaissance des activités et de la démarche en tant qu'applications de cet algorithme, et démontrons que sa performance sur ces tâches dépasse celle des approches de pointe.

Concernant la détection de l'emplacement, nous présentons une étude approfondie des signaux wifi dans des environnements dynamiques non-stationnaires, et concluons qu'un nouveau modèle multinomial est préférable au modèle gaussien standard utilisé pour modéliser les mesures de la force des signaux wifi sur un terminal mobile. Nous présentons un algorithme de groupement pour signaux wifi qui utilise un modèle multinomial sous-jacent, et démontrons que sur un ensemble de données de grande taille, la performance de cette approche se compare favorablement à celle du modèle Gaussien standard lorsque qu'elle est confrontée à la tâche d'identifier des emplacements physiques à partir de données wifi. En basant notre approche sur le processus hiérarchique de Dirichlet, le nombre d'emplacements identifiables varie en se basant sur les données, et n'a pas besoin d'être spécifié au préalable. Ceci permet de localiser un appareil mobile en temps réel de façon précise, à l'intérieur comme à l'extérieur, d'une façon qui, contrairement aux approches existantes, protège la vie privée de l'usager.

Même les informations détaillées ont une valeur limitée si on ne dispose pas de la capacité de les communiquer à l'usager de façon significative. Nous présentons donc un cadre pour traduire les données amassées par les capteurs en descriptions des évènements qui se produisent lisibles par les êtres humains. Ce cadre est évalué sur des corpora d'une taille combinée de 150 Go composé de données amassées par les capteurs de 38 usagers sur une période de 14 mois, et il permet de visualiser de larges quantités de données ainsi que de produire des rapports exacts, en langage naturel, des évènements dont la trace est extraite des données.

Acknowledgements

I would like to start by thanking my supervisors, Professors Doina Precup and Shie Mannor, for their support and guidance over the last six years. Doina and Shie pushed me to explore my own ideas, while providing the direction that I needed to succeed. The diversity of work being done by Doina's students serves as a testament to her remarkable breadth of knowledge, and the quality of the work speaks to its depth. Doina's dedication to her students is astounding and despite how busy she is, when the office door closes it feels like my research is the most important thing in the world. Even after moving to the other side of the hemisphere, Shie remained accessible and active in his role as a supervisor. Shie's work ethic is one that I seek to emulate, and his creativity inspired much of the work in this thesis. Both served as role models, mentors, and friends throughout this journey. I have, and will continue to unabashedly brag about the quality of supervision that I received and I look forward to future collaborations with both Doina and Shie.

I thank the former and current members of the Reasoning and Learning Lab for making my time at McGill so enjoyable. I consider myself fortunate to be a part of such a diverse, creative, smart group of people, and have benefited tremendously, both personally and academically, from interactions with my labmates. In particular, Sophia, Sylvie, Jon, Kamal, Cosmin, Maja, Monica, Bert, Bob, Rob, Mahdi, Gheorghe, Pablo, Yuri, André, Keith, Susan, and Guillaume have provided for countless enlightening and entertaining discussions. I am honoured to call these people my friends. Jesse kept me well-caffeinated, and Yogesh and Ivan came along for the rocky journey that was learning Bayesian nonparametrics. The Barbados workshops were some of the highlights of my time as a grad student, and I thank Brian Tanner, Elliot Ludvig, Rich Sutton, Michael Bowling, Csaba Szepesvári, Anna Koop, and others from the UofA group who helped make those events so much fun. The Hackers hockey team has provided a much needed distraction and source of exercise over the past few years, and I thank my teammates for the camaraderie, and Mathieu, in particular, for encouraging me to take up the sport.

The faculty in the School of Computer Science (SOCS) is made up of exceptional teachers and researchers, and in particular I thank Professors Joelle Pineau, Prakash Panangaden, and Gregory Dudek for being available to provide advice and assistance when I needed it. I thank the SOCS support and administrative staff, as they have always been responsive and incredibly helpful. The system administrators, Ron Simpson, Andrew Bochego, and Kailesh Mussai have set up and maintained a great computing environment, and have been extremely patient with my never-ending requests to install new software. Diti Anastasopoulos and Sheryl Morrissey have made the administrative side of being a grad student a breeze, for which I am grateful. I thank the National Science and Research Council of Canada (NSERC), the Molson and Hart fellowship, and the School of Computer Science for supporting and funding my research.

Without data, this thesis would be a collection of ideas, presented without any reason to believe them. Some data I collected, and I am grateful for the participants in my data collection activities. Most of the data, however, was made available by others to whom I am heavily indebted. I thank Prof. Dieter Fox and his group at UW, Prof. Tanzeem Choudhury and her groups at Dartmouth and Cornell, Prof. Eamonn Keogh's group at UC Riverside, and especially Juha Laurila, Daniel Gatica-Perez and Jan Blom at Nokia Research and the participants in their LDCC data collection campaign. I am also thankful for the MSP units provided by Intel© that were used in our early data collection efforts, back when smartphones weren't as powerful.

I have an incredibly supportive family, and I thank my parents, Joe and Cyra, and my brother Shaun for shaping who I am. I am fortunate to have extended family in Montreal, all of whom have helped make this city feel like home. I thank Zaida and Roz for the breakfasts, Leon and Carol for the dinners, and Ryan, Dov, and Oliver for the movies and laughs. It has been tremendously rewarding to get to know them better over the last six years and they make leaving quite difficult. Finally, I would not have been able to do this without the love and support of Gina and Charlotte, who make this all worthwhile.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

1-NN one-nearest-neighbour classifier.

AP wifi access point.

AR(p) autoregressive models of order p.

ARIMA auto-regressive integrated moving average.

cDTW constrained dynamic time warping.

DFT discrete Fourier transform.

DTW dynamic time warping.

DWT discrete wavelet transform.

ECG electrocardiography.

FFT fast Fourier transform.

GeTeM geometric template matching.

GPS the global positioning system.

HDP hierarchical Dirichlet process.

HMM hidden Markov model.

LBS location-based service.

LDA latent Dirichlet allocation.

LPC linear predictive coding.

MCMC Markov chain Monte Carlo.

MDC Nokia© Mobile Data Challenge.

NER named entity recognition.

NLP natural language processing.

PCA principal components analysis.

RSS received signal strength.

SMT statistical machine translation.

SVM support vector machine.

VB variational Bayes.

Dedicated to Gina and Charlie

Chapter 1

Introduction

1.1 Motivation

In the last decade, we have seen the instrumentation of society at a pace unrivalled in our history. Smartphones, devices capable of accurately monitoring movement, location, communication, and information consumption, have become ubiquitous. Smart grids and intelligent power meters are capable of monitoring electricity usage in real time at a metropolitan level, and recent advances have enabled monitoring the usage of individual devices within a home. With RFID-enabled transit and transportation systems, cities can monitor and optimize resource allocation. According to Google© CEO Eric Schmidt, as of 2010, every two days we create as much information as we did from the dawn of civilization up until 2003. As of 2012, every day over half a billion people share details of their private lives on sites like Facebook© and Twitter©, billions of conversations take place through services like GMail©, and the social network graph of Facebook is approaching one billion nodes (i.e., users).

There now exist more cellphones than people on earth. A particularly poignant recent article in the New York Times, titled "That's No Phone. That's My Tracker" [4], cites a recent survey by the mobile carrier O2© which reports that making phone calls is only the fifth most frequently used feature of a mobile phone. Smartphones are privy to our communication patterns over voice, text messaging, email and instant messaging, our browsing and purchasing habits, and with the integration of GPS, accelerometers, and other sensors, accurate information about our mobility patterns and physical activities.

A new field has emerged called *computational social science* [5], where data comes not from human observation and self-reporting as is traditional in social sciences, but from sensors and sources capable of collecting data with an unprecedented breadth, depth, and scale. The *Quantified Self* movement[1] consists of thousands of members in over 50 cities around the globe, and is a community of researchers, engineers, and others interested in self monitoring and analytics for understanding behaviours from panic attacks to monitoring fitness levels, mainly through the incorporation of wearable sensors in daily activities. These new fields rely both on engineering advances, such as miniaturization of sensors and their incorporation into smartphones and clothing, and on advances in scalable information processing.

We are motivated by the technical challenge of extracting information about human behaviour from large volumes of data in a distributed fashion. A central tenet of the map-reduce framework [6], which is arguably one of the most significant advances in distributed computing of the last decade, is to move computation to where the data reside. In this spirit, we develop algorithms that can run on the same devices that collect the data. By processing the data at its source, we can lighten the computational load downstream and thus take a step towards the development of systems that can scale to population-level processing. In this thesis, we present tools for *ubiquitous machine learning*: learning algorithms that integrate with technologies that are a part of daily life.

[1]`http://quantifiedself.com/`

1.2 Contributions

The overarching theme of this thesis is the proposal of methods for modelling and analyzing data about human behaviour collected from mobile devices, namely smartphones. The emphasis is on efficiency, and the proposed approaches are designed to be viable for use on mobile devices with limited impact on battery life. All of the approaches presented in this thesis are thoroughly evaluated on real data, most of which was collected from mobile devices. Here we highlight the novel contributions in this thesis.

1.2.1 The Geometric Template Matching Algorithm

We present an efficient algorithm for classifying time series that can operate in real time on streaming data. Geometric template matching (GeTeM) is based on a nonlinear time series analysis technique called time-delay embedding which reconstructs models of dynamical systems from univariate observations. Models for segments of time series are constructed and then compared using a novel spatiotemporal nearest-neighbour algorithm to produce a similarity measure. This similarity measure is applied to time series classification tasks, time series clustering tasks, semi-supervised learning tasks, and for real-time novel activity detection in streaming data. In all cases, GeTeM is shown to outperform state of the art approaches, with lower computational cost. We also develop a boosting approach for more complex time series classification problems that uses GeTeM for feature extraction, much in the way radial basis functions are typically used. This boosting approach is used to classify subjects by their gait patterns in real time on a mobile phone, and is shown to be far more robust to environmental and clothing variations than existing approaches.

1.2.2 Wifi Location Labelling

We adapt the online hierarchical Dirichlet process (HDP) learning algorithm [7] to the streaming setting and develop a semi-supervised learning algorithm for clustering and labelling wifi data. To motivate a new model for wifi data, we first demonstrate deficiencies in existing approaches, and describe how the model we propose addresses these issues. Our approach is evaluated on over a year's worth of real wifi observations, and outperforms existing approaches for classifying and labelling locations from wifi data.

1.2.3 Generating Storylines From Sensor Data

We present a novel framework for translating between mobile sensor data and natural language descriptions of events, thus adding intelligibility to the data. By incorporating techniques from statistical machine translation, we produce human-readable transcriptions of events from data collected from a mobile phone, and demonstrate our approach on a large dataset provided by Nokia©. We also describe an approach for visualizing large datasets of this nature. Our approach is able to summarize large datasets and provide intelligibility to mobile sensor data.

1.3 Organization

The organization of this thesis is somewhat nontraditional. As the background material related to each of the three main topics are mostly disjoint, we include the necessary background material in the same chapter as the contribution itself, rather than have a single chapter solely for background material. Thus, each chapter can be read as a self-contained unit.

This thesis is structured as follows. Chapter 2 presents the GeTeM algorithm for time series classification and an evaluation of GeTeM on activity and gait classification tasks, as well as a large benchmark dataset. In Chapter 3 we describe our novel approach to location discovery and an evaluation on a large set of human mobility data. Chapter 4 describes our approach to generating storylines from sensor data and the results of our participation in the 2012 Nokia© Mobile Data Challenge (MDC). Finally, we conclude with a discussion of privacy issues, potential applications, and opportunities for future work in Chapter 5. We also include a description of our mobile data collection platform in Appendix A.

Chapter 2

Geometric Template Matching

Time series are challenging for machine learning practitioners. Traditional assumptions about data on which classification algorithms operate, namely that the data are independent, do not hold when elements of a time series are considered as individual objects. Moreover, notions of similarity and dissimilarity for time series can be difficult to define and are often problem-specific. For instance, one may be interested in comparing trends that occur in time series (e.g., a stock price rising, then falling sharply), without regard to the magnitude of the values (amplitude and offset invariance), the temporal location at which the trend starts (phase invariance), the temporal length of the trend (temporal scaling invariance), or some combination of these [8].

A typical approach for the treatment of time series is to consider windows (i.e. subsequences) of the data and treat the elements in the window as coordinates of a point in a high-dimensional space, where traditional notions of similarity such as Euclidean distance can be used. To combat the curse of dimensionality, transformations such as the discrete Fourier transform (DFT), the discrete wavelet transform (DWT), or principal components analysis (PCA), are often applied, and uninformative components or coefficients are discarded.

An alternative approach is to treat the data as a sequence of observations of a dynamical system. Methods in this category usually consider a particular class of dynamical system models and compare two time series by fitting models to the data, then comparing properties of the models. For example, Kalpakis et al. [9] fit auto-regressive integrated moving average (ARIMA) models to segments of data and compare the parameters. The intuition is that two time series should be considered similar when the underlying physical models that generated them are the same or close. The approach that we propose in this chapter follows this line of reasoning. However, we replace the parametrized linear dynamical system models that are typically used [9–11] with nonparametric nonlinear dynamical system models called time-delay embeddings. Note that the word "nonparametric" does not mean "no parameters"; rather, the complexity of the model is not fixed *a priori*, but scales with the data. Nonparametric models make fewer assumptions about the system that generated the data, and allow model complexity to scale with the data, but may sacrifice the compactness offered by parametric models. A typical example of nonparametric model is a k-nearest-neighbour classifier, where k is a parameter, and the decision boundary is defined by the training data. This is in contrast to a parametric model such as a perceptron, where the decision boundary is defined by a set of parameters (or weights) whose cardinality is fixed in advance.

Time-delay embeddings are an effective technique for creating models of periodic time series [12]. They have been used extensively in the analysis of chaotic systems [13, 14] but their use in other fields is limited. A time-delay embedding of a time series is a reconstruction of a nonlinear dynamical system that could have generated the observed data. In the idealized case in which the observations are noise-free and capture the degrees of freedom inherent in the dynamical system, time-delay embeddings can be shown to reconstruct perfectly both the state and the dynamics of arbitrary finite-dimensional deterministic dynamical systems [15]. This is of great interest as it allows assessing properties such as dimensionality or predictability (by estimating Lyapunov exponents, for example) of a latent dynamical system of arbitrarily high complexity through a sequence of univariate measurements.

Since the models are nonparametric, the process of comparing two models is more involved. The main contribution of this chapter is an algorithm called geometric template matching (GeTeM) which uses time-delay embeddings for building models from segments of time series and then computes a score that represents how similar the reconstructed dynamical systems are, in terms of their state space as well as their dynamics. GeTeM is computationally efficient and is designed to be used with resource-constrained embedded devices.

This chapter is organized as follows. In Section 2.1 we provide an overview of existing approaches for treating time series in a machine learning context, and a brief introduction to time-delay embeddings. In Section 2.2, we present the main contribution of this chapter, the geometric template matching (GeTeM) algorithm. Section 2.3 describes the use of GeTeM for clustering, classification, visualization, and semi-supervised learning, and includes an empirical evaluation with real data for each of these settings. We present a more powerful classification algorithm for long time series, called TDEBOOST, which uses GeTeM for feature extraction, in Section 2.4. TDEBOOST is evaluated on a large, challenging data set that we collected and are making available for benchmarking time series classification algorithms, particularly those suited for gait recognition.

2.1 Background

In this section we review a number of existing approaches for machine learning with time series and give a brief overview of time-delay embeddings, which form the basis of our technique.

2.1.1 Machine Learning with Time Series

We begin by giving an overview of some existing approaches for treating time series in a machine learning context. We acknowledge that this is a large field, and our selection of topics in this section is guided by the specific problem settings considered in this thesis.

We consider supervised, unsupervised and semi-supervised learning with time series data. For supervised learning, we consider time series which are associated with some finite (typically small) set of output (class) labels. Each time series can have a single class label, or each value in the time series can have a class label. The former is typically used with short time series and the latter with long time series. The goal is to produce a classifier that can label unseen data, based on a labeled training set. Performance is typically measured as the accuracy of the classifier on data on which it was not trained. When each individual time series is associated with a single class label, standard metrics for supervised learning algorithms, such as precision and recall, are used. If labels are associated with the values in the time series, other metrics can provide additional insight. Ward et al.[16] present a thorough taxonomy of performance metrics for these types of problems. For example, in physical activity detection, a single time series may contain segments of walking, running, etc., which are referred to as *events*. One may be interested only in detecting whether or not an event occurred, and may be less interested in detecting at exactly what time the event began or finished. Therefore, classifying any individual value or window (*frame*) within an event could be considered a correct labelling for the entire event, even if some of the frames are mislabelled. For our purposes, we ignore these event-based metrics, and only consider standard frame metrics (in the language of [16]), in which scores are computed based on the number of labels correctly predicted.

Unsupervised learning is concerned with finding structure in unlabelled data. For example, in clustering one tries to group the data into clusters. A good clustering is one in which the data within a cluster are more similar to each other than to data in other clusters. This definition relies on having a notion of similarity. Unlike in supervised learning, where performance is easy to measure, the performance of an unsupervised learning algorithm is more difficult to assess. One approach is to compute a score based on the distance between points within a single cluster and the distance between points in different clusters. Two examples of such scores are the Dunn index [17] and the Davies-Bouldin index [18]. Alternatively, if labeled data is available, one can ignore the labels when clustering, and assess the quality of a clustering based on whether data within each cluster have consistent class labels.

Semi-supervised learning considers the case in which training data contains both labeled and unlabelled instances. In many real-world problems, unlabelled data is plentiful, but labelling data is costly. For instance, in medical applications, one could obtain large quantities of x-ray images, but labelling the images would require medical doctors or experts in the field. Semi-supervised algorithms exploit the structure of the unlabelled training data to obtain classifiers that perform better than ones trained only on the labeled portion of the data [19].

2.1.1.1 Types of Time Series Data

In this work, we consider univariate, real-valued time series, of the type generated by measurements of some latent dynamical system. Before we discuss methods for processing such data, it is important to note the distinction between *time series* and *sequential data*. In both cases, the data consists of a sequence, or list of values, in which the order is important. Time series is a sub-class of sequential data where the longitudinal component corresponds to time. Not all data sets that are considered “time series” benchmarks in the literature have this property. For instance, a number of the data sets in the UCR Time Series Classification Database [1] consist of sequential data that are not time series. An example is the

Swedish Leaf data set [20] where each object is generated by taking a static image of a leaf, choosing a starting point, tracing the outline, and measuring the distance from the centroid of the leaf. This gives a sequence of values, which the authors call a “pseudo time series”; however, the temporal aspect in this case is artificial.

We emphasize this distinction because for many instances of sequential data, the assumptions that underlie dynamical systems-based approaches for the classification of time series are not satisfied. In these situations, pattern-matching approaches are more appropriate, as the data is typically well-segmented and well-aligned. Surprisingly though, as we will see later in this chapter, approaches based on fitting dynamical systems often outperform pattern-matching approaches even on these types of data.

Time series also come in different flavours. Most important for our work is the distinction between segmented (short) time series, and long time series; techniques developed for the former often require special care when applied to the latter, as we will see in the following sections. There are a large number of techniques for analyzing time series, and the list of techniques we describe in the following sections is not meant to be exhaustive. We have selected techniques that are commonly used and for which published results exist on the data sets that we use to evaluate our proposed approach.

2.1.1.2 Segmented Time Series

In the time series classification, clustering, and indexing literature, most work focuses on applications in which there is some set of patterns of interest, or motifs, in the data. The data is segmented such that some fixed number (typically 1) of repetitions of a motif occur in each segment. For example, an electrocardiography (ECG) data set might contain 100 segments where each segment represents one period of the heart’s electrical cycle. In classification, each class is associated with a particular motif, and the goal is to determine which motif occurs in each segment.

For clustering, the goal is typically to partition the data such that segments within a partition have motifs that are similar. In the indexing problem, the goal is to compile a set of data into a data structure such that, given a new segment, segments that most closely resemble it can quickly be retrieved. In all these applications, the segments can be treated as individual objects.

The simplest approach for comparing segments of equal length is to treat the n elements of the segment as coordinates of points in $\mathbb{R}^n$, then compute some traditional measure of distance, typically an L_p norm such as the Euclidean distance, or autocorrelation. Normalizing the magnitude of the values in the data typically improves the performance of this approach.

When data are not well-aligned, or the segments are of unequal length, dynamic time warping (DTW) [21] can be used to align the segments before computing the distance between them. DTW stretches sequences in the temporal dimension in order to minimize some measure of distance (typically Euclidean) between them. Given two time series $x = x_1, \ldots, x_n$ and $y = y_1, \ldots, y_m$ (WLOG assume $m \leq n$), a warping path is a sequence W of tuples $w_k = (i, j)$ where the first element indexes an item in x and the second an item in y. It is assumed that the sequence does not contain duplicate elements, and we use the notation $w_k^{(i)}$ and $w_k^{(j)}$ to refer to elements i and j of the tuple w_k, respectively. For a warping path W to be valid, every index in $1, \ldots, n$ must appear in the first position of at least one element of W, and every index in $1, \ldots, m$ must appear in the second position of at least one element of W. Additionally, a valid warping path W must contain the elements $(1, 1)$ and (n, m), and must satisfy a monotonicity condition: for all pairs of elements $w_k = (i, j)$ and $w_k' = (i', j')$ in W, if $k \leq k'$ then $i \leq i'$ and $j \leq j'$. A valid warping path contains K elements, where $n \leq K \leq n + m - 1$.

Given a valid K-length warping path W, let

$$\begin{aligned} x' &= x_{w_1^{(i)}}, x_{w_2^{(i)}}, \ldots, x_{w_K^{(i)}}, \quad \text{and} \\ y' &= y_{w_1^{(j)}}, y_{w_2^{(i)}}, \ldots, y_{w_K^{(i)}}, \end{aligned}$$

denote the aligned sequences. The cost of warping path W using distance function d is $d(x', y')$, and the DTW cost, or distance, is the minimum cost over all valid warping paths.

To improve the performance in practice, a locality constraint can be added such that for all elements $w_k = (i, j)$ in W, $|i - j| \leq r$ for some $r \geq n - m$. Constrained dynamic time warping (cDTW) is computationally more efficient than DTW, and has been shown to outperform DTW when used as a distance measure for certain classification problems [22]. Note that if $m = n$ and the locality constraint is used with $r = 0$, the DTW distance is equal to the distance between the unaligned segments. If $m = n$, then the identity map, $w_k = (k, k)$ for $k = 1, \ldots, n$ is a valid warping path, and therefore the unaligned distance is an upper bound on the DTW distance. Computing the DTW distance can be done using dynamic programming at a computational cost of $O(nm)$ (where n and m are the lengths of the two sequences), or $O(rn)$ if the locality constraint with threshold r is used.

Time series segments often contain hundreds or thousands of measurements. The approaches reviewed so far involve distances in high dimensional spaces, and therefore often suffer from the "curse of dimensionality". It is well known that nearest-neighbour classification in high dimensions is wrought with pitfalls, from both a computational perspective (data structures that work well in low dimensions do not offer the same computational advantages in high dimensions), and in terms of the quality of their results (the difference in distance between one's nearest and farthest neighbours in high dimensions is often small) [23]. To address these problems, the dimensionality of the sequences is often reduced. Two approaches that are well-suited for time series data are the DFT[1] (DFT) [24, 25] and the discrete Wavelet transform (DWT) [26]; traditional dimensionality reduction techniques such as principal component analysis (PCA) [27] can also be used. All three of these approaches operate by projecting the data into an alternative basis, then discarding dimensions that are are deemed less informative. The DFT decomposes a signal into its constituent frequencies and represents it as coefficients of a linear combination of sinusoids. The DWT is similar to the DFT, but augments the frequency information with temporal locality information by using a basis generated by a wavelet function. Commonly used wavelet functions include the Haar wavelet and the Daubechies wavelet [28]. Unlike the DFT and DWT, for which the bases are predetermined, PCA finds the orthonormal basis such that the greatest variance by any projection of the data lies on the first axis, the second greatest variance on the second axis, and so on. Dimensionality reduction involving the DFT and the DWT typically assume that high-frequency components are more likely to account for noise, and can be discarded without losing information about the signal. In the case of PCA, the assumption is that directions on which the data have low variance can be discarded.

The approaches mentioned so far all attempt to match shapes or motifs by looking at the raw data, or some projection thereof. We will refer to these types of techniques as *pattern matching*. An alternative viewpoint is to consider the raw data to be observations from some underlying dynamical system, and to compare properties of the systems that generated the different sequences of data. These approaches operate by choosing some class of dynamical systems, typically parametrized by some finite set of parameters, fitting a model to each sequence, and then computing a measure of distance based on the parameters of the fitted models.

One popular family of dynamical system models are autoregressive models of order p (AR(p)) [29], where each value in the sequence is assumed to be a linear function

[1]The DFT is often called the fast Fourier transform (FFT), which simply refers to the algorithm commonly used to calculate the DFT.

of some number p of previous elements:

$$X_t = \theta_0 + \sum_{i=1}^{p} \theta_i X_{t-i} + \varepsilon_t,$$

where $\theta_0, \theta_1, \ldots, \theta_p$ are the parameters of the model, and ε_t is zero-mean Gaussian noise.

Dynamical system models are often used for data generated by periodic dynamical systems, such as measurements of cardiac or pulmonary systems. For example, Ge et al. [10] modelled ECG data with an AR(p) model, then used the parameters in a linear classifier and achieved over 92% accuracy in identifying cardiac disorders. Kalpakis et al. [9] used the Euclidean distance between the linear predictive coding (LPC) cepstrum, which is a function of the AR(p) parameters. This distance measure was used to classify a number of data sets, including ECG data, per capita personal income, temperature readings, and population levels. The performance of the LPC cepstrum was competitive with DFT, DWT, and PCA-based approaches.

Another popular dynamical system model is the hidden Markov model (HMM) [30], which assumes an underlying Markov process over a set of hidden states. Each state has a distribution over output tokens that are emitted upon visiting the state. The sequence of tokens forms the time series, while the underlying state sequence is unobserved. The state-to-state transitions and the emission probabilities that were most likely to have generated observed data can be learned using the Baum-Welch algorithm [31].

To perform time series classification, one can learn an HMM for each class from training data. For new data, each HMM can be used to estimate the likelihood of the observations; the class associated with the HMM giving the highest likelihood is then picked. Similarly, for clustering, each cluster has an associated HMM and the likelihood is used as a similarity measure to associate sequences with the clusters [32]. Alternatively, one can learn an HMM for each sequence, and compute a measure of similarity between the HMM parameters, such as the Kullback-Leibler divergence between the distributions [33].

2.1.1.3 Long Time Series

In the real world, time series are likely to arise as the result of some measurement apparatus, such as a wearable sensor, an ECG recording device, or a stock price index. For these types of data, there is often no natural notion of a segment. Even if segments exist, the data may be too noisy for automatic segmentation; manual segmentation on long sequences is also very tedious. To avoid segmentation, a common approach is to use a sliding window of data and extract overlapping segments. By increasing the amount of overlap, one increases the chances that a segment will contain a pattern of interest, which might otherwise be split among neighbouring segments. However, increasing the size of the overlap increases the number of segments considered. The size of the window greatly affects the performance of this approach. If the window is too small, longer patterns of interest will be missed. If the window is too large, dimensionality reduction techniques are required, which may discard properties of interest (such as high-frequency components). A standard approach is to take multiple passes over the data with different window sizes. Segments produced by a sliding window can then be compared using the approaches for segmented data described in the previous section.

While the goal in the supervised learning setting with segmented data is to assign a label to each segment, with long time series, the goal is typically to assign a label to each element in the time series. With a sliding window, each data element will be present in multiple segments; typically, each segment is labeled using one of the methods described in the previous section, and then each element of the time series is assigned the majority label over the segments that contain it.

One example application of long time series is classifying primitive physical activities from data provided by wearable sensors, such as accelerometers. Table 2.1

Feature	**Description**
Cepstral Coefficients	The FFT of the log FFT spectrum, that is FFT(log(FFT(x)))
Log FFT Frequency Bands	Real valued FFT values grouped into logarithmic bands
Spectral Entropy	Measure of the distribution of frequency components
Energy	The sum of the real FFT spectrum
Mean	The average value of the time series
Variance	The square of the standard deviation
Linear FFT Frequency Bands	Real valued FFT values grouped into linear bands from 100Hz - 2kHz
Correlation Coeffs	Correlation between axis pair, XY, XZ, YZ
Integration	Integration of the time series over a window

TABLE 2.1: Features typically used for classification of physical activities. Reproduced, with permission, from Lester et al. [3].

(reproduced with permission), gives an example of a set of features used by a state-of-the-art activity recognition system [3]. Nearly all existing activity recognition systems use some subset of these features. For example, the activity recognition system of Bao and Intille [34] uses mean, energy, spectral entropy and correlation features, while the experiments of Heinze et al. [35] considered mean, integration and variance features. Ravi, Mysore, and Littman [36] extracted mean, standard deviation, energy and correlation features, and Huynh and Schiele [37] analyzed mean, variance, energy, spectral entropy, correlation coefficients and log FFT frequency bands.

2.1.2 Time-Delay Embedding

We now present a brief overview of time-delay embedding, the main approach on which we build; we refer the reader to the standard text by Kantz and Schreiber [13] for further details. For notation, we use lowercase letters, such as x and τ, to denote integer-value variables and sequences; boldface letters, such as $\boldsymbol{u}$, represent vectors; lists of values surrounded by angle-brackets, such as $\langle x_1, \ldots, x_k \rangle$ or equivalently $\langle x_i \rangle_{i=1}^{k}$, describe the elements of a sequence, where the order is important; and uppercase letters, such as N, denote integer-valued constants.

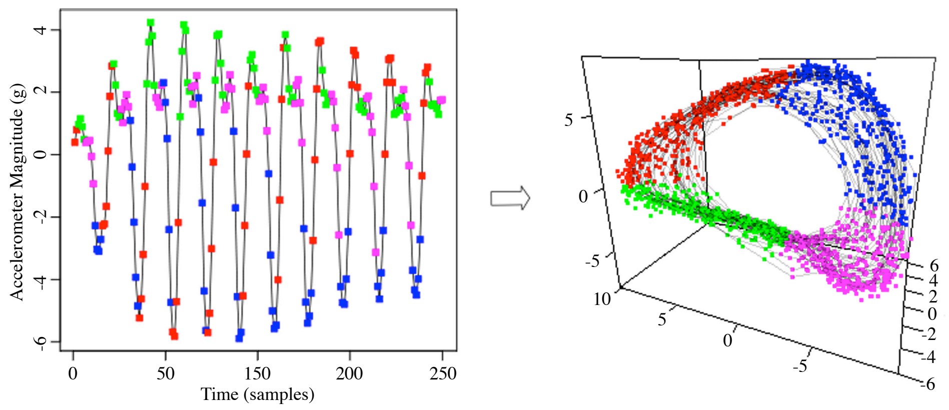


FIGURE 2.1: Example of time-delay embedding for accelerometer data from biking activity. Left: raw accelerometer data. Right: time-delay embedding with $m = 3, \tau = 11$. The coloured points in the left diagram are mapped to points of the same colour in the embedding.

The purpose of time-delay embeddings is to reconstruct the state and dynamics of an unknown stationary, deterministic dynamical system from measurements or observations of that system taken over time. The state of the dynamical system at time t, $x_t \in \mathbb{R}^k$, contains all the information necessary to compute the future of the system at all times following t (determinism), and the dynamics is assumed to not change over time (stationarity). The set of all states is called *phase space*. The state cannot be measured directly, and even the true dimension of the phase space k may not be known. However, a time series $o = \langle o_t = \omega(x_t) \rangle$ can be obtained by an unknown measurement function $\omega : \mathbb{R}^k \to \mathbb{R}$, which is a smooth[2] map from the phase space to a scalar value. While individual measurements do not provide a complete description of the system (with the exception of one-dimensional systems), sequences of successive measurements do. Intuitively, the sequence $\langle o_t, o_{t-\delta_1}, \ldots, o_{t-\delta_{m-1}} \rangle$, referred to as *delay coordinates*, for sufficiently large m and sufficiently small δ_i, contains information about the derivatives up to order $m - 1$. Therefore, if the dimension of the system is less than m, one should have enough information to describe the evolution as a system of $m - 1$ differential equations.

[2]Here, and throughout this chapter we use "smooth" to mean continuously differentiable.

The key problem is to determine how many measurements m have to be considered, and at what times they should be taken (i.e., what are the values of δ_i), in order to capture well the system dynamics. In particular, the map from $\mathbb{R}^k$ to $\mathbb{R}^m$, encapsulating the measurement and subsequent projection to delay coordinates, should be one-to-one and should preserve local structure (i.e., the dynamics). If this were the case, by looking at the time series in $\mathbb{R}^m$ (which is called the *reconstruction space*), one would essentially have all the information from the true system state and dynamics.

The true state of the system x_t is assumed to lie on an attractor[3] $A \subset \mathbb{R}^k$ in an unknown phase space of dimension k. An *embedding* is a map from A into the reconstruction space $\mathbb{R}^m$ that is one-to-one and preserves local structure. In differential topology, such a map is called a local diffeomorphism on A.

Let F be a function of a sequence o of T measurements, a time index t, and time-delay parameters m and τ, defined as:

$$F(o, m, \tau, t) = \langle o_t, o_{t+\tau}, o_{t+2\tau}, \ldots, o_{t+(m-1)\tau} \rangle. \tag{2.1}$$

The function F represents the state of the system at time t by an m-length sequence of τ-lagged measurements, and is referred to as the time-delay reconstruction function.

Takens showed that if A is a d-dimensional smooth compact manifold, then provided $m > 2d$, $\tau > 0$, and the attractor contains no periodic orbits of length τ or 2τ and only finitely many periodic orbits of length $3\tau, 4\tau, \ldots, m\tau$, then for almost every smooth function ω, the map from $\mathbb{R}^k$ to the time-delay reconstruction (i.e., the measurement function ω composed with the time-delay reconstruction function F as defined in Equation 2.1) is an embedding [15]. In other words, any

[3]We use the term "attractor", as in dynamical system theory, to mean a closed subset of states towards which the system will evolve from many different initial conditions; once the state reaches the attractor, it will typically remain inside it indefinitely (in the absence of external perturbations).

time-delay reconstruction will accurately preserve the dynamics of the underlying dynamical system, provided m is large enough and τ does not conflict with any periodic orbits in the system. However, from a practical standpoint, m should be as small as possible, as it controls the dimensionality of the reconstruction as well as the size of the time window considered for each point in the reconstruction. If the data is sampled at a rate of r measurements per second, then the time window captured by each point is $((m-1)\tau+1)/r$ seconds.

Takens' theorem only holds when the measurement function ω is deterministic, but in practice measurements are corrupted by noise from a variety of sources, ranging from the measurement apparatus to rounding errors due to the finite precision of a computer. In the presence of noise, the choice of m and τ becomes very important. There are a number of techniques for choosing good values for m and τ based on geometrical and spectral properties of the data [38, 39]. In practice, since computing the reconstruction for a given parameter setting requires little computation, it is feasible to perform a grid search over a reasonable parameter space and cross-validation, using a problem-specific performance criterion. Empirically, we have found that for a choice of τ that does not conflict with periodic events in the time series, the quality of the embedding is robust to the choice of m (see Section 2.3.1, for example). Choosing a small value for τ will make it less likely that there is a conflict, but for smaller values of τ, the effects of noise substantially increase with the value of m. Therefore, we prefer larger values of τ, which make the quality of the embedding more robust to the choice of m.

Figure 2.1 illustrates the concept of time-delay embedding using a segment of data captured from an accelerometer placed on the hip of a subject while riding a stationary bicycle. Clearly, the system evolves along a set of tightly clustered trajectories, in a periodic fashion. This is quite intuitive, considering that cycling involves a periodic, roughly two-dimensional leg movement. However, it is very nice that this structure can be reconstructed automatically from the time series on the left.

Methods developed for nonlinear time series analysis have not been widely used in machine learning, but are quite popular in other areas of research, especially in the study of electrophysiological time series data [40, and references therein]. By estimating the dimensionality of the underlying attractors captured in electroencephalographic (EEG) and intracranial electroencephalogram (IEEG) data, researchers have been able to characterize various changes in the dynamics of the brain, such as different stages of sleep [41], and epileptic seizures [42]. More recently, time-delay embeddings have been used to model epileptic seizures in order to learn effective treatment regimes [43].

2.2 Geometric Template Matching

Given a time series o consisting of T univariate measurements, the *time-delay reconstruction* of o is given by the following sequence of points in $\mathbb{R}^m$:

$$u = \langle \mathbf{u}_t = F(o, m, \tau, t) \rangle_{t=1}^{M},$$

where $M = T - (m-1)\tau$. We refer to u as the *model* for the sequence o. The points u_t represent the reconstructed states, while vectors connecting successive pairs of points, u_i and u_{i+1} represent the dynamics. GeTeM is an approach for comparing two models in terms of both their states and dynamics[4]. To gain an intuition as to how well such a method may work, consider three examples of embedding results from noisy accelerometer data for three different activities (walking, running and biking), presented in Figure 2.2. All are embedded in a 3-dimensional space. By visual inspection, the three manifolds look rather different; intuitively, a distance measure on the state and dynamics of these models should be able to distinguish between them easily. Note that walking and running appear more similar than biking, as one would expect.

[4]Reference implementation available at `https://github.com/jwf/tdetools`

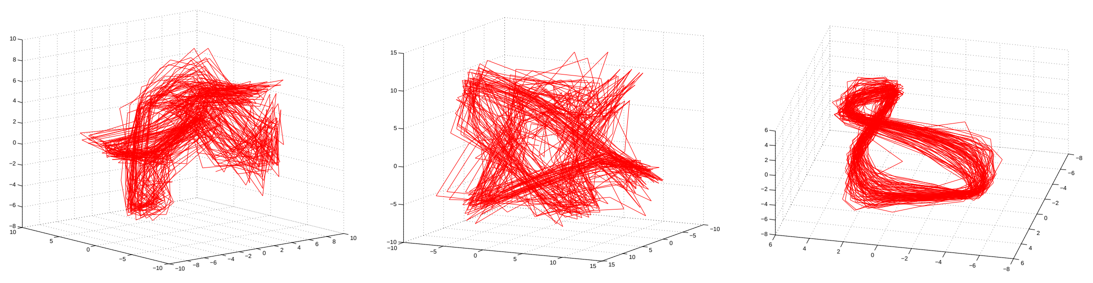


FIGURE 2.2: Time-delay embedding of three example activities for walking (left), running (middle) and biking (right), $m = 3, \tau = 4$.

To compare $o = \langle o_i \rangle_{i=1}^{N}$ with another time series $o' = \langle o'_i \rangle_{i=1}^{N'}$, GeTeM starts by building models $u = \langle \mathbf{u}_i \rangle_{i=1}^{M}$ and $u' = \langle \mathbf{u}'_i \rangle_{i=1}^{M'}$ for o and o' respectively, using the same time-delay parameters m and τ. Let $n_{i,1}, \dots, n_{i,k}$ be the indices in u of the k nearest neighbours of $\mathbf{u}'_i$ in terms of Euclidean distance. That is,

$$\begin{aligned} n_{i,1} &= \operatorname*{argmin}_{j} ||\mathbf{u}'_i - \mathbf{u}_j||, \\ n_{i,2} &= \operatorname*{argmin}_{j \neq n_{i,1}} ||\mathbf{u}'_i - \mathbf{u}_j||, \text{ etc.} \end{aligned}$$

Let $\bar{\mathbf{u}}_i = k^{-1} \sum_{j=1}^{k} \mathbf{u}_{n_{i,j}}$ and $\bar{\mathbf{u}}_{i+1} = k^{-1} \sum_{j=1}^{k} \mathbf{u}_{1+n_{i,j}}$ be the mean of the k nearest neighbours of $\mathbf{u}'_i$ in u and the mean of the subsequent points[5] in u, respectively.

The similarity between models u and u' is defined as

$$S(u, u') = \frac{1}{M' - 1} \sum_{i=1}^{M'-1} \frac{(\bar{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_i) \cdot (\mathbf{u}'_{i+1} - \mathbf{u}'_i)}{\max(|\bar{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_i|, |\mathbf{u}'_{i+1} - \mathbf{u}'_i|)^2},$$

where $|\mathbf{x}|$ denotes the vector norm and $\cdot$ the vector dot-product. Each term in the sum computes the cosine between a vector formed by subsequent points in u' and the "average" of nearby vectors formed by subsequent points in u, multiplied by the ratio of the length of the smaller of the two vectors to the length of the larger of the two vectors. Therefore, $-1 \leq S(u, u') \leq 1$. Note that the similarity scores are not symmetric and unless $k = 1$, in general $S(u, u) < 1$ due to the effect

[5]It is important to note that these points are not necessarily the nearest neighbours of $\mathbf{u}'_{i+1}$.

of averaging over nearest neighbours. To convert this measure of similarity to a measure of distance, we define

$$d(u, u') = \exp(-S(u, u')). \tag{2.2}$$

We could also make this measure symmetric by letting

$$d(u, u') = \exp(-(S(u, u') + S(u', u))/2),$$

but we have found that this makes little difference in practice, and is not worth the extra computational cost.

GeTeM has a number of attractive properties. Most importantly, by matching nearest neighbours in the reconstruction space, GeTeM allows for the comparison of unaligned segments. It does this in a manner that is quite different from DTW, which stretches the segments; GeTeM finds the closest match between the state and dynamics of an underlying model, and compares those. Unless the data are simply out of phase, DTW will perform a nonlinear warping, which may perturb useful characteristics of the data. As evidence that this warping can distort important properties of the data, it has been shown that on many real data sets, constrained DTW, which limits the amount of distortion, outperforms standard DTW [22] (i.e., the optimal DTW alignment often distorts the data so pairs of segments appear more similar than they really are). GeTeM, on the other hand, does not distort the data. Two segments are similar, according to GeTeM, if the trajectories they follow through a nonlinear dynamical system have a similar shape, regardless of the starting and ending points of the trajectories.

For similar reasons, GeTeM is well-suited for comparing segments of different lengths. If two segments contain different numbers of repetitions of the pattern of interest, then DTW will perform poorly, as it tries to globally align the segments.

Since GeTeM only matches short subsegments, it is suitable for this type of situation. Techniques based on DFT and DWT also allow for comparison of unaligned segments of differing lengths, but both are parametric, making strong assumptions on the structure of the data.

Additionally, GeTeM is computationally efficient. For segments of lengths m and n, GeTeM requires $O(n \log m + m \log m)$ time. The $m \log m$ term is the time required to build a structure suitable for nearest-neighbour queries (e.g., a kd-tree), and $n \log m$ is required for the nearest-neighbour queries. If the lengths of the two sequences differ, we let m be the length of the shorter sequence (i.e., let the shorter sequence be o, and the longer o'). Many practical applications involve finding occurrences of a short segment within a long time series, in which case the time to build the nearest-neighbour data structure on the short segment is negligible, and the running time is dominated by the $n \log m$ term. Furthermore, if we wish to perform repeated comparisons, for instance when computing pairwise distances between points for clustering, we only need to build the nearest-neighbour structures for each segment once, and can reuse them for future comparisons. Constrained DTW with a locality constraint of length r requires $O(rn)$ time, and in practice r depends linearly on m, making it effectively $O(nm)$.

In practice, approaches such as approximate nearest neighbour [44] can be used to speed up GeTeM. For repeated comparisons, the nearest-neighbour data structures can be cached. It has also been shown (empirically) that for large data sets, the cost of DTW amortized over the data set is closer to $O(n)$ [20]. For DTW, lower bounding techniques [45] can greatly improve performance. Hence, in practice, both techniques can be quite efficient, and, for large data sets, both algorithms are easily parallelizable.

2.3 Empirical Evaluation

In order to analyze the utility of the GeTeM-based distance measure for time series analysis (both segmented and long time series), we consider six different applications. First, we use GeTeM to cluster an ECG data set from the PhysioNet ECG database [46], which has been analyzed previously by Kalpakis et al. [9]. Second, we use the GeTeM distance with a one-nearest-neighbour classifier (1-NN) to classify all of the available data sets from the UCR segmented time series database [1][6]. Third, we consider the task of physical activity recognition from wearable sensors, comparing features generated from a time-delay embedding of the data with features typically used for activity recognition. Fourth, we consider the problem of gait recognition, or identifying users based on how they walk, using sensor data collected from a standard mobile phone. Fifth, we demonstrate the use of GeTeM for clustering and visualization unlabelled data collected from people performing different exercise routines. Finally, we demonstrate that given only sparsely labeled training data, clusters found using GeTeM perform well in a semi-supervised classification algorithm. We conclude this section with a discussion of the results and the efficiency of the algorithms that were considered.

2.3.1 Clustering ECG Data

To demonstrate the use of GeTeM for clustering periodic time series, we used a data set composed of 40 two-minute ECG time series from the PhysioNet ECG database, which has previously been used as a clustering benchmark by Kalpakis et al. [9]. The authors used several distance measures and performed hierarchical clustering with the goal of separating the 18 traces that correspond to normal sinus rhythm from the 22 traces taken from patients with malignant ventricular arrhythmia. They considered distance measures based on differences in coefficients from DFT,

[6]Data was retrieved on September 30, 2011

DWT, PCA, and their own approach based on Linear Predictive Coding (LPC) cepstrum. Like GeTeM, the LPC cepstral distance involves fitting a dynamical model to the data, but unlike GeTeM it uses linear modelling. The best reported performance is achieved using the DFT and autocorrelation coefficients, which grouped 3 malignant traces with the normal traces, and 1 normal trace with the malignant traces. The authors justified the mislabelling by examining the raw data and concluding that the malignant traces qualitatively "look more similar to the normal time series than to the malignant arrhythmia time series."

We constructed one time-delay model for each of the 40 traces using parameters $m = 7$ and $\tau = 11$. These parameters were chosen using the false nearest neighbours technique [13]. We computed the similarity matrix S using GeTeM and converted it to a distance matrix according to Equation 2.2. In the reported results, we only consider the lower-triangular portion of D, that is the distance between points with indices i and j was $D_{i,j}$ if $i > j$, or $D_{j,i}$ otherwise. Symmetrization, as described in Section 2.2, of D leads to almost identical results, and therefore did not justify the extra computational cost. Finally, we performed hierarchical clustering [47] with Ward's minimum-variance criteria for cluster merging [48]. The resulting cluster dendrogram, shown in Figure 2.3, contains two clearly separated clusters, one containing only malignant samples (M) and the other containing only normal samples (N). The time-delay embedding is able to tease out the underlying dynamical systems, and the distances produced by GeTeM enable a standard hierarchical clustering algorithm to clearly separate the data into normal and malignant traces, despite their visual similarity. We also performed the same hierarchical clustering using DTW with Euclidean distance, as described in Section 2.1.1.2. This approach found a cluster with 9 of the 22 malignant traces, and placed all of the remaining traces in the other cluster.

Note that the approaches considered by Kalpakis et al. [9] required a non-trivial amount of preprocessing: removing trends, normalization, etc. Our approach works with the raw data and the clustering takes approximately 60 seconds of

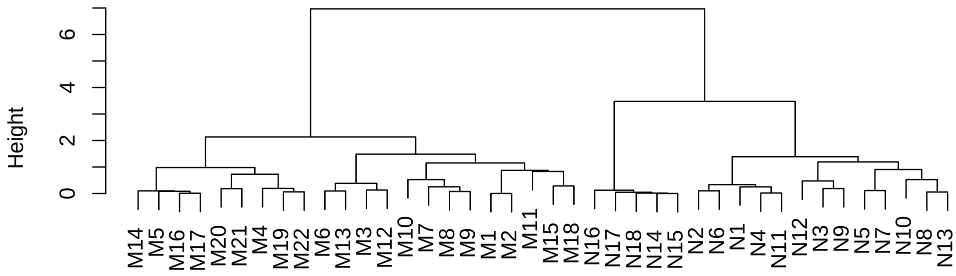


FIGURE 2.3: Clustering dendrogram for ECG data using GeTeM. Labels correspond to traces in the ECG data set. We clearly see two distinct clusters, one containing only malignant (M) traces and one containing only normal (N) traces.

computation on a 3.0GHz Intel Xeon Quad-Core desktop computer with 2 GB RAM. To demonstrate the robustness of GeTeM to the time-delay embedding parameters m and τ, we performed the clustering for a number of different settings. For all of the 20 possible combinations of $m \in \{4, \dots, 8\}$ and $\tau \in \{3, \dots, 6\}$, a perfect clustering was found. Expanding the range of parameters to the 88 possible combinations of $m \in \{3, \dots, 10\}$ and $\tau \in \{3, \dots, 13\}$, 39 of the parameter settings resulted in a perfect clustering (with the poorest clusterings occurring towards the extreme ranges of parameter settings, namely for small τ and large m), and 75 of the combinations found clusterings that were at least as good (in terms of accuracy) as the best clustering found by any of the methods in Kalpakis et al. [9].

2.3.2 Nearest-Neighbour Classification

Next, we assess the performance of the GeTeM-based distance (Equation 2.2) in a 1-NN classifier for segmented data. We use the UCR time series database [1], which contains 43 sequential data sets ranging in length from 24 to 1882 samples, with most falling in the 60 to 600 range. The UCR database contains classification accuracies for each data set using 1-NN with Euclidean distance, 1-NN with DTW, and 1-NN with cDTW with the locality constraint r tuned by cross-validation [1]. All of these results are computed on a single training and testing split provided in

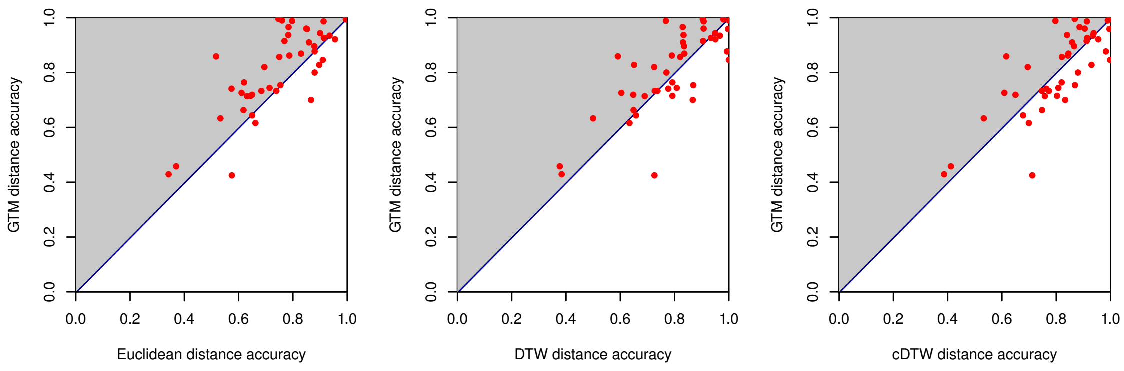


FIGURE 2.4: Performance of GeTeM distance for 1-NN, compared to the Euclidean distance, DTW distance, and constrained DTW with tuned locality constraint, from left to right, respectively, on the 43 data sets in the UCR time series database [1]. The shaded area represents the region in which GeTeM is outperforming the other algorithm.

the database. To facilitate comparison with the published results, we also present our results on these same splits.

We perform 1-NN classification using the GeTeM distance measure. We use leave-one-out cross validation on the training set to find optimal time-delay parameters m and τ, and the number of neighbours k for computing the similarity measure. Figure 2.4 depicts the performance of GeTeM compared to Euclidean distance, DTW distance, and constrained DTW with the optimal locality constraint tuned on the training set, as described in Section 2.1.1.2. The full results are in Table 2.2. On 19 of the 43 data sets, GeTeM has higher accuracy than any of the other approaches. The 1-NN with Euclidean distance approach performs best on 3 of the data sets, the tuned constrained DTW approach performs best on 15 of the data sets, and the standard DTW approach performs best on 12 of the data sets. Since results are computed on a single split of the data, and in many cases the error rates within 1-5%, these values should not be interpreted as saying that any one algorithm is significantly better than the others. Comparisons of this nature require that the results be reliable estimates of the algorithms' performance on each data set, and values computed on a single training and testing split, as given in the UCR database, do not meet this criteria.

Name	**Number of Classes**	**Size of Training Set**	**Size of Testing Set**	**Time Series Length**	**1-NN Euclidean**	**1-NN cDTW (r)**	**1-NN DTW**	**1-NN GeTeM (m,τ,k)**
50Words	50	450	455	270	0.369	**0.242 (6)**	0.310	0.286 (16,7,3)
Adiac	37	390	391	176	0.389	0.391 (3)	0.396	**0.274 (6,1,2)**
Beef	5	30	30	470	0.467	0.467 (0)	0.5	**0.367 (10,16,3)**
CBF	3	30	900	128	0.148	0.004 (11)	**0.003**	0.0411 (3,13,1)
Chlorine Conc.	3	467	3840	166	0.35	0.35 (0)	0.352	**0.281 (12,13,1)**
CinC ECG torso	4	40	1380	1639	0.103	**0.07 (1)**	0.349	0.172 (16,7,3)
Coffee	2	28	28	286	0.25	0.179 (3)	0.179	**0.143 (12,16,1)**
Cricket X	12	390	390	300	0.426	0.236 (7)	**0.223**	0.259 (15,1,1)
Cricket Y	12	390	390	300	0.356	**0.197 (17)**	0.208	0.285 (15,1,1)
Cricket Z	12	390	390	300	0.38	**0.18 (7)**	0.208	0.236 (15,1,1)
Diatom Sz. Red.	4	16	306	345	0.065	0.065 (0)	**0.033**	0.0654 (4,13,3)
ECG 200	2	100	100	96	**0.12**	**0.12 (0)**	0.23	0.2 (4,16,3)
ECG Five Days	2	23	861	136	0.203	0.203 (0)	0.232	**0.0116 (7,13,3)**
Face (all)	14	560	1690	131	0.286	**0.192 (3)**	**0.192**	0.256 (9,4,1)
Face (four)	4	24	88	350	0.216	0.114 (2)	0.170	**0.0341 (7,10,3)**
FacesUCR	14	200	2050	131	0.231	0.088 (12)	0.0951	**0.0849 (7,7,2)**
Fish	7	175	175	463	0.217	0.16 (4)	0.167	**0.063 (7,4,3)**
Gun-Point	2	50	150	150	0.087	0.087 (0)	0.093	**0.0133 (4,10,3)**
Haptics	5	155	308	1092	0.63	0.588 (2)	0.623	**0.542 (16,16,5)**
Inline Skate	7	100	550	1882	0.658	0.613 (14)	0.616	**0.571 (15,1,1)**
Italy Pwr. Dem.	2	67	1029	24	**0.045**	**0.045 (0)**	0.0496	0.0787 (6,1,1)
Lightning-2	2	60	61	637	0.246	**0.131 (6)**	**0.131**	0.246 (3,1,1)
Lightning-7	7	70	73	319	0.425	0.288 (5)	**0.274**	0.575 (15,1,1)
MALLAT	8	55	2345	1024	0.086	0.086 (0)	**0.066**	0.0738 (15,13,1)
Medical Images	10	381	760	99	0.316	**0.253 (20)**	0.263	0.267 (15,1,1)
MoteStrain	2	20	1252	84	0.121	0.134 (1)	0.165	**0.104 (3,1,1)**
OliveOil	4	30	30	570	0.133	0.167 (1)	**0.133**	0.3 (7,13,3)
OSU Leaf	6	200	242	427	0.483	0.384 (7)	0.409	**0.141 (10,7,4)**
Sony AIBO Surf	2	20	601	70	0.305	0.305 (0)	0.275	**0.18 (6,1,1)**
Sony AIBO Surf2	2	27	953	65	0.141	0.141 (0)	0.169	**0.09 (6,4,1)**
Star Light Crvs.	3	1000	8236	1024	0.151	0.095 (16)	0.093	**0.0397 (12,10,1)**
Swedish Leaf	15	500	625	128	0.213	0.157 (2)	0.210	**0.138 (9,1,1)**
Symbols	6	25	995	398	0.1	0.062 (8)	**0.0503**	0.0563 (4,10,3)
Synthetic Control	6	300	300	60	0.12	0.017 (6)	**0.007**	0.123 (3,13,1)
Trace	4	100	100	275	0.24	0.01 (3)	**0**	0.01 (7,13,1)
Two Patterns	4	1000	4000	128	0.09	0.0015 (4)	**0**	0.154 (6,13,1)
TwoLeadECG	2	23	1139	82	0.253	0.132 (5)	0.096	**0.00439 (3,13,1)**
uWave X	8	896	3582	315	0.261	**0.227 (4)**	0.273	0.267 (15,16,4)
uWave Y	8	896	3582	315	0.338	**0.301 (4)**	0.366	0.384 (15,16,3)
uWave Z	8	896	3582	315	0.35	**0.322 (6)**	0.342	0.356 (12,10,2)
Wafer	2	1000	6174	152	**0.005**	**0.005 (1)**	0.020	0.00584 (9,7,1)
Word Synonyms	25	267	638	270	0.382	**0.252 (8)**	0.351	0.337 (9,13,3)
Yoga	2	300	3000	426	0.170	0.155 (2)	0.164	**0.131 (16,4,5)**

TABLE 2.2: UCR data set results [1]. Shaded cells indicate lowest error rate.

To address this shortcoming, we also evaluated 1-NN using GeTeM, Euclidean, and DTW distances on each data set using 10-fold cross-validation. By performing a more thorough evaluation of the performance of each of these algorithms, we are able to make a stronger statement about the performance of the algorithms, under the reasonable assumption that the data sets are independent[7]. For all results, we use significance level $\alpha = 0.05$. We use Friedman's method [49], which found a statistically significant difference between the performance of the three algorithms, with GeTeM being ranked the best performing, DTW second best, and Euclidean the worst (p-value computed by Iman and Daveport test was 0.0028). With Bergmann-Hommel's post-hoc procedure [50] we found that the differences in accuracies for GeTeM vs. DTW and GeTeM vs. Euclidean were both statistically significant (adjusted p-values were 0.029 and 0.0022 respectively). The difference in accuracies for DTW vs. Euclidean was not, however, found to be statistically significant (adjusted p-value was 0.23).

Note that DTW is more suited (by design) for these data sets, given that the data is segmented, the segments are short, and the problem is to find a single matching pattern or motif in pairs of segments. Most of the data sets do not contain periodic data, as GeTeM (in principle) requires, but GeTeM still outperforms DTW.

2.3.3 Activity Recognition

The data set used for the activity recognition experiment was provided to us by Prof. Dieter Fox's group at the University of Washington. The data was collected using the Intel Mobile Sensing Platform (MSP) [51], which contains a number of sensors including a 3-axis accelerometer and a barometric pressure sensor; these are the only sensors we will use. Six participants were outfitted with the MSP

[7]We note that a few of the data sets are related (e.g., Lightning 2 and Lightning 7 contain the same data but with different class labels; uWave X, Y, and Z are measurements on different dimensions from a single dynamical system), however removing these data sets has only a small effect on our results.

units, which clip onto a belt at the side of the hip, and data was collected from a series of 36 two-hour excursions (6 per user) which took place over a period of three weeks. The participants were accompanied by an observer who recorded the labels as the activities were being performed. The labels were: walking, lingering, running, up stairs, downstairs, and riding in a vehicle. We omitted the vehicle activity for now (because we are not processing GPS signal). Taking into account equipment failures and data with obvious errors in the labelling, the working data set consists of 50 hours of labeled data, roughly equally distributed among the six participants.

The accelerometer data is sampled at 512Hz, which we decimate to 32Hz. The accelerometer data consists of three measurements at each time step, corresponding to the acceleration along each of the three axes, x, y, and z. We combine these three measurements to form a single measure of the magnitude of the acceleration vector $a = \sqrt{x^2 + y^2 + z^2} - g$, where $g = 9.8m/s^2$ is the Earth's gravity. Subtracting g causes the acceleration when the device is at rest or moving at a constant velocity to be 0. The barometric pressure is sampled at 7.1Hz, smoothed, and then upsampled to 32Hz. The gradient of the signal is used as an additional feature.

For the experiment, we split the data set up into six parts, each containing the data from a specific participant. Note that we did not use GeTeM for this experiment, instead we used features generated directly from the time-delay embedding of the data. We began by projecting all of the accelerometer data into a time-delay reconstruction space with parameters $\tau = 3$ and $m = 16$. For each user, we constructed a training set by selecting randomly 25% of these embedded data points. This corresponds to approximately 2 hours of data, or 230,000 samples for each participant. Next, we performed PCA on the data points in the training set. We then projected all the reconstructed accelerometer data onto the principal components corresponding to the 9 largest eigenvalues from the training data. This produces 9 features (i.e., the coordinates in the 9-dimensional projection) for each data point in the original time series, and we combined these features with

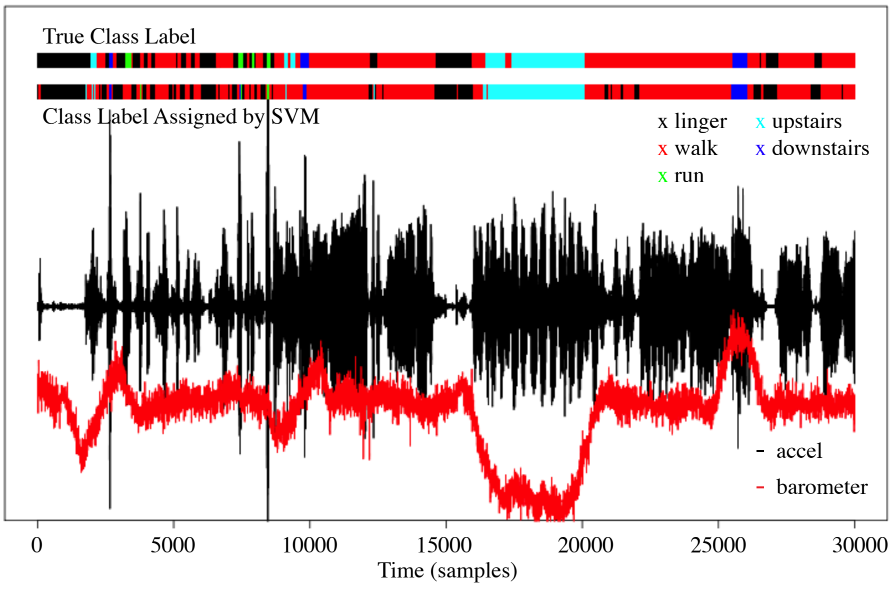


FIGURE 2.5: Classification results on a segment of the data from user 6 using a classifier trained on user 4. The top coloured bar above the data shows the labels assigned by a human labeller and the bottom coloured bar shows labels assigned by the classifier. The black line shows the magnitude of the accelerometer data and the red line is the gradient of the barometric pressure.

the barometric pressure to form the input features for a support vector machine (SVM) classifier [52]. As a basis for comparison, we also trained an SVM on the raw accelerometer value and gradient of the barometric pressure as inputs at each time step.

We tested each classifier on the entire data set for each user. The results are shown in Table 2.3. The first row contains the results when training using only the single raw accelerometer value and the gradient of the barometric pressure, trained on data from User 1, and testing on all of the users. It is clear that using features obtained by time-delay embeddings significantly improves classifier performance. Figure 2.5 shows the result of using the SVM trained on User 3 to label a segment of the data for User 6.

The classifier performs well across all users, regardless of the user on which it was trained. The average accuracy for the experiments using the time-delay embedding

		Testing						
		User 1	User 2	User 3	User 4	User 5	User 6	Total
	User 1, simple	77.26%	75.61%	76.94%	77.86%	78.55%	75.90%	77.11%
Training	User 1	87.89%	82.6%	84.29%	86.73%	85.76%	84.67%	85.27%
	User 2	85.95%	84.35%	84.04%	86.59%	86.04%	85.50%	85.49%
	User 3	86.28%	82.98%	85.08%	87.03%	85.99%	85.07%	85.44%
	User 4	86.00%	82.69%	84.11%	89.41%	85.56%	86.12%	85.79%
	User 5	86.15%	82.89%	84.06%	86.24%	86.61%	83.83%	85.04%
	User 6	86.29%	83.48%	83.85%	87.56%	85.56%	87.75%	85.83%
							Average:	85.48±0.30%

Table 2.3: Results for activity recognition task. Each row corresponds to the results of training on the participant given in the first column, and the values indicate the classification accuracy on the data set for the user specified by the column header. The last column gives the results on the entire data set. The first row corresponds to using only the raw accelerometer and barometric pressure gradient data, without using time-delay embedding.

on the entire data set is 85.48±0.30%. This result demonstrates that these features are useful for activity recognition devices, because the system can be calibrated on one user, then deployed to other users, and the performance is very similar (the accuracy loss, as can be seen from the table, is 1-5%).

The accuracy figures could be improved with further post-processing, because as can be seen in Figure 2.5, most of the mislabelled segments are short. State of the art activity recognition systems routinely perform temporal smoothing, which reduces such errors. As a simple test, we aggregated sequences of 8 predictions and took a majority vote at 1/4 second intervals (instead of making 32 separate predictions every second), matching the rate of existing activity recognition systems [3]. With this approach, the error rate decreased by 2-4%, depending on the experiment. However, in this chapter we want to focus on the power of the representation, and not try to use other methods to mitigate errors. Better performance could also be achieved using other classification methods, such as the decision stumps classifiers used by Lester et al. [3] or the semi-supervised CRFs used by Mahdaviani and Choudhury [53]. We chose SVMs simply because of the ease of using off-the-shelf code.

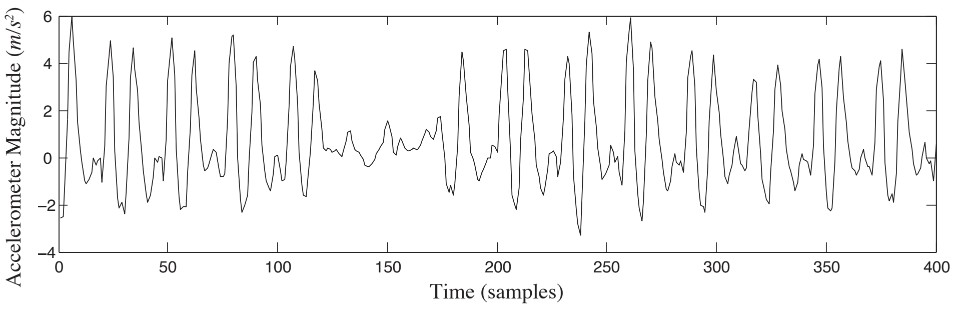


Figure 2.6: An example segment of accelerometer data.

2.3.4 Gait Recognition

Gait recognition is the problem of identifying a person by their manner of walking, or carriage. The problem of gait recognition has been studied in depth in the computer vision and biomechanics communities, where the goal is to identify an individual based on a sequence of images or silhouettes captured while the individual is walking [54]. Recent work has considered gait recognition using wearable sensors as a means for biometric identification [55]. For our experiment, we implemented a data collection and analysis tool for the Google Android operating system, which runs on the HTC G1© mobile phone[8]. In our experiments, the phone was placed in the front trouser pocket of an individual, and collected data from the 3-axis accelerometer in the device at a rate of 25Hz[9] as the subject walked a certain distance (between 10 and 50 meters), turned around, and returned to their starting point. We collected a set of 40 traces, each containing between 12 and 120 seconds of data from 40 individuals. After manually inspecting the data, two traces were discarded as the device malfunctioned and truncated the data. An example of the data is shown in Figure 2.6.

We used 3-fold cross-validation, where each fold consists of a sequential block consisting of 1/3 of the data and, as is standard for cross-validation, at each round one block is chosen as the test data, and the others are used as training data. While

[8]Specifications: 528 MHz ARM 11 CPU; 192 MB RAM.

[9]The sampling rate on the phone is inconsistent, and ranges from 17Hz to 30Hz depending on the processor load, which is an additional source of noise.

this adds a single point of discontinuity in the training set when the test set is not at the very beginning or end of the data set, this does not have a noticeable effect on our results. We used a 1-NN classifier and the GeTeM distance with parameters $m = 5, \tau = 3$, and $k = 4$ neighbours. These parameters were chosen based on previous experiments before this data set was collected. The accuracy, averaged over the 3 folds, was approximately 93.9%. 7 of the users were misclassified on one of the iterations, but no user was misclassified on more than one iteration.

While the accuracy of GeTeM is high, it is important to note that this data is collected in a controlled environment. However, this setting matches the experimental setting used by most work on gait recognition using wearable sensors [55–57], and thus it is fair to compare our results with those of existing work. Gafurov et al. [55] report an accuracy of 86.3% on a data set with 50 individuals, collected under similar conditions. Ailisto et al. [56] and Bächlin et al. [57] specifically consider the problem of biometric authentication based on gait, and report equal error rates (EER) of 7% and 2.8%, respectively. Performing a thorough comparison between approaches is difficult, as methods are often patented and details are kept secret, and data sets consist of private data and cannot be shared. We attempt to ameliorate what we perceive as significant problems with this field by both making our code available (see Section 2.2) and by collecting and sharing a realistic gait recognition data set (see Section 2.4.2). Our hope is that by providing a realistic benchmark data set, we can both spur fair comparisons between gait recognition algorithms (even if researchers choose to keep the code or specific implementation details secret), and properly evaluate how these approaches would work in the real world.

2.3.5 Activity Data Visualization

In this section, we discuss how GeTeM can be used to visualize and identify clusters of similar activities in an unlabelled data set. Our approach is to partition the data set into N short (5 to 10 second) segments and build an $N \times N$ similarity matrix $\boldsymbol{A}$ where each element a_{ij} is the GeTeM distance between time-delay models for segments i and j.

The resulting similarity matrix is asymmetric. For clustering, we employ a technique for spectral mapping of asymmetric data [58]. First, the similarity matrix is converted to a stochastic matrix $\boldsymbol{P}$ with elements:

$$p_{ij} = \frac{a_{ij}}{\sum_k a_{ik}}.$$

The matrix $\boldsymbol{P}$ can be considered to represent the transition matrix for a random walk on a graph G with N vertices. Let the vertices be labeled $1, \ldots, N$, and let K denote the number of clusters. A clustering on N vertices is a vector $\boldsymbol{c} \in \{1, \ldots, K\}^N$, such that $c_i = j$ indicates that vertex i is a member of cluster j.

Intuitively, spectral clustering methods find a partition of the vertices into K clusters such that the characteristics of a random walk on the graph G can be accurately summarized by a Markov chain on K states where the transition matrix gives the cluster to cluster transition probabilities. In other words, the vertices get clustered based on how similarly they transition to each of the clusters. We say that a transition matrix $\boldsymbol{P}$ for a random walk on a graph G with N nodes admits a perfect K-clustering if there exists a clustering vector c and a $K \times K$ stochastic matrix $\hat{\boldsymbol{P}}$ with elements $\hat{p}$ such that for all $i = 1, \ldots, N$ and $j = 1, \ldots, N$, $p_{ij} = \hat{p}_{c_i, c_j}$. That is, for any pair of clusters i and j, the transition probabilities from any node in i to any node in j are the same.

Spectral clustering methods [59] are based on computing the eigendecomposition of the transition matrix $\boldsymbol{P}$. Let $\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(N)}$ be the eigenvectors and $\lambda_1 \geq \cdots \geq \lambda_N$ be the corresponding eigenvalues of $\boldsymbol{P}$. Since $\boldsymbol{P}$ is stochastic, $\lambda_1 = 1$, and all of the elements of $\boldsymbol{u}^{(1)}$ will be equal. Since $\boldsymbol{P}$ is asymmetric, the eigenvectors will be complex. However, the vector $\tilde{\boldsymbol{u}}^{(i)} = \mathrm{Re}(\boldsymbol{u}^{(i)}) + \mathrm{Im}(\boldsymbol{u}^{(i)})$, the sum of the real and complex components of $\boldsymbol{u}^{(i)}$, preserves all of the information about the eigenvector

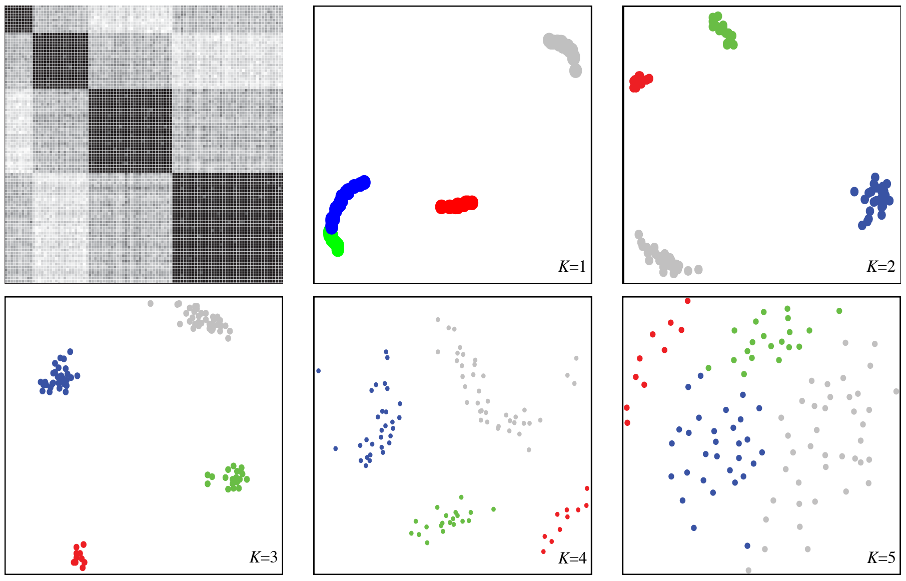


FIGURE 2.7: Example of data visualization for artificial data set. On the top-left is the asymmetric distance matrix, and the remaining plots depict the results of applying the spectral mapping and force-directed layout for $K = 1, \ldots 5$. The colours indicate the actual clusters that the data belong to.

$\boldsymbol{u}^{(i)}$ [58]. In the remainder of this chapter, when we refer to the ith eigenvector of $\boldsymbol{P}$, we are considering the real vector $\tilde{\boldsymbol{u}}^{(i)}$.

Given the number of clusters, K, then for each segment i, we can compute a vector that represents its spectral properties, $\boldsymbol{v}_i = (\boldsymbol{u}_i^{(2)}, \ldots, \boldsymbol{u}_i^{(K+1)})$, which we call the *spectral profile* for i. Pentney and Meila [58] show[10] that if a transition matrix P admits a perfect K-clustering $\boldsymbol{c}$, then the set $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N\}$ will contain exactly K unique vectors, where for all $i, j = 1, \ldots, N$, $\boldsymbol{v}_i = \boldsymbol{v}_j$ if and only if $\boldsymbol{c}_i = \boldsymbol{c}_j$.

The spectral profiles are vectors in $\mathbb{R}^K$, so we can compare them based on their Euclidean distance. In practice, when there are K clusters of nodes in the graph in which the transition probabilities between pairs of elements in the same clusters are similar but not identical, the spectral profiles for elements in the same cluster

[10]Their theorem is stronger than what we present here, in that each component of the spectral profile vector uniquely distinguishes the cluster, and so just one of the eigenvectors, $\boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(K+1)}$, is enough to identify the clusters. However, when the data does not admit a perfect clustering, it is useful to make use of more than one eigenvector.

will also be similar. The spectral profiles can be used as input to a standard clustering algorithm such as k-means, to find clusters in the data.

Since the true number of clusters is typically unknown, K is a parameter that we choose. This approach is fairly robust to the choice of K. If K is too small, we may collapse multiple clusters into single clusters. If K is too large, the noise in the data is amplified and clustering becomes more difficult. In our experiments, we try values of K from 2 to 6, and pick the value for which the clusters appear most clearly defined.

To visualize data, we choose K, compute the spectral profiles, then compute a new symmetric distance matrix consisting of the Euclidean distances between the spectral profiles for each node. We use the Kamada-Kawai [60] force-based layout algorithm to draw a 2D image of the data[11]. It is important to note that this approach constrains the visualization to 2D, and so objects may appear to be clustered together when they are, in fact, far apart. Additionally, the Kamada-Kawai algorithm will find a locally optimal representation, and can produce very different visualizations when run with different initial conditions. For our plots, we ran the algorithm a number of times, and picked the most visually pleasing results.

First, we demonstrate this approach on synthetic data. We generated an artificial data set consisting of 100 points drawn from four 2D multivariate normal distributions. The distances between all points are computed, and then a small amount of white noise is added to the distances to create an asymmetrical distance matrix. Figure 2.7 shows the distance matrix on the top left, and then five visualizations where the value for K, the number of eigenvectors to use in computing the spectral profiles, varies from 1 to 5. When K is 1, two of the clusters (green and blue) become indistinguishable. When K is 2, 3, or 4, the clusters are easily identifiable.

[11]The Kamada-Kawai algorithm can also operate in higher dimensions, however for ease of presentation, we restrict ourselves to 2D.

As K grows to 5, which is more than the number of clusters in the data, noise begins to interfere with our ability to clearly distinguish the clusters.

Next, we demonstrate this approach on real activity data. We collected data from four individuals, three males and a female, who were asked to perform a sequence of four activities at the gym for approximately 2 minutes each. The activities were: using an elliptical machine, riding a stationary bike, using a stationary rowing machine, and using a stair-climbing device. The three male subjects also ran on a treadmill. One of the male subjects only completed the treadmill and biking activities. Data was collected by custom software (see Appendix A) running on an HTC G1© mobile phone clipped to the waistband of the subject's clothing. We collected data from the triaxial acceleration sensor in the G1 at the maximum frequency allowed by the phone's operating system, 35Hz. From the three input channels, x, y, and z, we formed a univariate time series by taking the Euclidean vector norm, $(x^2 + y^2 + z^2)^{1/2}$.

The data was manually segmented into the various activities, then split into 5-second windows where the first and last second of each segment overlapped with the previous or next segment, respectively. The first and last 5-second segments were discarded, and a time-delay model was built for each 5-second window. This resulted in 27-46 models per activity. There were 597 models in total corresponding to 597 5-second segments. A 597×597 similarity matrix was generated by running each of the segments through each of the models. This is a process that can be easily parallelized, and took approximately 10 minutes on a 3.0GHz Intel Xeon Quad-Core desktop computer with 2 GB RAM. A spectral mapping was performed, with $K = 4$ (the number of subjects), and the Kamada-Kawai algorithm was used to generate a plot of the data.

Figure 2.8 shows the visualization of the entire data set. The clusters stand out, and we see that, in general, the distances between clusters for the same activity (depicted by points with the same colour), but different subjects (i.e., different

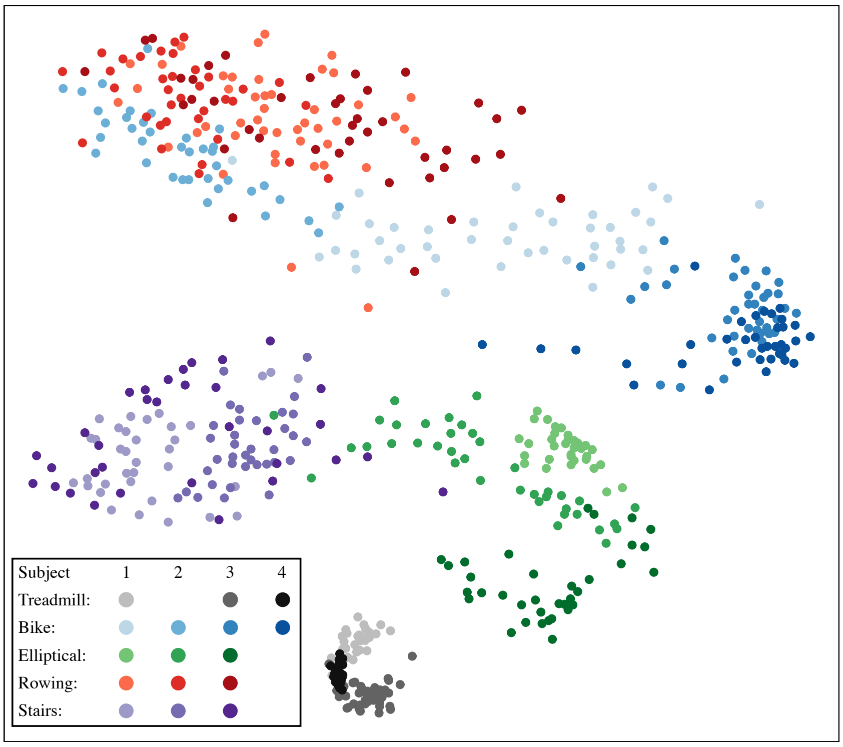


FIGURE 2.8: Result of applying unsupervised visualization technique to data generated from all four subjects.

shading), are closer to each other than the clusters for different activities. With the exception of the biking data from Subject 2, the activities appear fairly well-separated.

It is clear from Figure 2.8 that the unsupervised approach presented in this work is very useful for visualizing this kind of data in the absence of labels, and that, together with a clustering algorithm such as k-means, is useful for identifying distinct activities in a data set. In the following section, we demonstrate that the structure induced by our clustering can also assist in classifying activities with sparsely labeled data.

2.3.6 Semi-Supervised Learning

Semi-supervised learning considers the case where the training set contains both labeled and unlabelled examples. The algorithm we present is intentionally simple, so that the performance explicitly depends on the quality of the distances used for clustering. We note that this does not reflect the state of the art in semi-supervised algorithms (see [19] for a thorough survey), and it is likely that a more powerful algorithm would achieve better results.

First, we perform hierarchical clustering on the entire training set, as in Section 2.3.1, to produce a dendrogram, then we traverse it in a depth-first manner. Each node represents a set of points, some of which may be labeled. If all of the labels on points in the node agree, we assign that label to all of the points in the node and cut the tree at that point. If there are no labels, then we also cut the tree at that node, but leave the points unlabelled. If there are multiple unique labels in the set of nodes, we move on and consider the children of that node. Once finished, we will have a larger set of labeled nodes, but some unlabelled nodes may remain.

In the second pass, we assign labels to the remaining unlabelled nodes. To do this, we compute the distance from each unlabelled node to its nearest labeled neighbour. The unlabelled node that is closest to a labeled node is assigned the label of its nearest labeled neighbour. If there are remaining unlabelled nodes, we repeat the process until all nodes have been assigned a labeled neighbour. In the experiments, we found that often there are no unlabelled nodes, so this second pass can be skipped. In some cases, however, as many as 30% of the nodes remained unlabelled after the first pass.

We use the same data set as in Section 2.3.5. We removed all but n labels from each of the 15 subject-activity pairs, with n from 1 to 25, and used the semi-supervised technique with the GeTeM-based and DTW-based distances. Since we had the correct labels, we were able to assess the accuracy of the labelling, for both the activity and the subject. For each value of n, we ran the experiment 100 times, picking random examples to be unlabelled each time. The results are shown in Figure 2.9, with error bars representing 95% confidence intervals around the mean. GeTeM consistently outperforms DTW for all values of n, and is able to correctly classify the subject and activity with over 90% accuracy using only two labels per subject-activity pair. The small size of the confidence intervals is not surprising given the repetitive nature of these activities.

2.3.7 Discussion

The results in this section demonstrate the versatility of the GeTeM distance measure. In the activity and gait recognition tasks, the results have high accuracy, despite the fact that we use significantly fewer features than are generally used for time series analysis based on signal processing techniques. For the gait recognition task, our method requires a time window of 22 samples. The average walking cycle has a period of approximately one second, and so for a 1Hz signal to be detected confidently using any method that computes the spectrum of the data, hundreds of samples must be considered.

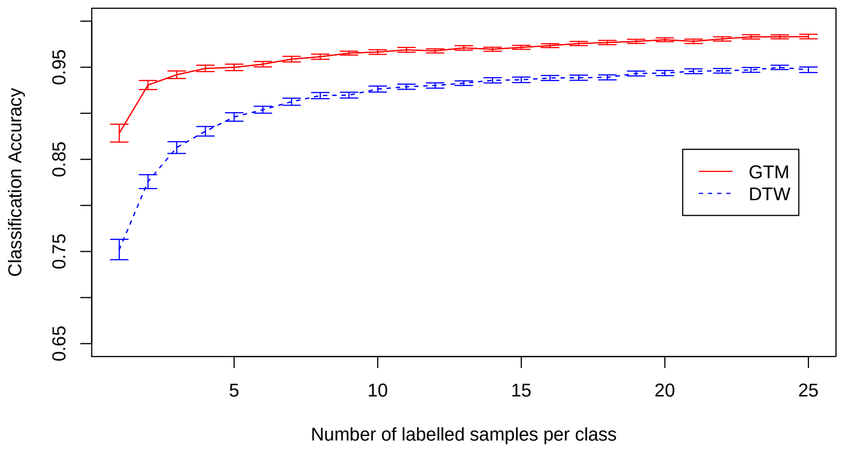


Figure 2.9: Result of applying the semi-supervised technique to the activity data set. The x-axis represents the number of labeled examples for each class, and the accuracy is computed over data with missing labels. Error bars represent 95% confidence intervals.

Our activity recognition results use significantly fewer features than current approaches, but achieve similar results. To provide some perspective, we note that a similar data set was considered by Subramanya et al. [61], who applied the methods proposed by Lester et al. [3] and achieved an accuracy of 82.1% on the same set of activities we consider. Their system used 650 features of the time series, composed of cepstral coefficients, FFT frequency coefficients, spectral entropy, band-pass filter coefficients, correlations, and a number of other features that require a non-trivial amount of computation. A modified version of AdaBoost was then used to select the top 50 features for classification. Comparatively, we used 16 samples of the raw signal, corresponding to a window of 48 samples, or one and a half seconds, and then computed a linear projection to get the 9 features that are used to train the classifier. We emphasize that existing results are obtained on different data sets, and we only report their results because their work is the closest in nature to our own. We make no claim that our method obtains better results, only that the features that we use are considerably easier to compute, while the classification accuracy is similar.

While originally intended for extracting features from long, unsegmented sequences of time series, such as the data used for activity and gait recognition, GeTeM also provides good distance measures for clustering and nearest-neighbour classification of short segmented time series. When data can be carefully segmented so that the patterns sought are temporally aligned between pairs of segments, then treating n-length sequences as points in $\mathbb{R}^n$ and using the Euclidean distance between segments to train a 1-NN classifier is hard to beat [62]. When the patterns in the sequences are out of phase, running an alignment technique such as DTW [63] can compensate and 1-NN with Euclidean distance between aligned elements also performs well.

As described in Section 2.2, GeTeM allows for comparisons between unaligned segments without distorting their temporal properties, while DTW performs a warping in the temporal domain. This is likely the cause of the poor performance of of DTW on the ECG data in Section 2.3.1. In the semi-supervised experiment in Section 2.3.6, the data is segmented into windows of fixed time lengths. Therefore, segments contain unaligned data and may also contain different numbers of repetitions of the patterns of interest. It is most likely due to these properties that GeTeM greatly outperforms DTW on this data. Based on these results, GeTeM appears preferable for real data, which is usually not well segmented or aligned.

In terms of computational cost, GeTeM required less time than DTW. For the ECG clustering example in Section 2.3.1, the 40×40 GeTeM distance matrix took approximately 190 seconds of CPU time to compute, while the DTW distance matrix took approximately 426 seconds. For the semi-supervised experiment in Section 2.3.6, the 597×597 GeTeM distance matrix took approximately 36 minutes to compute, while DTW took approximately 70 minutes. We note that neither algorithm was optimized, and that there are techniques (see Section 2.2) for speeding up both GeTeM and DTW.

GeTeM has been ported to the Android mobile operating system as a part of our Android Data Collection Platform (see Appendix A). On the HTC G1 mobile phone, which was used for our data collection tasks, activity and gait recognition classification can be done in real time on the mobile phone. This includes both training the models, where time-delay parameters are tuned automatically using cross-validation, and performing classification on streaming data from the accelerometer. Both the gait and activity recognition systems have been demonstrated the following conferences: Neural Information Processing Systems (NIPS), Dec., 2009; Ubiquitous Computing (UBICOMP), Sept., 2010; and the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), Sept., 2011.

2.4 Boosted Time-Delay Classifiers

In this section, we focus on analyzing long, unsegmented time series data, which arise from many real-world problems, such as activity recognition from wearable sensor data [3, 61], heart rate and breathing monitoring, seizure detection and financial prediction. As described in Section 2.1.1.3, the goal in this case is to identify the segments that correspond to a particular class. We use GeTeM in this context in order to extract useful features of the data.

More precisely, we work in the framework of radial-basis functions, in which one picks a set of points in the space of the input values, and then represents new input data using the set of distances to each of these points, which become input features. We propose to compute features for new time series by computing the GeTeM distances between the data and a set of selected segments from a training set The challenge is how to pick the segments from the training data that will provide the most informative features. Our algorithm selects segments using a boosting-based approach. We evaluate performance on a new, challenging benchmark problem, comparing against two strong baseline approaches.

2.4.1 TDEBOOST

We propose an algorithm called TDEBOOST for classifying time series. TDE-BOOST is based on the SAMME.R algorithm [64], a boosting algorithm for multi-class classification. Boosting is a powerful ensemble method for classification, exemplified by the AdaBoost algorithm [65] for binary classification problems. AdaBoost associates a weight with each example in a training set of labeled data $\langle(\mathbf{x}_i, y_i)\rangle_{i=1}^{N}$. The weights are initially all equal. At iteration l, a base classifier h_l is trained on the weighted data set and then used to classify the entire training set. The weights of misclassified examples are increased (relative to the other examples). The weights are then renormalized to sum to one, and the next iteration begins. Once a sufficient number of iterations M has been performed, a classifier is formed as a weighted majority vote of the outputs of the base classifiers trained on each iteration, $h(x) = \sum_{l=1}^{M} \alpha_l h_l(x)$. The weight α_l for a base classifier h_l is computed based on its accuracy on the weighted data set on which it was trained.

For multi-class problems (i.e., $y_i \in \{1, \ldots, K\}$), a common approach is to break the problem into multiple two-class problems, training multiple binary classifiers and aggregating their results. AdaBoost requires that the base classifiers have an error rate at most 1/2 at every iteration, in order to work correctly. For the K-class problem, this requirement is often hard to satisfy. SAMME uses a correction term in the weight calculation, such that the accuracy of the base classifier at each round need only be at least $1/K$. A recent theoretical result [66] shows that this approach is actually too permissive, and proposes a better theoretical error bound. However, this new method has not yet been thoroughly tested in practice (though initial results are promising). We base the experiments in this chapter on a variant of SAMME called SAMME.R that builds an ensemble of base classifiers, each of which output a probability distribution (or set of confidence scores) over the class labels, rather than selecting a single class label.

The most straightforward approach to boosting in our setting would be to build classifiers directly out of the time-delay embedding models, by taking a collection of models (one for each class) and defining the classifier for a new snippet of data by picking the class with the most similar model. However, this approach can be problematic for time series (especially for the practical applications we envision). If the models picked initially are noisy, the base classifier is bound to have very high error, and even if we continue picking new data segments and re-weighting them, the classifiers would continue to be poor as the base classifiers would not meet the required $1/K$ accuracy. Thus, there is no way to recover from a bad initial model.

```
**Input:** A training set $\langle (o_i, y_i) \rangle_{i=1}^{N}$, a specified number of rounds $L$, model and window-length parameters $W$ and $s$, time-delay embedding parameters $m$ and $\tau$, and a hypothesis class $\mathcal{H}$.
Let $N' = N - s + 1$, set weights, $w_i = 1/N', i = 1, \dots, N'$.
for $l = 1, \dots, L$ do
  If $l = 1$, build a set of $M$ time-delay models $\mathcal{M}_1 = \{u_1, \dots, u_M\}$, otherwise update the set of models $\mathcal{M}_{l-1}$ to produce $\mathcal{M}_l$.
  for $t = 1, \dots, N'$ do
    Build model $v_t$ for subsequence $\langle o_t, \dots, o_{t+s-1} \rangle$.
    Compute features $\mathbf{x}_t = (S(u_1, v_t), \dots, S(u_M, v_t))$.
  end for
  Fit a classifier $h_l \in \mathcal{H}$ to $\langle (\mathbf{x}_t, y_t) \rangle_{t=1}^{N'}$ using weights $w_i$.
  For each class label $k = 1, \dots, K$, obtain the weighted hypothesis
    $g_l^k(\mathbf{x}) \leftarrow (K-1) \left( \log h_l^k(\mathbf{x}) - \frac{1}{K} \sum_{k'=1}^{K} \log h_l^{k'}(\mathbf{x}) \right)$.
  For $i = 1, \dots, N'$, update the weights
    $w_i \leftarrow w_i \exp \left( -\frac{K-1}{K} \sum_{k=1}^{K} \left( -\frac{1}{K-1} \right)^{\mathbb{I}(y_i \neq k)} h_l^k(\mathbf{x_i}) \right)$.
  Renormalize the weights $w_i$.
end for
**Output:** Models $\mathcal{M}_1, \dots, \mathcal{M}_L$ and classifier
  $h(\mathbf{x}) = \operatorname*{argmax}_k \sum_{m=1}^{M} g_m^k(\mathbf{x})$.
```

FIGURE 2.10: TDEBOOST algorithm pseudocode

To prevent the problem of bad initial models, we modify the boosting framework to allow the set of features to change over time. More precisely, at each boosting round we use GeTeM with an ensemble of time-delay models to extract from the time series the features that are used by the base classifiers. We allow the set of models to change between boosting rounds, which means that the set of features used by the base learners also changes. The algorithm is presented in Figure 2.10.

The algorithm begins by generating a set of M models $\mathcal{M} = \{u_1, \dots, u_M\}$. Each model u_i is generated by first selecting an index j at random or according to some application-specific criteria, and constructing a time-delay model for the subsequence $\langle o_j, \ldots, o_{j+W-1} \rangle$, where W is a model-size parameter. The set of models $\mathcal{M}$ is used to extract an M-dimensional feature vector $\mathbf{x}_t$: a time-delay model v is constructed from $\langle o_t, \ldots, o_{t+s-1} \rangle$, where s is the segment-length parameter, and for $i = 1, \ldots, M$, the ith element $\mathbf{x}_t$ is set to $S(u_i, v)$. In other words, the set of similarity scores between each of the models in $\mathcal{M}$ and a new subsequence become the feature vector. The feature vector and class label pairs $(\mathbf{x}_t, y_t)$ are then used to train a classifier, exactly as in the SAMME.R algorithm. Note that the feature vector $\mathbf{x}_t$ is built from a window of observations starting at o_t; one could also consider using the mode of the class labels within this window, rather than the label of the first observation, y_t, as the target for training the classifier.

We use the notation $h_m^k(\mathbf{x})$ to denote the probability assigned to class k by the hypothesis built at round m when evaluated on input feature $\mathbf{x}$. Classifiers that output only a class label can be used by modifying the SAMME algorithm in the same way that we modified SAMME.R. We also experimented with the SAMME variant, but found that modifying SAMME.R gives better performance on our data.

At each iteration, the set of models $\mathcal{M}$ is updated. This step can be application-specific. The easiest approach is to generate a new set of models at each step, either by randomly sampling new segments, or by a more informed method. Or, specific models can be replaced by new models. In our gait recognition application, discussed in more detailed in the following section, the set of models $\mathcal{M}$ contains one model for each class, and at each iteration one model is selected and replaced by a new model for the same class. The choice of model is informed by the current boosting weights: we select a segment of the data on which the boosting weights (i.e., the cumulative training error) is high. Based on the assumption that short subsequences of training points are likely to all have the same label, the intuition here is that by building models for regions of the time series in which performance is poor, the features will focus the ensemble hypothesis on these “harder” regions, and improve overall classification.

The algorithm differs from classical boosting because of this feature selection step that precedes the construction of the base classifiers. However, the theory of boosting still applies to our setting. To see this, consider base classifiers in which *all* possible models are included as features in a linear classifier. In this case, minimizing the error on the new distribution is equivalent to increasing the parameter (in the linear classifier) of the nearest examples to those with high weights in the boosted distribution. However, from a computational point of view, we cannot afford to include a large set of models in the classifier. What we do can be viewed as setting to 0 the weight of the existing model and picking greedily the area that most reduces training error. This is an approximation to the case when all models are included, but is still guaranteed to reduce error on the boosting distribution. Since this is the main requirement of boosting theory, existing results still apply.

2.4.2 Walking Dataset

The problem of identifying an individual using data collected from a wearable accelerometer sensor, called gait recognition, is a challenging and well-studied time series classification task. However, empirical evaluations consider data collected in very controlled environments. As described in Section 2.3.4, Ailisto et al. [56] and Gafurov et al. [55] collected data sets composed of subjects walking a distance of approximately 20m, in a straight line, on a level surface, with a sensor carefully attached to a specific body part, and Bächlin et al. [57] further controlled the speed by having subjects walk on a treadmill. Due to the personal nature of the data, it is typically not shared between groups, and while each of the proposed methods performs well on the data collected by the corresponding authors, thorough comparisons on the same data set are rarely performed. Additionally, since these algorithms have potential for commercialization, authors are hesitant to share code or specific implementation details, making empirical comparisons difficult.

We envision gait recognition as a useful practical technology. Mobile devices are ubiquitous, and frequently contain accelerometers. Gait recognition algorithms provide opportunities to improve both the security and the user experience of these devices. For example, if more than one person shares a mobile phone, the phone can be aware of its current user and personalize the contact list or email accounts that are available to the user. As a security feature, mobile phones could detect if someone other than the owner is carrying the device, and prevent access to private information. However, the conditions in which these algorithms have to operate are much noisier than existing benchmarking data sets.

We present a data set that was collected in real circumstances, analogous to the environment in which a practical gait recognition algorithm would operate. We have made the data set publicly available[12], and hope that it will provide a challenging benchmark for gait recognition as well as machine learning research. The data was collected from 20 subjects, 10 males and 10 females, ranging in age from 19 to 35, ranging in weights from 47kg to 100kg, and ranging in heights from 160cm to 193cm. Each subject was asked to carry an HTC Nexus One© mobile phone[13] in their pocket and walk around outdoors for 15 minutes, then return on a subsequent day to perform another 15 minute walk. The subjects were allowed to change clothing and footwear between the two days at their discretion (most did), and the phone was casually placed in the pocket, rather than being affixed at a specific position. Subjects were also asked to walk both on grass and concrete surfaces, up and down hills, and to pause occasionally. The device also logged GPS data, to verify that the subjects followed instructions. Specific details, as well as descriptions of the footwear and clothing worn by the subjects are included in the data set.

[12]Data available at `http://www.cs.mcgill.ca/~jfrank8/gait.html`

[13]Specifications: 1 GHz Scorpion CPU; 512 MB RAM.

The HTC Nexus One phone runs the Google Android operating system, and data was collected using a custom application (see Appendix A) that collects data from the sensors as quickly as possible. The sampling frequency is variable and depends on factors such as the processor load. The data is timestamped with millisecond precision. In the data we collected, the average frequency is approximately 28.57Hz, with a standard deviation of approximately 4.17Hz. The frequency is relatively low compared to the data analyzed in previous work; this is representative of what one would expect from a commercial mobile device.

2.4.3 Results

We use frame-based performance metrics [16] as the evaluation criterion. We resample the data to a fixed frequency of 25Hz using linear interpolation, and compute a single magnitude measurement from the three acceleration directions, as in Section 2.3.6. The data contain segments of walking, as well as segments where the subjects were stationary. We associate a common label, *lingering*, to all portions of the data, for all subjects, that contain *non-walking* data. We use a crude but effective method for labelling lingering data. We consider a window of 50 samples; if the sum of the absolute values of the acceleration magnitudes in this window was less than 125 m/s^2, then the first sample in the window was labeled as lingering. These parameters were determined by trial-and-error, visually inspecting the resulting labels. Lingering data was assigned the class label 0, and the subjects were identified with labels 1 to 20. Every sample is assigned a label. After filtering, there are between 18,000 and 50,000 samples per walker, and roughly 87,000 samples for the lingering class.

Since every sample has an associated label and there is no significant class skew, we evaluate performance by computing the accuracy, or percentage of samples correctly labeled. When considering a single subject, we report both precision (i.e., the fraction of examples that our algorithm labels as that subject that are correct), and recall (i.e., the fraction of examples that should have been labeled as a certain subject that were correctly labeled).

We compare TDEBOOST with a baseline approach which extracts 159 features used in the activity recognition system of Lester et al. [3], comprised of Fourier coefficients, cepstral coefficients, and bandpass filter responses over windows ranging from 2 seconds to a minute. Features are associated with the first sample in the window. The features are used to train an AdaBoost.M1 classifier [65] with random forests composed of 100 trees, each trained on 10 randomly selected features as the base classifier. 100 iterations of boosting were performed. We also compared against selecting models at random from the training data and using them as feature extractors to train a classifier, which we call the bagging approach. We were unable to compare against other gait recognition approaches as they involve segmenting the data into individual footsteps, and these techniques do not scale to large, noisy data sets.

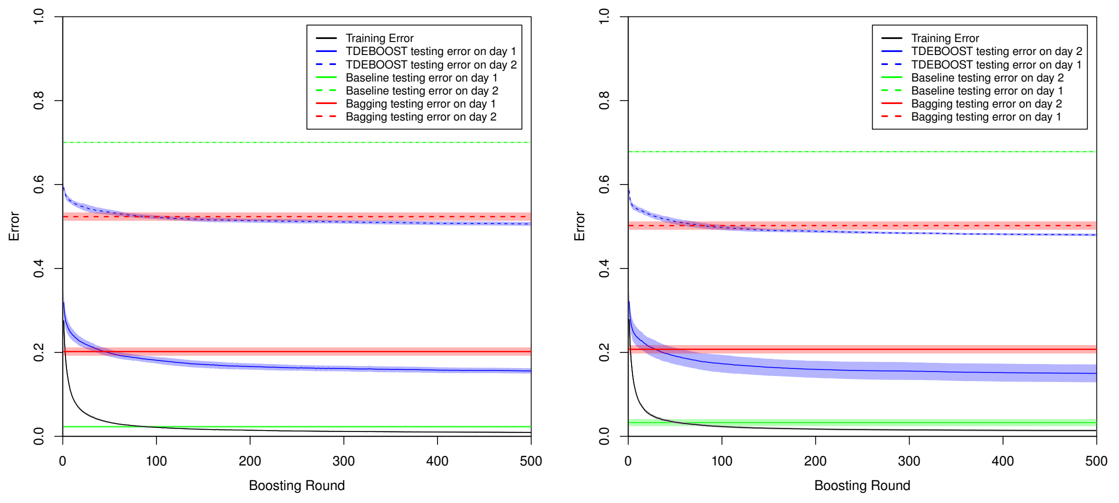
For TDEBOOST, we maintain a set of 20 models, one for each subject. At each round of boosting, the set of models is updated by sampling a training example according to the boosting weights. If the sample corresponds to the lingering class, then we resample, and repeat until the sample corresponds to a subject. Let i be the label associated with the sampled subject; a new model u is built from the window of data beginning with the selected sample, and it replaces the model corresponding to subject i in the current set. As in the baseline approach, we use a random forest with 100 trees, each built on 10 randomly selected features, as the base classifier. We collected an additional trace of two subjects that did not participate in the large data set, walking for approximately one minute each; we used this small amount of data to select parameters. This additional data was split into two halves. The first half was used to tune the values of the time-delay embedding parameters, using initial guesses of $W = 50$ (two seconds) and $s = 32$. Then, fixing the time-delay embedding parameters, the second half was used to tune the parameters W and s. The values selected by this procedure were $m = 7$,

$\tau = 2$, $W = 125$ (5 seconds), and $s = 16$. The number of neighbours chosen to be $k = 2$, based on the fact that the models were small, and thus contained few gait cycles.

We ran TDEBOOST for 500 iterations. Although the training error converged after approximately 100 iterations, we wanted to assess whether the algorithm would overfit the data. We considered two scenarios: training on a segment of the data from one day, and testing on the remaining data from that same day; and training and testing on different days. When training and testing on the same day, we performed 10-fold cross-validation, where each fold consisted of a contiguous segment of the data. When training and testing on different days we used the 10 classifiers that were built in the previous scenario (one per round of cross-validation), and evaluated them on all of the data from the day on which they were not trained. Since the baseline algorithm operates with a larger number of features, we subsample 10% of the training data when training the base classifier. To ensure that the results are not an artefact of the data sampled, we repeat this 10 times for each fold, and average the results.

For the bagging algorithm, we use 100 randomly selected training segments (5 per subject) for models, and a random forest classifier with 500 trees. This was repeated 10 times with 10 different sets of models for each of the 10 rounds of cross-validation, and we report the average accuracy (i.e., in total 100 classifiers were built and tested).

Figure 2.11a shows the results when the classifiers are trained on data collected on the first day, and Figure 2.11b shows the results when the classifiers are trained on data from the second day. The solid lines depict the testing error when trained on data from the same day, and the dotted lines depict the testing error when trained on the other day. The shaded regions represent 95% confidence interval around the mean. All three methods perform significantly worse when the training and testing sets are from different days, confirming the findings of Bächlin et al. [57].



(A) Results when classifiers are trained on day 1.

(B) Results when classifiers are trained on day 2.

FIGURE 2.11: Learning curves for TDEBOOST, baseline, and bagging algorithms. The shaded regions represent 95% confidence intervals around the mean.

If we do not consider the lingering data, the accuracy of TDEBOOST after 500 rounds is 0.42, while the accuracy of the baseline method is 0.20. Chance is 0.05, disregarding the lingering class.

When the classifiers were trained and tested on data from the first day, the average performance of bagging 100 models was roughly equivalent to the performance of TDEBOOST with 45 models, and roughly equivalent to TDEBOOST with only 29 models for the second day. When training and testing on different days, the improvement over bagging was not as drastic, but still significant. When training on day 1 and testing on day 2, bagging with 100 models is roughly equivalent to TDEBOOST with 90 models, and when training on day 2 and testing on day 1, bagging is roughly equivalent to TDEBOOST with 78 models. In both cases, when training and testing on data from the same day, the baseline outperforms both the boosting and bagging approaches, and when training and testing on different days, both boosting and bagging classifiers built with the GeTeM-based features outperform the baseline.

The training error for TDEBOOST drops quickly, and then continues to slowly

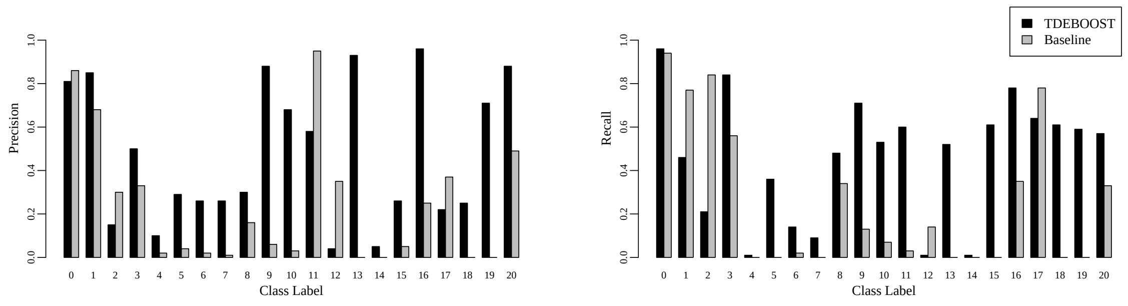


FIGURE 2.12: Precision (left) and recall (right) when training on day 1 and testing on day 2. Class 0 is lingering, the other classes correspond to the subjects.

decrease. However, the testing error also continues to decrease, which highlights an attractive property of boosting: it is typically not prone to overfitting. When trained and tested on the same day, the baseline algorithm outperforms TDE-BOOST after 100 rounds by approximately 0.15. On the other hand, when the training and testing data come from different days, TDEBOOST outperforms the baseline by approximately 0.18. After 500 rounds, TDEBOOST is able to build a classifier from the data collected on one day and label over half of the data from the other day correctly.

We gain more insight into the performance differences by looking at the average precision and recall values for particular classes. For example, Figure 2.12 shows these values when the training data is from day 1, and the testing data is from day 2. Assuming the task is to identify the user of the device, then loosely speaking, precision gives a measure of how often the person that is identified as the user is in fact the real user, and recall gives a measure of how often the real user is correctly identified when they carry the phone.

There are 11 subjects on which the precision for the baseline is less than 0.1, but only 3 for which TDEBOOST has precision less than 0.1. TDEBOOST achieves higher scores than the baseline for both precision and recall on 15 of the 20 subjects, while the baseline achieves higher scores for both precision and recall on only 3 of the subjects. Subjects 4 and 14 were poorly classified by both algorithms; these are the two subjects for which the change in clothing between day 1 and day 2 was most drastic; subject 4 changed from a loose flowing dress to tight shorts, and subject 14 changed from jeans to baggy shorts. Collecting data over many days could substantially improve performance.

We performed an additional post-processing step in order to smooth the labels. Our approach is simple, and based on the assumption that walking segments are separated by segments of lingering. Therefore, whenever we encounter a segment of data that has been labeled as something other than lingering, we begin to accumulate counts for each label, and reassign to the sample the label with the majority vote. We continue until we encounter a sample that is labeled as lingering, in which case we reset the counters. This is equivalent to assuming that each uninterrupted segment of walking corresponds to only one person (i.e., the identity of the subject does not change mid-walk), which is reasonable. With this smoothing, if the classifier is trained on data from day 1 and tested on data from day 2, the average accuracy for TDEBOOST increases from approximately 49.4% to 63.0%, while the accuracy for the baseline decreases from approximately 29.5% to approximately 27.4%

Finally, we compared against a nearest-neighbour classifier. We built a classifier with 2,000 randomly selected windows of 70 samples. The number 2,000 was chosen so that the nearest-neighbour classifier has roughly 50 times as many training points as the 119 time-delay models used for 100 rounds of TDEBOOST (20 initial models and one new model after each subsequent round). Each window was associated with the class label of the majority of the samples in the window. The experiment was repeated 100 times and the accuracy for training on day 1 and testing on day 2 was approximately 42.4% and training on day 2 and testing on 1 was approximately 41.5%. These values are roughly equivalent to the accuracy of TDEBOOST after the first round, with 20 models (each one roughly the size of four 70-sample windows) selected at random. It is interesting, however, that these accuracies are higher than the baseline accuracies of 32.2% and 29.9% in these two scenarios.

2.5 Online Novel Activity Detection

In this section we describe how GeTeM can be used to detect novel patterns (patterns on which it was not trained) in streaming data. We describe our approach in the activity recognition setting, but the method described is general and can be used in other settings. Our goal is to have a system that can be trained on a set of people performing a set of activities, and then tested by a new person who will perform some of the activities on which the system was trained, as well as novel activities that the system has not seen. The system should correctly classify the activities that it has seen, and recognize when a new activity is performed. When a new activity is detected, the system will prompt the user for a class label, and build a classifier for the new activity, on-the-fly, and add it to its ensemble of classifiers.

2.5.1 Novel Activity Detection Algorithm

We assume a training set consisting of traces of accelerometer data for each training subject performing each training activity. That is, we have data sets $D^{k,a}$ for each training subject $k = 1, \ldots, K$ and activity $a = 1, \ldots, A$. For each training set we select a subsequence $\mathcal{M}^{k,a} = \langle o_i \rangle$ which we use as the model for that subject-activity pair. We use the mean GeTeM score from all of the models for an activity as the score for that activity. More precisely, for a training segment o, we compute, for each $a \in \{1, \ldots, A\}$, a score $S^{(a)} = K^{-1} \sum_{k=1}^{K} S(\mathcal{M}^{(k,a)}, o)$. This is the same approach for computing scores as in the TDEBOOST algorithm (Section 2.4.1), except we average over a set of models.

The training set is split into two parts: the first is used to build models for each activity, the second is used to compute reference GeTeM scores for each of the models. Let Y denote the class label and X a set of GeTeM scores. The algorithm maintains an estimate of the conditional distributions $P(X|Y = y)$ and $P(X|Y \neq y)$ for each class y. For $P(X|Y = y)$ we use the empirical distribution of the reference scores $S^{(y)}$ from training segments in which the activity is y, and $P(X|Y \neq y)$ is computed as the empirical distribution of the scores $S^{(y)}$ from training segments where the activity is not y. For the purpose of our experiment, we assume a uniform prior $P(Y) = 1/A$ for the activities, and we use kernel density estimation with an automatically tuned Gaussian kernel to approximate all empirical distributions [67].

The algorithm proceeds by looking at each window of incoming data and computing likelihoods $P(Y = a|X) = P(X|Y = a)P(Y = a)/(P(X|Y = a)P(Y = a) + P(X|Y \neq a)P(Y \neq a))$ for each activity a in the set of known activities. If any of the likelihoods are greater than a threshold δ (we use 0.6 in our experiments), then the current window is assigned the activity corresponding to the maximum likelihood value. If none of the likelihood values are greater than δ, the activity is considered to be novel.

When a novel activity is detected, the current user k is queried for the class label a. If a is already in the set of known activities, this constitutes an error (i.e., a false positive with respect to detecting novel activities). To prevent this error from occurring again in the future, a new model $\mathcal{M}^{k,a}$ is built from the current window of data and added to the ensemble.

If the true label a is not in the set of known activities, this is considered to be a correct prediction and a new model $\mathcal{M}^{k,a}$ is constructed from the current window of data and added to the ensemble. In this case, estimates for $P(X|Y = a)$ and $P(X|Y \neq a)$ need to be constructed for the new activity a. We compute $P(X|Y \neq a)$ using the data in our training set, none of which correspond a. For $P(X|Y = a)$, we make the assumption that the subsequent window of incoming data also corresponds to the a, and compute $P(X|Y = a)$ as described above. In practice, it seems reasonable to assume that that once a person starts performing a new activity, they will continue to perform that activity for at least 20 seconds (twice the window length that we use for our experiments).

We also update the estimates for the conditional distributions $P(X|Y = y)$ and $P(X|Y \neq y)$ for each activity at each step, based on the most recent scores. In this way, we track the nonstationarity of the data, corresponding to fluctuations in the data. These fluctuations may correspond to a person getting tired and performing the activity at a slower pace, or increasing the pace of the activity after a warm-up period. We also maintain a second threshold $\delta' > \delta$. When the maximum likelihood score is between δ' and δ, we consider the prediction to have low-confidence. In this case, we build a new model for the activity from the window of incoming data, and add this new model to the ensemble for the currently-predicted activity.

2.5.2 Results

We evaluate our novel activity detection algorithm using the exercise routine data from Section 2.3.5. This data contains traces from four subjects performing five activities. We consider the four activities: *biking*, *elliptical*, *stairs*, and *treadmill*, and omit the subject who only performed two of the activities.

For the training set, we use data from two of the three subjects, and two of the four activities. Our test data is from the third user, performing all four of the activities. We use a sliding window of 10 seconds, overlapping by 2.5 seconds, and set $\delta = 0.5$ and $\delta' = 0.6$. For GeTeM we use the same parameters ($m = 5$, $d = 2$, $n = 2$, and $s = 16$) that were used in semi-supervised learning experiments. A window is considered to be correctly labeled if it is in the known set and we predict the correct activity, or if it is not in the known set and we predict that we do not know the activity. If an activity is predicted as unknown, we create the new model and consider it to be known for subsequent windows, as described above. Performance

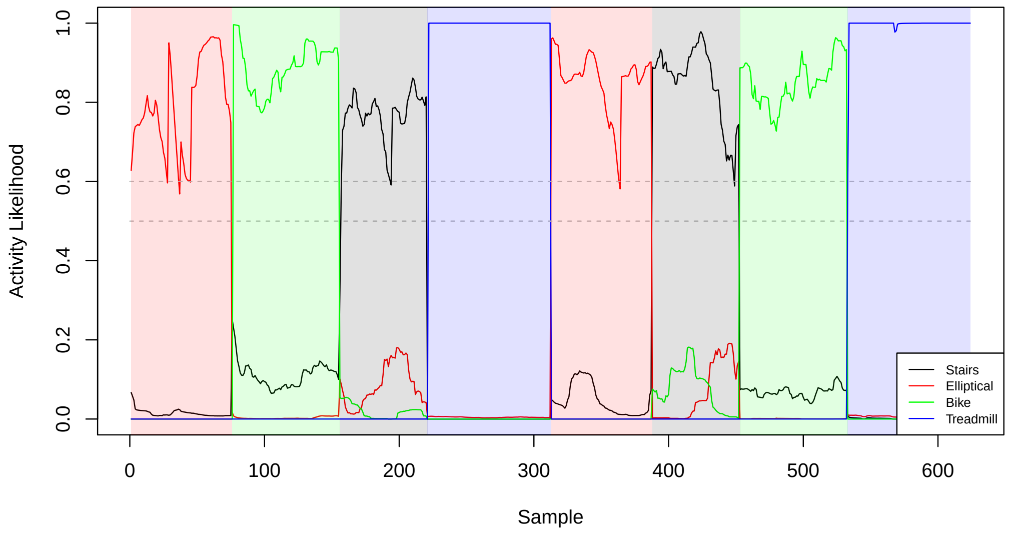


FIGURE 2.13: Likelihood scores for novel activity detection. The training set consists only of the *stairs* and *treadmill* activities. Dotted grey lines are the thresholds δ and δ', and the shaded colour represents the activity being performed (the true class label).

is measured by how many windows are correctly labeled, as well as by how many times we query the user for a class label. In our experimental setting, the user should be queried twice, as there are two unknown activities in the testing set.

We split each test set in half, so each activity is presented twice, and shuffle the order in which the activities are presented. We ran the experiment 300 times, and the accuracy was 74.29% with a standard deviation of 23.31%. The average number of times the user was prompted was 3.23 with a standard deviation of 2.12. The average number of additional models constructed was 9.83 with a standard deviation of 10.02. We note that the standard deviation for all three performance measures are high; the algorithm's performance depends largely on the order in which the activities are presented and the initial models.

Figure 2.13 shows an example of the output of the algorithm. The training set consists of users 1 and 2 and activities *stairs* and *treadmill*, and the test set consists of all four activities for user 3. The likelihoods for each activity are shown as the coloured lines, and the thresholds δ and δ' are shown as the dotted grey lines. The shading indicates the activity being performed. The first two segments were for the *elliptical* and *bike* activities, respectively, both of which were unknown and correctly detected. During the first segment, the likelihood dropped below the threshold δ' 3 times, causing 3 additional models to be built for the *elliptical* activity. In the fifth and seventh segments, when the activities *elliptical* and *bike* occur again, they are properly recognized, without querying the user. The likelihood drops below δ' a total of 7 times, leading to the construction of 7 additional models, but at no time drops below δ, meaning that the user is only queried twice, when the novel activities are first performed.

2.6 Summary

In this chapter, we presented the GeTeM algorithm, and evaluated its use as a distance measures for time series in unsupervised, semi-supervised, and supervised learning. GeTeM compared very favourably to other commonly used time series distance measures, such as those based on DTW, DFT, and DWT. On the ECG data set in Section 2.3.1, the GeTeM-based distance is the only method able to correctly cluster the data. On the large benchmark data set from the UCR time series repository [1], the improvements in accuracy of GeTeM over DTW, and GeTeM over Euclidean distance, when used in a nearest-neighbour classifier, are both statistically significant ($\alpha = 0.05$). Combined with a spectral clustering algorithm, GeTeM can be used to analyze and visualize unlabelled traces of data consisting of multiple subjects performing multiple activities, and the structure discovered by this technique can be used in a semi-supervised setting where unlabelled training data can be used to improve the performance of a classifier.

We also presented a boosting algorithm, TDEBOOST, that uses GeTeM for computing features from long time series. For larger data sets that are not segmented a priori, TDEBOOST is able to locate segments on which to build models for extracting GeTeM features. TDEBOOST outperforms two strong baseline algorithms on the difficult problem of gait recognition when trained and tested with clothing and footwear variations.

Finally, GeTeM can be used to detect novel patterns in streaming time series. We present an algorithm for performing novel activity detection in data collected from an accelerometer, and demonstrate it's ability to recognize physical activities, and detect new activities on which it has not been trained. While the performance varies greatly on the intial models, on average it performs quite well.

GeTeM is not meant to be a replacement for DTW or other approaches for comparing time series. It makes different assumptions about the process that generated the data, and is therefore more suited for problems where the data corresponds to measurements of a latent, periodic, nonlinear dynamical system. However, GeTeM is nonetheless robust to violations of these assumptions. GeTeM is also computationally efficient (for example, it takes less computation time than DTW), and thus suitable for real-time analysis of streaming data. GeTeM should be considered a powerful addition to the suite of tools that a time series analyst has at their disposal.

Chapter 3

Location Detection

In this chapter we consider *wifi localization*, the problem of determining the location of a device from observable wifi signals. In contrast to using GPS, the de facto approach for localization, wifi signals have the advantage of working well indoors, and in environments such as dense metropolitan areas where GPS is unavailable or noisy. A specific form of the localization problem is that of *positioning*, or determining the physical coordinates of a device. Positioning using wifi signals is well-studied, and commercial applications are widespread and frequently used on many popular mobile devices. We consider a slightly different problem, where the goal is not to place the user on a map, but to predict the label that a user would use to describe the location they are currently in. We call this *subjective location detection*, as the labels and the places where those labels apply can be different for each user.

We present a novel, efficient algorithm for clustering wifi signals to build fingerprints for locations. A semi-supervised learning algorithm is presented that uses both labeled and unlabelled data to associate with fingerprints user-defined labels. We evaluate our algorithm on a data set collected over the course of more than a year, consisting of over 6 million wifi measurements.

This chapter is organized as follows. First, in Section 3.1, we place our approach in the context of the existing literature, and present the necessary background material. We describe the subjective location detection problem in Section 3.2. In Section 3.3, we describe the clustering and labelling algorithms. We evaluate our approach in Section 3.4, and conclude with a discussion and remarks on future directions for this work, in Section 3.5.

3.1 Background and Related Work

In this section, we present the necessary background material for this chapter. We begin with an overview of existing work on localization on mobile devices and a practical account of issues that arise when dealing with wifi signals, thus motivating the alternative model we present in this chapter. This is followed by a review of the hierarchical Dirichlet process (HDP), and the online extension of the HDP introduced by Wang et al. [7], on which our algorithm is based.

3.1.1 Localization of Mobile Devices

Localization is the process of determining the location of a device. The most common approach to localization is *positioning*, where one determines the physical coordinates of a device, which can be used to place the device on a map. In the context of positioning, location refers to a point or region in physical space. Many systems have been proposed for device positioning, using RF signals, ultrasound, infrared, vision, or even smart-floors, for example (see [68] for a thorough survey). Here, we limit our discussion to technologies whose hardware requirements are met by typical commercial mobile devices, namely GPS, GSM, and wifi.

3.1.1.1 GPS

The canonical example of a positioning system is the global positioning system (GPS), which provides accurate location (in the positioning sense) and time anywhere on Earth with an unobstructed view of at least four of a network of approximately 31 satellites[1]. Satellites continuously transmit their location and a highly-accurate timestamp, which are used, with the aid of multilateration, to compute an accurate approximation of the position of a receiver. Roughly speaking, multilateration approximates the position of a mobile device by estimating the difference in distance from the device to two or more objects whose coordinates are known[2]. In the case of GPS, these objects are the satellites and the difference in distance is estimated by measuring the difference between timestamps received at the same time from multiple satellites and multiplying by the speed of light. When signals from four or more satellites are available, the position of the receiver in 3D can be uniquely determined. In practice, signals from more than four satellites are used to improve the quality of the approximation.

While GPS-derived locations can be accurate in the range of a few centimetres [69], the requirement for line-of-sight to satellites precludes its use indoors. Additionally, in dense metropolitan areas signals can be scattered by buildings and reception is often significantly degraded (often referred to as multipath conditions [70]).

3.1.1.2 GSM

An alternative method for positioning, aimed at mobile phones, is to use multilateration of signal strengths to nearby cell towers. Commonly referred to as GSM

[1]Old satellites are occasionally decommissioned and new ones are launched, and so the number of active satellites can vary.

[2]Multilateration differs from trilateration, which estimates the absolute distance between a receiver and a transmitter with a known location. An advantage of multilateration is that it does not require the clocks of the device and the transmitter to be perfectly synchronized, only those of the transmitters.

localization[3], research into this approach was heavily spurred by the USA Federal Communications Commission's 1996 enhanced 911 (E911) [71], and the EU's 2003 enhanced 112 (E112) [72] mandates. Both mandates place requirements on how accurately cellular network providers have to be able to locate devices, to allow for more effective dispatching of emergency responders. For example, E911 requires wireless providers to be able to approximate the location of any mobile phone to within 100 meters of it's actual location for at least 67% of all 911 emergency calls.

Various approaches for GSM localization are used in practice: the coarsest approximation simply places the mobile device within the cell covered by the tower with the greatest signal strength, which has an accuracy between 100m and 3km; more accurate approximations based on multilateration, such as the enhanced observed time difference (E-OTD) can achieve an accuracy of approximately 30m in suburban commercial and residential environments, for a stationary device [73]. In addition to problems like multipath conditions that also effect GPS signals, GSM signals are susceptible to interference from nearby devices.

One significant advantage of GSM localization as compared to GPS is that, for mobile phones, it does not require additional hardware. Mobile phones frequently communicate with nearby cellular towers, so GSM localization can be performed without consuming power above and beyond the computation required for multilateration. GPS, on the other hand, requires specialized hardware; thus, GPS consumes considerably more power. Another advantage is that, as communication between the device and the towers is two-way, GSM localization can be performed both on the device and on the network side[4].

[3]As GSM (2G) is currently the most common digital cellular network communication standard, the term GSM is often used, broadly, to describe cellular networks, including those employing more recent standards such as 3G and 4G.

[4]From the privacy perspective this may not necessarily be seen as an advantage

3.1.1.3 Wifi

With the proliferation of wireless networks in urban areas, wifi has become a popular tool for positioning. The term wifi is used to describe wireless networks that operate on the 2.4 GHz and 5 GHz bands and conform to the IEEE 802.11 family of standards [74]. Localization using wifi signals is an active area of research [75–86] A number of commercial wifi positioning products exist, from vendors such as SkyHook Wireless[5], Google[6], and Apple[7], that allow for device positioning with reported accuracies of around 10-20m, and 100% coverage of most urban areas in North America and Europe[8]. Specialized hardware from vendors such as Cisco[9] and Nokia [87] allow for mapping of indoor environments, and companies such as Ekahau[10] and Point Inside[11] offer indoor localization services that integrate with existing wifi networks.

The main difference between positioning with GSM signals and with wifi signals is that the locations of GSM towers are known and do not change, as they are installed and maintained by the cellular providers. Wifi access points (APs), on the other hand, are generally installed ad-hoc, and can be easily moved. In addition, multilateration requires careful synchronization of the clocks in the beacons, which, for cellular towers, is typically facilitated via GPS. APs generally do not have the required hardware to enable accurate synchronization, and are typically found indoors where GPS reception is unavailable. Finally, wifi signal ranges are much smaller than GSM signal ranges, and the distance to what is considered a "nearby" AP (generally less than 30m) is much smaller than the distance to a "nearby" cellular tower (anywhere from 100m to 3km). Therefore, computing distances

[5]`http://www.skyhookwireless.com/`

[6]`http://www.google.com/maps/`

[7]`http://www.apple.com/`

[8]According to information retrieved from `http://www.skyhookwireless.com` on June 13, 2012.

[9]`http://www.cisco.com/go/location`

[10]`http://www.ekahau.com/`

[11]`http://www.pointinside.com/`

based on time differences for wifi signals would require sub-nanosecond accuracy to be useful[12]. Due to these factors, multilateration using wifi signals is rarely used.

Approaches for positioning using wifi signals are typically categorized [68] as those that explicitly build a signal map (examples include RADAR [76, 77], Nibble [75], Horus [78], and others [79, 80, 86]), and those that rely on a signal propagation model (examples include ARIADNE [81], EZ [82] and others [83, 84]). While we use this terminology for consistency, we note that the distinction is not clear-cut, as map-based approaches typically make assumptions about signal propagation to interpolate between points on the map where measurements are taken. A more appropriate distinction might be parametric versus nonparametric. Most map-based approaches are nonparametric, in that they are typically data-driven and employ techniques from nonparametric statistics such as nearest neighbour and kernel-based methods. Signal modelling techniques still build an implicit map, however they tend to rely on parametric models of signal propagation.

Signal Mapping

RADAR, from Microsoft© Research [76, 77], was one of the earliest wifi localization projects, and actually predates the term wifi. Three wireless base stations were placed in known locations in a 980 sq. m single-floor office containing over 50 rooms. As is typical for map-based approaches, a data-collection phase (referred to as the off-line phase) was performed where, at each of 70 locations, measurements of the wifi signals were collected and associated with the physical location. Although not specified in the paper, judging by the rough map of the locations, all but 4 locations were in hallways and roughly all locations were such that there was a direct line-of-sight to at least one of the base stations. It was noted that the measured signals vary “quite significantly” depending on the orientation of the person holding the data collection device, and so measurements were taken

[12]Even the atomic clocks on the GPS satellites can have discrepancies of a few nanoseconds.

with the user facing in each of the directions: north, east, south, and west. At least 20 measurements were taken for each location and each orientation, and the received signal strengths (RSSs) for each of the three base stations were averaged for each unique location, orientation pair. This resulted in 280 measurements, which were treated as expected values of the RSS for each location and orientation. Measurements were represented in a 3-dimensional Euclidean space, and a one-nearest-neighbour classifier (1-NN) was used to predict the current location. The reported error distance was less than 3m for 50% of the locations [76] and subsequent improvements reduced the median error to 1.3m [77].

We detail the RADAR system above because it is prototypical of map-based systems. Two phases are defined: an off-line calibration phase, in which RSS measurements are taken at known locations to build a wifi signal map, and an online estimation phase, in which RSS fingerprints from a device are used for positioning. With 1-NN, as used in RADAR, the signal map is a Voronoi partitioning of the physical map. Probabilistic models such as Bayesian networks, kernel density estimation, and histogram-based approaches have also been used, with similar results [75, 78, 79]. Horus [78] uses a combination of techniques including temporal smoothing and locally-weighted regression, and was shown to outperform RADAR and the system proposed by Roos et al. [79].

While the exact algorithms used in commercial products from Google©, Apple©, and Skyhook© are not published, it is well-known that each of these companies employs a fleet of vehicles that drive around cities, recording AP and RSS measurements[13]. Assuming good coverage in the calibration step, these methods can achieve accuracies within a few meters. However, the calibration step can be expensive, if large areas are to be covered, and as APs are installed, removed, replaced, and moved fairly often, keeping the map current and up to date is challenging. Commercial vendors have more recently begun using their customers'

[13]This process is often known as “war driving”.

devices to maintain current estimates, collecting wifi signal measurements from mobile phones when their GPS receivers are enabled and have an accurate fix[14].

Signal Modelling

An alternative to building a map from measurements is to compute an expected signal map by knowing the locations of APs and modelling the propagation of signals from these APs. Assuming that RSS can be expressed as an invertible function of distance, then from measurements, the distance to each AP can be computed, and trilateration can be employed to compute a position estimate. The models for signal propagation can be quite complex, incorporating both path loss (natural loss in signal strength due to distance), and penetration loss (loss in signal strength due to physical barriers such as walls, doors, windows, etc.). However, they can be quite accurate, and avoid the time-consuming calibration step required by map-based approaches. Signal modelling-based approaches tend to be less accurate map-based approaches, but make up for the loss in accuracy by avoiding the expensive calibration phase.

Deasy and Scanlon [84] used the propagation model from Durgin et al. [88], and were able to achieve results within a meter or two of a map-based approach calibrated by collecting samples on a 2m resolution grid (335 measurement locations in total for the two environments). While the propagation model requires an accurate floor-plan of the area, in order to compute the penetration loss due to walls and other obstacles, the map-based approach, assuming measurements require one minute per location, requires over 5.5 hours for calibration in addition to requiring a floor-plan.

Cavalieri [83] performed a thorough comparison of signal propagation models in outdoor environments. They compared a signal model from Lee and Lee [89] (similar to that of Durgin et al. [88], except it model separately the case where

[14]For example, Navizon© offers cash rewards for submitting wifi measurements with corresponding GPS data: `http://www.navizon.com/navizon_rewards.asp` (Retrieved June 22, 2012)

there is line-of-sight to the AP and the case where there is not), and a ray-tracing approach [90] that models such phenomena as reflection and diffraction of signals. The approach used a Kalman filter [91] to smooth sequential estimates taken by a moving device. Similar to the approach of Deasy and Scanlon [84], with an accurate map of the environment the signal propagation models produced estimates accurate to within a few meters of those produced by a map-based approach and in the dense urban environment in which the methods were evaluated, produced estimates accurate to within a few meters of GPS-based estimates as well. Deasy and Scanlon [84] report, however, that a map-based approach, despite requiring more expensive calibration, outperformed their best signal-modelling approach by 30% in terms of RMS localization error.

The EZ system, also from Microsoft Research [82] uses a much simpler propagation model, called the log-distance path loss model (LDPL), that posits a logarithmic scaling of the signal strength with respect to the distance between the user and the AP, and ignores penetration loss. By treating the AP signal transmission power, the path loss rate, and the distances as unknowns, and setting up a system of LDPL equations, EZ is able to learn a signal propagation model without a priori knowledge of the AP locations. EZ requires measurements from five known locations per AP, and uses additional measurements from unknown locations to improve the estimates. EZ, calibrated with measurements from 15 known locations was compared to Horus and RADAR, both calibrated with hundreds of measurements at known locations. Horus performed best with median error of 4m, RADAR had a median error of 5m, while EZ, which required significantly less calibration effort, had a median error of 7m.

Practical Issues

Wifi measurements gathered by a mobile device are noisy. Most existing work uses high-quality measurements devices, often with antennas. Even with specialized equipment, map-based approaches tend to require at least 20 measurements from a single location to obtain a sufficient signal fingerprint for a location. In the RADAR work [76, 77], it is noted that changes in orientation (i.e., changing whether the person carrying the device lies between the device and the AP or not), lead to substantial changes in the signal at location. In this work, we are interested in measurements from mobile phones, which are often carried in a pocket or a bag, and thus the issues of noise are substantially magnified.

Noise can arise from a number of sources. Along with other mobile phones, sources of interference includes microwave ovens, cordless phones, Bluetooth devices, wireless video cameras, outdoor microwave links, wireless game controllers, Zigbee© devices, fluorescent lights, and WiMAX© devices [92]. This type of interference presents itself as variations in RSS measurements, and is similar to noise present in typical scenarios involving RF signals, where it is modeled as a white noise process, typically Gaussian [79, 80, 86].

Another type of noise that is far more insidious, but common in real wifi measurements, is the appearance and disappearance of visible APs in subsequent measurements, even for a stationary device. Naturally, APs will appear and disappear when they come into, and go out of the range of the signal, and it would be expected that APs near the maximum range might oscillate between being present and absent in subsequent measurements. However, we regularly observe the situation where an AP with a high RSS at one instant will not be present in a subsequent measurement, or, conversely, an AP not present at one instant will become present with a high RSS at the next.

The IEEE 802.11 standard divides the wireless spectrum into a set of channels [74] that vary in number for different geographic locations. For instance, in Canada and the US, the 2.4 GHz range (the most commonly used range) has 11 channels, Japan has 14, and most of the rest of the world has 13. To generate a wifi measurement, a device tunes to each channel in turn and broadcasts a probe. It then listens for a period of time for any responses from any APs currently using that channel.

For a wifi measurement to be generated in a timely manner, the amount of time spent listening for responses on each channel is quite limited. For instance, in our experience with the Google Android© mobile operating system, performing a wifi measurement takes between 0.5 and 2 seconds to scan the 11 channels. In environments with many wireless networks, multiple APs will share a channel, and it is possible for responses to be delayed if the channel is currently in use. In busy networks, APs may have a backlog of network traffic, and may simply not process the probe until it is too late. Due to possible delays, as well as the fact that noise can cause both probes and responses to be lost in transit, it is common that AP will not be observed during a scan, and thus not appear in the corresponding measurement. We note that recent work [86] has considered using information from the PHY network layer instead of RSS. While this appears to be a more robust measure of signal strength than RSS, hardware support is extremely limited. However, this is certainly a technology to keep an eye on for more robust wifi localization, as it provides signal strength information in every packet, rather than relying on a probe-response protocol.

We placed an HTC Nexus One© mobile phone on a desk, where it remained stationary and collected wifi measurements at 30 second intervals for six days. The results are displayed in Figure 3.1 and Figure 3.2. While a total of 158 unique APs were observed, the mean number of APs per measurement was approximately 23.1 with a standard deviation of approximately 4.6. From the gaps in every row of Figure 3.1, it is clear that access points appear and disappear regularly.

Figure 3.2 shows the number of visible access points in each measurement (left), and the differences in the number of access points in subsequent measurements. Of the 17,607 measurements, only 1,249 (7.1%) subsequent pairs of measurements had the same number of visible APs, while $3,833$ (21.8%) subsequent pairs differed by at least 5 in the number of visible APs. Figure 3.3 depicts how often APs appear in measurements. We see that of the 158 unique APs that were observed in the 17,607 measurements, approximately 70% of them appeared in fewer than 10% of the

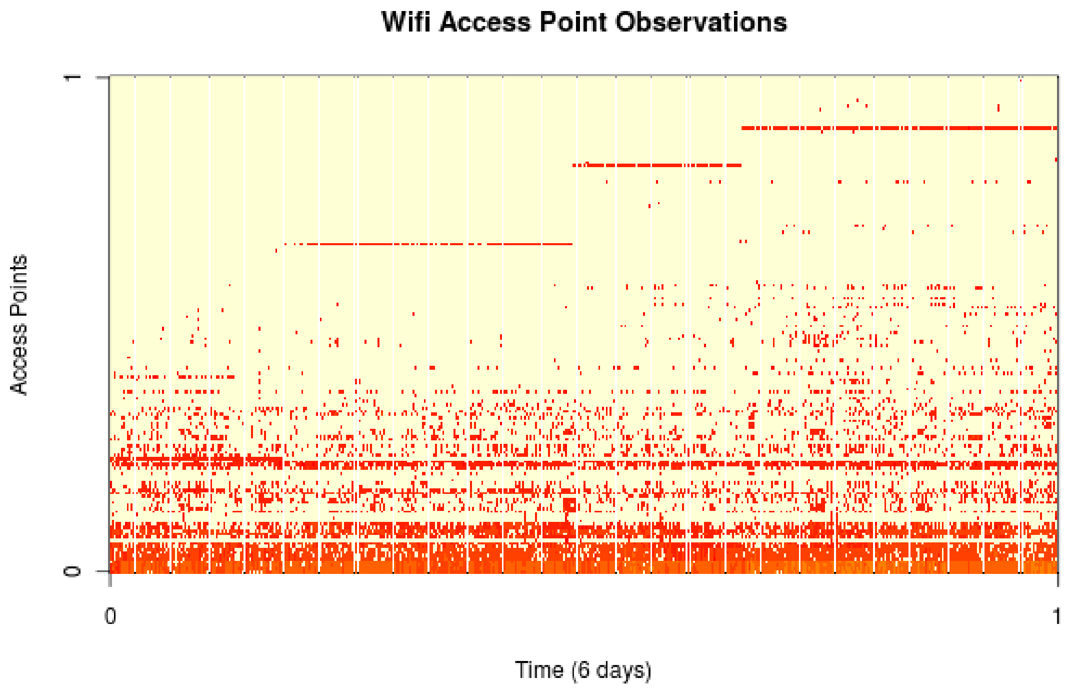


FIGURE 3.1: Wifi measurements over a one-week period. The horizontal axis corresponds to time, and each row depicts the observations for a particular access point. The shading represents the received signal strength (RSS), with darker corresponding to a higher RSS and the lightest shading corresponding to the access point not being present.

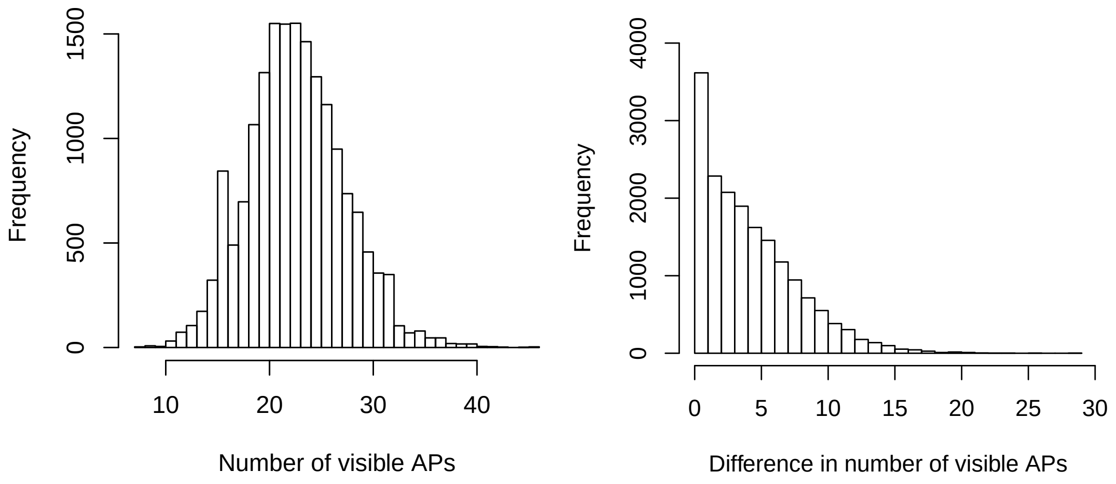


FIGURE 3.2: Number of observable access points in each measurement (left) and the difference in the number of visible access points in subsequent measurements (right) for a stationary device measuring at 30 second intervals for six days.

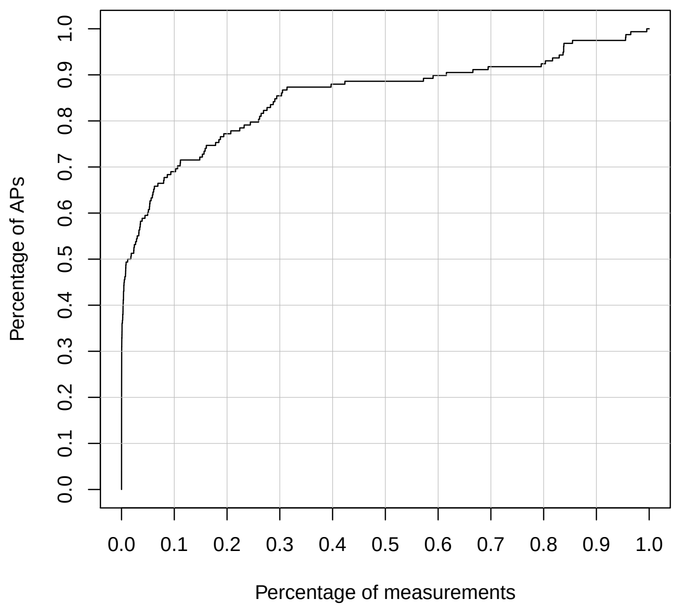


FIGURE 3.3: CDF of the number of APs appearing in each measurement for a stationary mobile phone, taken over a 6-day period. A point at coordinate (x, y) indicates that y of the observed APs appear in at most x measurements. For example, we see that approximately 70% of the APs appear in fewer than 10% of the measurements, and 90% of the APs appear in fewer than 60% of the measurements.

measurements, and 90% of them appeared in fewer than 60% of the measurements. Only four of the APs appeared in more than 90% of the measurements.

A Shapiro-Wilks normality test [93] rejects all of the 158 signals as having a Gaussian distribution with $p < 10^{-40}$. If we only consider the non-zero valued signals strengths from each AP, the Shapiro-Wilks test rejects 127 of the 158 APs (80.3%) as having a Gaussian distribution, at significance level $p < 0.001$.

When measurements are represented as points in an N-dimensional space (where N is the total number of observed access points) and Euclidean distance is used as a measure of similarity between measurements, missing access points become problematic. If missing values could be detected, one could attempt to impute the missing values [94]. The issue is that there is no way to determine, in a measurement, if an access point is out of range (effectively has RSS of 0), or just absent from the measurement, but would have a non-zero RSS if it were present.

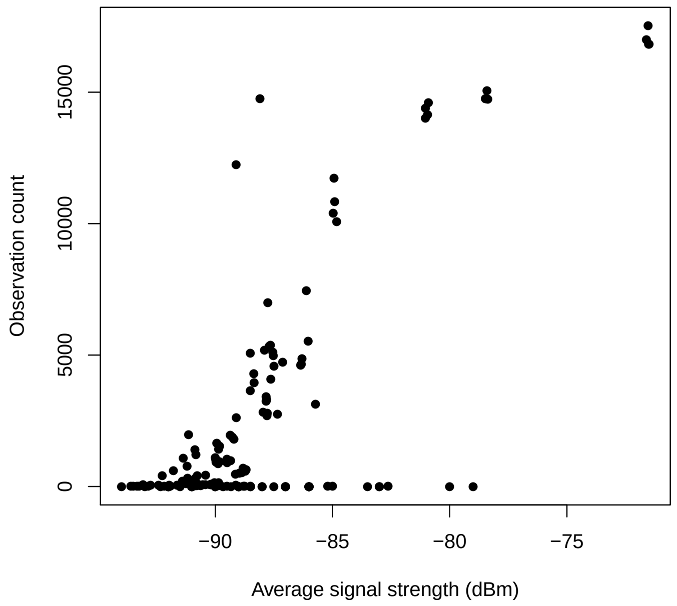


FIGURE 3.4: Relation between average received signal strength of an access point and the number of times it was observed over a 6-day period.

Youssef and Agrawala [78] proposed ignoring access points with RSS below a signal-strength threshold and demonstrated that access points with RSS below the receiver sensitivity were the ones that appeared and disappeared more often. Figure 3.4 shows the relationship between the average RSS from an access point and the number of times it appeared in the 17,706 measurements. While previous work considered only 7 access points and a 5-minute scanning interval (300 measurements) [78, Figure 3], we see that over a longer period (17,607 measurements), with more access points (158), the monotonic relationship between RSS and observation count is not as clear, nor is the rapid drop in observation counts that they observe around the receiver sensitivity level. Instead, while we certainly see a fairly prominent correlation between average RSS and observation count ($r = 0.756$), we also see that many access points with large RSS values are observed infrequently. In fact, 25 of the 158 access points had average RSS higher than the median average RSS (-89 dBm) but appeared in fewer than 1% of the observations. Therefore, simply thresholding the RSS values in order to remove intermittent APs will not be a reliable approach, nor is it clear how to select a threshold.

This leads to problems when using Euclidean distances or Gaussian kernels in signal space, as a missing value for an AP that typically has a high RSS can contribute a large amount to the distance or dissimilarity. An obvious solution to this problem is to average over many measurements. For example, Chintalapudi et al. [82] averaged readings over a 5 minute period for each location. However, if we wish to have a responsive positioning system, we cannot expect the user to remain stationary for 5 minutes in every location in order to detect where they are. Instead, in Section 3.3.1, we propose to use a multinomial distribution, such that missing values are penalized less significantly. To further motivate the use of a multinomial distribution over the traditional Gaussian model, we consider the data from the stationary device. We fit both a multinomial and a Gaussian model to a small subset of the data (training set), and then observe the perplexity over the remaining data (test set). Let q be the model fit to the training data, then the perplexity is $2^{-\frac{1}{N}\sum_{i=1}^{N}\log q(x_i)}$, where $x_1, \ldots, x_N$ are the test data. Informally, the lower the perplexity, the less "surprised" the model is by the test data, indicating a better model.

As the perplexity values between models aren't directly comparable (the Gaussian model has roughly twice the number of parameters), we instead observe how the likelihoods change as we decrease the size of the training set. Figure 3.5 shows the log-relative increase in the perplexity as we decrease the size of the training set from 1,000 to 1, averaged over 40 runs, each with different randomly selected training sets. Let p_n be the perplexity with n training examples. Figure 3.5 shows the relationship between $\log(p_n/p_{1000})$ and n. It is clear that the quality of the Gaussian model degrades much more rapidly than the multinomial model, keeping in mind that the vertical axis represents logarithmic units. While the Gaussian model is a good fit when there is lots of training data, the multinomial model is a better fit when there is limited training data.

One final issue that effects the performance of wifi positioning systems in practice is receiver gain differences. In the same location, two devices may report different

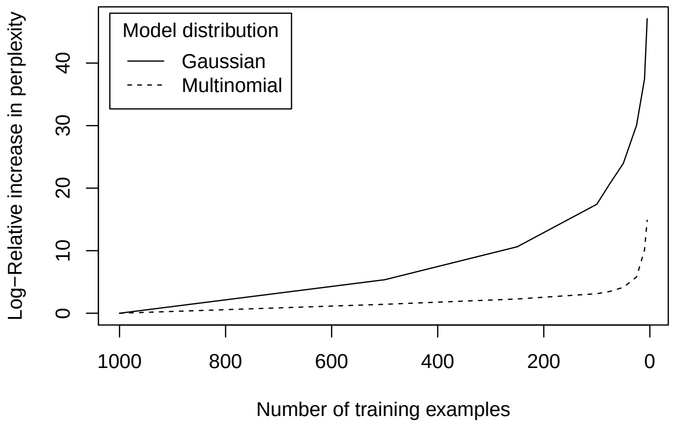


FIGURE 3.5: Log of the relative increase in perplexity for Gaussian and multinomial models of wifi signals for a stationary device, as the number of training examples is reduced from 1,000 to 1.

RSS values due to hardware differences [80, 82, 85, 95]. Different makes and models can report differences as large as 20 dBm, and gain differences of between 2-7 dBm have been observed even among devices of the same make and model [82]. Note that an increase of 3 dBm is roughly equivalent to doubling the power received by the device. Fortunately, the differences in hardware lead to a linear scaling in the RSS, and various approaches to calibrate devices have been proposed [80, 82, 85]. The model proposed in Section 3.3.1 is based on representing signals as discrete counts, making it invariant to scaling and thus robust to receiver gain differences.

3.1.2 Online Hierarchical Dirichlet Process Algorithm

In this section, we present the HDP model [96] and the online algorithm of Wang et al. [7], on which our streaming algorithm is based. The HDP is a Bayesian nonparametric mixture model, typically used for topic modelling of text data, and can be interpreted as a nonparametric extension of latent Dirichlet allocation (LDA) [97]. For a more detailed treatment of these topics, we refer the reader to Müller and Quintana [98], Walker et al. [99], Jordan [100, 101], and Fox [102, Chapter 2]. The latter is the most thorough, textbook-like treatment of these topics that we have come across. For notation, we follow that of the previously-cited reference material and use $x \sim X$ to indicate that the random variable x is drawn from a distribution X, and $x|z \sim X(z)$ when X is a distribution conditioned on a (possibly random) quantity z. We make the simplifying assumption that all stated conditional distributions exist, and we use the terms distribution and density interchangeably.

3.1.2.1 The Dirichlet Distribution

Let $p(\boldsymbol{y}|\theta)$ denote a probability model for a set of observations $\boldsymbol{y}$ given a set of model parameters θ. The standard Bayesian approach places a prior distribution on θ, with the goal of predicting future data while incorporating those already observed. Given a set of N i.i.d. observations $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, the *predictive likelihood* of future observations is given by:

$$p(\boldsymbol{y}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N, \lambda) = \int_\Theta p(\boldsymbol{y}|\nu) p(\nu|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N, \lambda)\, d\nu,$$

where Θ is the space of parameters and λ is the parameter of the prior, generally referred to as a *hyperparameter* as it is typically not learned, but treated as a tuning parameter whose value is specified in advance. One may also be interested in the distribution over the parameters θ given the appropriate hyperparameters and a set of observations. This distribution is referred to as the posterior density on θ, and is given by:

$$p(\theta|\boldsymbol{y}, \lambda) = \frac{p(\boldsymbol{y}|\theta)p(\theta|\lambda)}{\int_\Theta p(\boldsymbol{y}|\nu)p(\nu|\lambda)\, d\nu}.$$

Let y denote a random variable that can take on values in $\{1, \ldots, K\}$, and let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ be an associated mass function where that π_i represents the probability of observing the value i. The *multinomial distribution* describes the probability of a sequence of observations $y_1, \ldots, y_N$:

$$p(y_1, \ldots, y_N | \boldsymbol{\pi}) = \frac{N!}{\prod_k N_k!} \prod_k \pi_k^{N_k},$$

where we use the shorthand N_k to represent the number of times k occurs in the sequence. We denote the multinomial distribution by $\text{Mult}(\boldsymbol{\pi})$. Note that the distribution is invariant to the order of the observations. When $K = 2$, this is referred to as the *binomial distribution*.

An oft-used convenience in the Bayesian framework is that of a *conjugate prior*, which provides a computationally tractable mechanism for maintaining a posterior distribution over θ while incorporating new observations. Let P_Λ be a family of distributions, a prior distribution $p(\theta|\lambda) \in P_\Lambda$ is said to be a conjugate prior for a distribution $p(\boldsymbol{y}|\theta)$ if the posterior $p(\theta|\boldsymbol{y}, \lambda)$ is also in P_Λ.

The Dirichlet distribution is a conjugate prior for the multinomial distribution. That is, samples from a Dirichlet distribution can be treated as parameters for a multinomial distribution, and the posterior distribution remains a Dirichlet. A K-dimensional Dirichlet distribution is defined by a set of positive-valued hyper-parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$, and has the form:

$$p(\boldsymbol{\pi}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \pi_k^{\alpha_k - 1},$$

where $\Gamma(\cdot)$ represents the standard Gamma function. We refer to this distribution as $\text{Dir}(\alpha_1, \ldots, \alpha_k)$. When the parameters $\alpha_1, \ldots, \alpha_K$ are all equal, this is referred to as a symmetric Dirichlet. A special case of this distribution, when $K = 2$ is called the beta distribution which we denote by $\text{Beta}(\alpha_1, \alpha_2)$.

The Dirichlet distribution has a number of computationally convenient properties. Given N multinomial observations $y_1, \ldots, y_N$, the posterior distribution of $\boldsymbol{\pi}$ is:

$$p(\boldsymbol{\pi}|y_1, \ldots, y_N, \boldsymbol{\alpha}) \propto \text{Dir}(\alpha_1 + N_1, \ldots, \alpha_K + N_K),$$

and the predictive likelihood is:

$$p(y = k|y_1, \ldots, y_N, \boldsymbol{\alpha}) = \frac{N_k + \alpha_k}{N + \sum_{i=1}^{K} \alpha_i},$$

where N_k denotes the number of times k occurs in the observations.

3.1.2.2 The Dirichlet Process

Stochastic processes extend the notion of a probability distribution to more general domains such as function spaces, spaces of probability measures, etc. The Dirichlet process is a stochastic process whose domain is the space of probability measures. Given a measurable space Θ, a base measure H on Θ, and a *concentration parameter* γ, a measure G drawn from a Dirichlet process, denoted by $\mathrm{DP}(\gamma, H)$ has the property that for any finite partition $\{A_1, \ldots, A_K\}$ of Θ:

$$(G(A_1), \ldots, G(A_K)) \,|\, \gamma, H \; \sim \; \mathrm{Dir}(\gamma H(A_1), \ldots, \gamma H(A_K)).$$

It follows that for any $A \subset \Theta$, $\mathbb{E}\left(G(A)\right) = H(A)$, where $\mathbb{E}(x)$ denotes the expected value x.

The existence of a Dirichlet process $\mathrm{DP}(\gamma, H)$ for any concentration parameter λ and base measure H was first proved by Ferguson [103]. Later, a constructive proof was provided by Sethuraman [104], which relies on the stick-breaking process. There are a number of representations of the Dirichlet process, for example the Pólya Urn [105] and the Chinese Restaurant Process [106]. However, as we refer to the stick-breaking process later in this chapter, we will focus on that construction.

The stick-breaking process defines an infinite partitioning of the unit interval, which can represent a probability mass function on a countably infinite set. Let

$\boldsymbol{\beta} = (\beta_k)_{k=1}^{\infty}$ define a sequence of discrete probabilities:

$$\begin{aligned} \beta'_k \,|\, \gamma &\sim \text{Beta}(1, \gamma) \quad k = 1, 2, \ldots \\ \beta_1 = \beta'_1, \quad \beta_k &= \beta'_k \prod_{i=1}^{k-1} (1 - \beta'_i) \quad k = 2, 3, \ldots. \end{aligned} \tag{3.1}$$

This process can be seen as starting with the unit interval, breaking off a random proportion β'_1, then taking the remainder and breaking off a random proportion β'_2, and so on, hence the name. We denote the stick-breaking process with parameter γ as $\text{Stick}(\gamma)$.

Sethurman [104] showed that with probability one, a draw from a Dirichlet process, $G \sim \text{DP}(\gamma, H)$ can be expressed as:

$$\begin{aligned} \theta_k \,|\, H &\sim H \quad k = 1, 2, \ldots \\ G &= \sum_{k=1}^{\infty} \beta_k \delta_{\theta_k}, \end{aligned}$$

where δ_θ denotes a point mass on θ. In other words, the distribution G is a discrete distribution over elements of H, whose support is a countably infinite set of atoms $(\theta_k)_{k=1}^{\infty}$. If H is a continuous distribution[15], then two samples from H are equal with probability 0, whereas two samples from G can take the same value.

Similar to the Dirichlet distribution, the Dirichlet process has a convenient posterior. Given a set of observations $\theta'_1, \ldots, \theta'_N$, drawn i.i.d. from the distribution G, the posterior distribution is given by:

$$G \,|\, \theta'_1, \ldots, \theta'_N, H, \gamma \;\sim\; \text{DP}\left(\gamma + N, \frac{\gamma}{\gamma + N} H + \frac{1}{\gamma + N} \sum_{i=1}^{N} \delta_{\theta'_i}\right).$$

The number of unique values in N draws from G scales roughly as $\gamma \log N$. This gives us insight into the effect of the concentration parameter γ: for small values

[15]Here and throughout this chapter we use continuous to mean absolutely continuous with respect to the Lebesgue measure.

of γ, we would expect to see the majority of the mass in the first few weights β_i, and thus we should see the same values frequently if we were to sample from G repeatedly; for large values of γ, we would expect the mass to be more evenly distributed between the weights β_i, and thus we should to see a wider variety of values if we sample from G repeatedly.

3.1.2.3 The Hierarchical Dirichlet Process

The HDP is a model for groups of data that are assumed to be generated by related but distinct generative processes. A motivating example is *topic modelling.* Assume the following generative story for documents, which are treated as a bag of words:

- There exist an infinite set of topics $\theta_1, \theta_2, \ldots$ defined as distributions over a vocabulary.
- Associated with each document D_j is a distribution over topics G_j and length N_j.
- D_j is generated by N_j times sampling a topic $\theta | G_j \sim G_j$ and a word $w | \theta \sim \theta$.

A document, then, is made up of words associated with a set of topics. For example, this section is made up of words from a mixture of topics including math, statistics, and probability theory.

Defining this generative process in terms of probability distributions, first the base distribution H is defined as a symmetric Dirichlet over the vocabulary simplex (i.e., draws from H represent parameters for a multinomial distribution over the vocabulary). Let w_{ji} be the ith word in document j, let M denote the total number of documents, and let N_j be the number of words in document j. The HDP model is defined as:

$$\begin{aligned} G_0 \,|\, \gamma, H &\sim \mathrm{DP}(\gamma, H) \\ G_j \,|\, \alpha, G_0 &\sim \mathrm{DP}(\alpha, G_0) \quad j = 1, \dots, M \\ \theta_{ji} \,|\, G_j \sim G_j, \quad w_{ji} \,|\, \theta_{ji} &\sim \mathrm{Mult}(\theta_{ji}) \quad j = 1, \dots, M,\, i = 1, \dots, N_j. \end{aligned} \tag{3.2}$$

The distribution G_0 is drawn from a Dirichlet process, and acts as the base distribution for a second Dirichlet process from which the distributions over topics associated with each document are drawn from, thus forming a hierarchical model, hence the name HDP.

The hierarchy is necessary for documents to share topics. The base distribution H is likely to be continuous (e.g., a symmetric Dirichlet). Therefore, if the document-level topic distributions G_j were generated as $G_j|\gamma, H \sim \mathrm{DP}(\gamma, H)$, then documents would share topics with probability zero. However, it is desirable to have topics that are shared between documents, and thus G_0 is used to place a discrete distribution over topics, which serves as a base measure for the document-level topic distribution. This creates a shared, but still unbounded, support for each of the document-specific topic distributions G_j.

Using an equivalent representation based on the stick-breaking process, correspondence between this model and the generative story for documents becomes more clear:

$$\begin{aligned} \theta_k \,|\, H &\sim H \quad k = 1, 2, \dots \\ \boldsymbol{\beta} \,|\, \gamma &\sim \mathrm{Stick}(\gamma) \\ \boldsymbol{\pi}_j \,|\, \alpha &\sim \mathrm{Stick}(\alpha) \quad j = 1, \dots, M \\ c_{jt} \,|\, \boldsymbol{\beta} &\sim \mathrm{Mult}(\boldsymbol{\beta}) \quad j = 1, \dots, M,\, t = 1, 2, \dots \\ z_{ji} \,|\, \boldsymbol{\pi}_j &\sim \mathrm{Mult}(\boldsymbol{\pi}_j) \quad j = 1, \dots, M,\, i = 1, \dots, N_j \\ w_{ji} \,|\, (\theta_k)_{k=1}^{\infty}, (c_{jt})_{t=1}^{\infty}, z_{ji} &\sim \mathrm{Mult}(\theta_{c_{jz_{ji}}}) \quad j = 1, \dots, M,\, i = 1, \dots, N_j. \end{aligned} \tag{3.3}$$

For notational convenience we write $\boldsymbol{w}_j = (w_{ji})_{i=1}^{N_j}$, $\boldsymbol{c}_j = (c_{jt})_{t=1}^{\infty}$, $\boldsymbol{c} = (\boldsymbol{c}_j)_{j=1}^{M}$, $\boldsymbol{z}_j = (z_{ji})_{i=1}^{N_j}$, $\boldsymbol{z} = (\boldsymbol{z}_j)_{j=1}^{M}$, $\boldsymbol{\pi} = (\boldsymbol{\pi}_j)_{j=1}^{M}$, and $\boldsymbol{\theta} = (\theta_k)_{k=1}^{\infty}$. The set $\boldsymbol{\theta}$ represent the set of all topics, and $\boldsymbol{c}_j$ contains indices to the topics present in document j with $\boldsymbol{\pi}_j$ representing the corresponding probabilities. The value z_{ji} represents the index in the sequence $\boldsymbol{c}_j$ that specifies the topic associated with word w_{ji}, and the word is generated by sampling from a multinomial with the parameter $\theta_{c_{jz_{ji}}}$. Since the indicator sets $\boldsymbol{c}_j$ for each document are drawn from a multinomial with a common parameter (i.e., their support is shared), documents will share topics with nonzero probability.

The representation in Equation 3.3 has a number of attractive properties. First, there is an explicit decoupling of the topic proportions $\boldsymbol{\pi}_j$ and the topics themselves $\boldsymbol{\theta}$. Rather than explicitly represent the Dirichlet processes, a system of indices[16] and distributions over indices are maintained, which are much simpler computational objects. Second, when considered independently, all distributions over parameters of interest $(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{c}, \boldsymbol{z})$ have conjugate priors, and thus efficient closed-form posterior updates are possible. However, their joint distribution does not have this property and exact posterior inference in this model remains intractable.

3.1.2.4 Online Variational Bayes

Approaches for approximate inference in complex Bayesian models typically fall into one of two categories: sampling-based or variational. Sampling-based approaches construct a Markov chain whose stationary distribution is the desired posterior, and generate samples by running the chain. These approaches are referred to as Markov chain Monte Carlo (MCMC) methods, and samplers for the HDP have been proposed and perform well in empirical evaluations [96, 107]. However, these methods typically require a pass through the entire dataset at

[16]These indices can be thought of as pointers, z_{ji} points to a value in $\boldsymbol{c}_j$ that points to a topic in $\boldsymbol{\theta}$.

each iteration, and samplers often require hundreds or thousands of iterations to produce a sufficient number of independent samples to give a good approximation of the posterior. Thus, scaling to large datasets poses significant computational challenges.

Variational approaches posit a simplified parametric distribution which is optimized to be close to the true posterior, in terms of Kullback-Leibler divergence [108]. When used with hierarchical Bayesian models, these approaches are referred to as variational Bayes (VB). This naturally induces bias, however VB approaches have shown to be faster and produce results as accurate as MCMC in practice [109, 110].

Online stochastic optimization proceeds by subsampling the data and computing an estimate of the gradient of each of the variational parameters based on the sample. Each parameter is updated (in parallel) by following its gradient according to a decreasing step size, a process referred to as coordinate ascent. Sato [111] analyzed stochastic optimization approaches for VB, and showed that coordinate ascent in models where conjugacy holds corresponds to stochastic natural gradient ascent, an approach known to be particularly efficient [112]. An advantage of online approaches such as stochastic optimization is that they process small batches of data at a time, making them particularly well suited for processing large datasets. In many large datasets, there is enough redundancy in the data that online VB algorithms can learn good models in less time than would be required for MCMC approaches to perform even a single iteration.

This stochastic optimization framework [111] has been applied to LDA [113] and more recently to the HDP [7]. For the HDP model, the representation in Equation 3.3 is used. The latent variables of interest are the stick-breaking proportions $\boldsymbol{\beta'} = (\beta'_k)_{k=1}^{\infty}$ and $\boldsymbol{\pi'} = (\pi'_{jt})_{t=1}^{\infty}$, the topics $\boldsymbol{\theta}$, and the indicator variables $\boldsymbol{c}$ and $\boldsymbol{z}$. Note that we use the stick-breaking proportions (see Equation 3.1), not the weights. As is typically done in variational inference, a mean-field approximation approach is used, where the joint distribution over the parameters is approximated by a product of independent distributions. Thus, the variational distribution approximates the true posterior by:

$$q(\boldsymbol{\beta}', \boldsymbol{\pi}', \boldsymbol{c}, \boldsymbol{z}, \boldsymbol{\theta}) = q(\boldsymbol{\beta}')q(\boldsymbol{\pi}')q(\boldsymbol{c})q(\boldsymbol{z})q(\boldsymbol{\theta}).$$

As mentioned in the previous section, each of the distributions in this factored representation have conjugate priors, and thus can be updated efficiently.

For computational tractability, two truncation parameters are required, K and T. The parameter K controls the number of components in $\boldsymbol{\beta}$, thus restricting the total number of topics. The parameter T controls the number of components in each $\boldsymbol{\pi}_j$, thus restricting the number of topics that can be present in any single document. Note that truncating the number of topics by K is different from running LDA with K topics. In LDA, the algorithm is encouraged to make use of all K topics, whereas in the HDP, the algorithm is encouraged to use only as many topics as necessary, which should be fewer than K, if K is chosen appropriately.

Finally, using standard variational theory [108], the marginal log likelihood of the observed data $\mathcal{D} = (\boldsymbol{w}_j)_{j=1}^{M}$ is bounded below:

$$\log p(\mathcal{D}|\gamma, \boldsymbol{\alpha}, H) \geq \mathbb{E}_q \left(\log p(\mathcal{D}, \boldsymbol{\beta}', \boldsymbol{\pi}', \boldsymbol{c}, \boldsymbol{z}, \boldsymbol{\theta}) \right) + \mathcal{H}(q) \triangleq \mathcal{L}(q),$$

using Jensen's inequality, where $\mathcal{H}(q)$ denotes the entropy for the variational distribution. The full derivation is in [7]. This is the variational objective function, and maximizing $\mathcal{L}(q)$ is equivalent to minimizing the Kullback-Leibler divergence between the variational distribution and the true posterior. The variational objective function is optimized using stochastic gradient optimization. Stochastic gradient optimization is an iterative approach where at each step the data is subsampled to form small batches[17] on which the gradient of the objective function

[17]In practice, these batches often have sizes in the hundreds or thousands, whereas the entire data set may have millions of examples.

with respect to the parameters is estimated. Based on the estimate, the parameters are adjusted in the direction of the gradient with a step size that decreases over time. As noted before, since the variational distribution is fully conjugate, updating the parameters in parallel corresponds to following the natural gradient, which results in an efficient optimization approach.

3.1.2.5 A Promising Future

From a computational perspective, most of the work on speeding up inference in Bayesian models has relied on improvements in sampling techniques, such as Hamiltonian Monte Carlo [114], slice sampling [115], and the recent No-U-Turn sampler [116]. Improvements in variational methods have also been presented, such collapsed variational inference [107]. Now that approaches from stochastic optimization have been brought into the fold, we can expect to see advances from the optimization field incorporated into the Bayesian modeller's toolkit.

Already, we have seen improvements from applying active learning for selecting informative batches at each iteration [117], and control variates have been used to reduce variance in the search process [118]. This is an active area of research, and online stochastic variational optimization approaches are likely to benefit from significant improvements in performance in the future. As the approach proposed later in this section relies on stochastic optimization, improvements in this area can be easily incorporated.

3.2 Subjective Location Detection

As previously noted, most work on wifi localization focuses on the positioning problem, where a location is clearly defined as a point on a map. In this work, we consider a different aspect of localization, where the goal is to answer the question "what label would a user assign to their current location?" Here, the notion of a location becomes less clear, and can vary from user to user. Our work is somewhat similar to that of Haeberlen et al. [80] in that their goal is to detect the office that a person is in, rather than a particular coordinate. However, they simply replace the notion of a point with the notion of a region, and these regions have clearly demarcated boundaries (i.e., walls and doors). They also assume a map that is shared by all users, and a common labelling for all users. As in our work, they use a topological representation of the environment. However, unlike in our work, their topology is known in advance.

In this work, we place no constraints on what constitutes a location other than that for a particular user at a particular point in physical space, the answer to "where are you currently?" will always be the same (i.e., time invariant). The level of spacial granularity is arbitrary, and we should, for example, predict the correct label for a user that calls any position on a university campus "School", a user that labels positions according to the building name, or a user that labels individual rooms within a building.

Existing location-based services (LBSs) rely on location labels in order to facilitate interaction with users. For instance, Google Latitude[18] reports the amount of time spent at "home" or at "work", and Apple's mobile operating system supports location-based reminders, so, for example, you can be reminded to do something when you leave or arrive at a particular location. However, these services determine the location by first performing positioning, and then by searching a database of labels associated with the position. Subjective location labels, such as "home" or "work" are typically determined via straightforward heuristics, based on the time spent at different positions. The obvious advantage of such systems are that they work without requiring user intervention, and are useful for discovering new places, or the name of nearby locations. The main disadvantages are the expensive calibration and maintenance required to do accurate positioning when

[18] `http://www.google.com/latitude`

GPS is unavailable, and the lack of personalization, with the exception of a few simple labels, namely “home” and “work”.

The use case for our work is different. In our system, the user provides labels for the locations that are of interest to them, and we use this as training data to train a classifier that can, at a later time, predict that location label when they return to the same (or nearby) position. Ideally, with only a small number of training examples, the system will be able to provide personalized location detection. Additionally, this allows for labelling of indoor locations, such as particular rooms within a house or office, which would not be available to traditional LBSs. While this requires manual training on the part of the user, it requires no centralized calibration effort, and has the added benefit of working without requiring a user to divulge their private data. This, and other privacy-related issues related to this work are discussed in Section 5.2. We could perform subjective location detection by combining positioning with an approach for learning subjective location labels given a position, however we bypass the positioning step and infer labels directly from wifi measurements, as described in the following section.

3.3 Algorithms

Our system has two main components. The first, presented in Section 3.3.1, is a dimensionality reduction technique for generating fingerprints from raw wifi signals that is robust to access points appearing and disappearing, as described in the previous section. The second, presented in Section 3.3.2, is a semi-supervised learning algorithm for classifying labels that uses unlabelled data to improve the performance of the classifier such that only a small number of labeled examples is sufficient to attain high labelling accuracy.

3.3.1 Streaming HDP Clustering

As described in the previous section, a multinomial distribution fits well wifi measurements. In this section we describe how to adapt the online HDP algorithm [7] for multinomial mixture models to work in a streaming setting. Translating the topic modelling analogy to the wifi signal modelling setting, we consider a wifi measurement to be a document, APs correspond to words, and the RSS corresponds to the number of times a word appears in the document. RSS values are discretized and rescaled to correspond to a reasonable range of counts, such as $1, \ldots, 10$, the range used in our experiments. We use the term *cluster* instead of topic, and thus a wifi measurement is generated by sampling a distribution over clusters ϕ, then N times sampling a cluster θ from ϕ and an AP from θ. As clusters correspond to multinomial distributions from which the observations are drawn and a measurement is composed of a mixture of a clusters, this model can be considered a multinomial mixture model.

One interesting property of the HDP model is that the number of clusters scales logarithmically with the size of the data, due to the prior on the cluster assignment distribution. Unlike agglomerative clustering methods such as DBScan [119] which tend to merge clusters as the number of data grow, the HDP model tends to split them. Agglomerative methods have achieved good performance in clustering wifi data [120], however in our experiments we have observed good performance for a small batches of data, but significantly worse performance as the amount of data increases. Using the HDP model, we obtain more refinement in the clusters corresponding to locations where the user spends more time, which leads to better discrimination between nearby locations.

While the batches considered in online HDP are assumed to consist of samples chosen uniformly randomly from the complete data set, we consider the streaming setting, in which data become available one example at a time, but are not assumed to be i.i.d. samples. Rather, subsequent samples are likely strongly correlated. We lose the theoretical guarantees provided by the theory of stochastic approximation, however the variational objective we are optimizing is itself an approximation of the posterior likelihood, and thus there were no guarantees to begin with. In practice, on the types of data we consider, when compared to online HDP we find no loss in performance.

A batch B is maintained, composed of b examples. New data are placed into a buffer, and once the buffer contains $b' < b$ samples, b' samples in B are chosen at random, discarded, and replaced with the samples in the buffer. The buffer is then emptied, and an iteration of stochastic gradient descent is performed using the batch B. At each iteration, each sample is removed with probability b'/b. The probability that a measurement will remain in the batch (and thus be processed) for n iterations is $(b'/b)^n$, and the expected number of iterations that a measurement will remain in the batch is $(b/b')^n$. In our experiments, we fix $b'/b = 1/8$; i.e., at each iteration we discard 1/8 of the data in B and replace it with new data.

Other than the streaming batch update rule, we use the online HDP as originally presented [7]. To track nonstationary data, we use a larger learning rate than is typically used when the mini-batches are i.i.d. samples, and decrease the learning rate slower than is typically done. The learning rate at time t in online HDP (and also online LDA) is $\rho_t = (\tau_0 + t)^{-\kappa}$, and while the best performance for the online algorithms is reported [7] with $\tau_0 = 64$ and κ in the range $[0.6, 1]$, we have found that a smaller decay rate $\kappa = 0.5$, and a larger value $\tau_0 = 128$ led to better results[19].

The streaming HDP algorithm learns a clustering model from data. This model can then be used to convert a wifi measurement to a vector of mixture components, or cluster probabilities. In practice, a wifi measurement is a sparse vector in a high dimensional (on the order of 100,000, in practice) space, while the cluster

[19]Note that $\kappa = 0.5$ leads to a learning rate that does not satisfy the Robbins-Monroe conditions for convergence [121], however we have found that this is preferable for tracking nonstationary streaming data. One could use $\kappa = 0.5 + \epsilon$ for some small $\epsilon > 0$, if satisfying the Robbins-Monroe conditions is a concern.

weights are a sparse vector in a much lower dimensional (on the order of 100, in practice) space, and we can interpret this process as dimensionality reduction. It is important to emphasize that the output of this algorithm is a set of cluster weights, which are typically spread over multiple clusters. In practice, the number of clusters with nonzero weight is small, but is typically greater than 1. Therefore, we do not get a one-to-one correspondence between clusters and locations, as is typical in systems that use clustering for location data [122, 123]. Instead, we interpret the cluster weights as input features for a classifier that learns to predict labels, as we will explain in the following section.

3.3.2 Graph-Based Location Labelling

For many practical applications, obtaining large amounts of unlabelled data is cheap, but obtaining labeled data is expensive. Traditional supervised learning algorithms learn classifiers only from labeled data. However, there is often structure in the unlabelled data that can be used to improve the performance of a classifier. Semi-supervised learning algorithms [19] produce classifiers trained using both unlabelled and labeled data. For our application, it is easy to collect wifi measurements, but we would like to limit the number of labels a user must provide. Ideally, it would be sufficient that a user label a location only once.

The intuition for our approach is as follows. Due to noise, a stationary device may be associated at time t_1 with cluster i, at time t_2 with cluster j, and at time t_3 with both clusters i and j (recall that our clustering approach returns a mixture of clusters). If the user assigns a label l to their location at time t_1, we associate with cluster i the label l. Then at time t_3, we would predict the correct label, but at time t_2 we would be unable to predict the correct label.

However, we can take advantage of the fact that clusters cooccur in a single measurement (as was the case at t_3). Since clusters that cooccur are clusters that can be present at a particular location, we can propagate labels associated with a cluster to those with which it cooccurs. The more frequently two clusters cooccur, the more certain we can be that a label associated with one cluster should also be associated with the other, and that the cooccurrence wasn't simply due to noise in the clustering process.

Our algorithm builds a graph, where clusters represent vertices, and two clusters are joined by an edge if an only if they cooccur in any measurement. An edge is weighted by the frequency of the cooccurrence of the two clusters it joins. Formally, let $n_{i,j}$ be the number of observations that are assigned nonzero weights to clusters i and j, and let n_i be the number of observations where cluster i has nonzero weight. Each cluster with $n_i > 0$ is represented by a vertex, and two vertices i and j are connected by an edge with weight $n_{i,j}/(n_i n_j)$ if and only if $n_{i,j} > 0$.

For each cluster c, we maintain a distribution $p(c|l)$, the probability of that cluster being present for a location labeled l. Given a set of labeled, clustered measurements, $p(c|l)$ is estimated for the clusters appearing in the labeled examples. However, some clusters may not appear in any labeled examples. The weighted graph is used to compute $p(c|l)$ for the remaining clusters by propagating information about neighbouring distributions through connected components of the graph in an iterative manner. At each iteration, for each cluster for which we have not computed $p(c|l)$, a weighted sum (according to the edge weights) of the distributions of its neighbours is computed and associated with the node. If none of the neighbours have an associated distribution $p(c|l)$, then the cluster is not updated. The procedure repeats until no nodes are updated.

Some nodes will still not have an associated distribution $p(c|l)$, but this is to be expected as there are certainly locations that occur in the data for which we do not have event descriptions. We compute $p(c)$ from the clustering of the entire data set, and counting the frequency with which clusters appear, and $p(l)$ from the event descriptions by counting the number of mentions of each location. Once the distributions $p(c)$, $p(l)$, and $p(c|l)$ have been estimated, the distribution $p(l|c) = p(c|l)p(l)/p(c)$ is computed, which is used to label wifi observations based on the observed clusters.

An additional and important advantage of the graph-based representation is that it can be updated as new labeled examples become available. Classifiers such as random forests [124] and support vector machines [52] must be retrained when new training data is made available, and this requires storing the old training data. Simpler models such as naive Bayes (multinomial or Gaussian) and nearest neighbour are updateable, that is they can incorporate new training data online, as they are based on summary statistics of the inputs. However, as we will show in the next section, they do not perform as well as more powerful classifiers. The graph-based representation uses only counts of the individual cluster occurrences and cooccurrences, and can therefore be updated as new labels are presented, after which the labeled data can be discarded. In an real application, we would expect the user to be able to provide new labels, or provide correct labels in the event the system predicts an incorrect label, and therefore a model that can be updated online is necessary.

3.4 Evaluation

Our approach is evaluated on a data set collected using the Android Data Collection Platform (see Appendix A) between March 10, 2011 and July 6, 2012. Wifi measurements were collected every 30 seconds while the phone was powered on. The data contains 949,631 wifi measurements containing 6,705,627 measured RSS values and 44,288 unique APs. Figure 3.6 shows a histogram of the number of visible access points per measurement; the mean number of visible access points per measurement is 7.06.

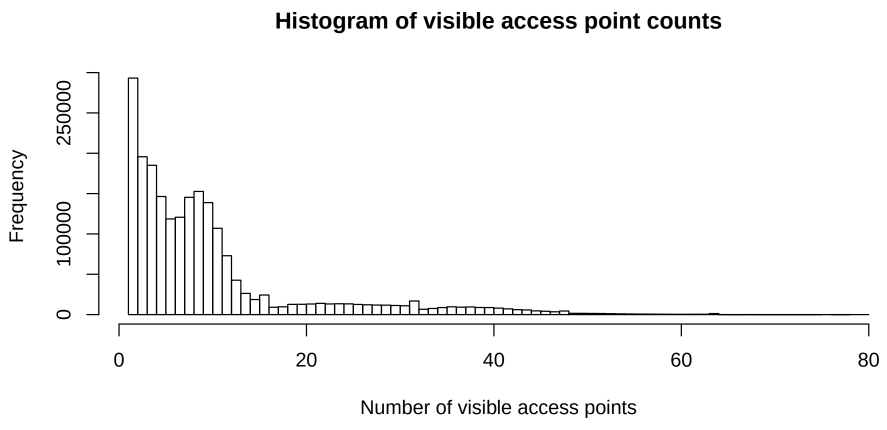


FIGURE 3.6: Number of visible access points per wifi observation.

Between March 14, 2012, and July 10, 2012, locations of interest were labeled by the user. The only restriction on what constituted a location of interest was that the user must spend at least one minute in that location after the label was assigned, to guarantee that a wifi measurement would be available for the location. The same location was not allowed to be labeled twice within a one-hour period. From this process, 900 location labels were collected for 19 unique locations. The location labels were not specified in advance, and were assigned by the user throughout the 4-month collection period as new locations of interest were visited. The location labels are listed in Table 3.1, anonymized as to remove any company names or identifying information.

Note that four of the locations correspond to different rooms in a “home”, and nine of the locations correspond to different rooms and offices within one building, referred to as “work”. The “home” labels are particularly challenging, as all four rooms are within 10m of each other, and in particular the kitchen and bedroom are separated only by a thin wall. There is significant class imbalance, but the classes are roughly proportional to the amount of visits to each location. Therefore, accuracy is an appropriate measure of how often, throughout the course of a regular day, the device will correctly choose the appropriate label for the location the user is in, for locations that the user has indicated are of interest. Each location label

Label	**Number of instances**
Bus Stop A	79
Bus Stop B	59
Home - Bathroom	12
Home - Bedroom	106
Home - Kitchen	63
Home - Living Room	157
Restaurant A	4
Restaurant B	3
Restaurant C	7
Restaurant D	10
Work - Bathroom	24
Work - Cafeteria	17
Work - Classroom	55
Work - Lounge	50
Work - My Office	221
Work - Office 111	11
Work - Prof. Lounge	9
Work - Seminar Room	8
Work - Supervisor's Office	3

Table 3.1: Location labels and the number of times the label was provided.

was associated with the wifi measurement that occurred closest to it in time. Thus, our labeled data consists of 900 wifi measurements, each with an associated location label.

Streaming HDP Parameter Tuning

The first 8 months of data was used for parameter tuning and experimenting with the streaming HDP algorithm. This phase consisted of running the algorithm with different truncation parameters K and T, and learning rate parameters τ and κ (see Section 3.3.1).

Increasing the truncation parameters slows down the algorithm considerably. We first ran the algorithm with large values for K and T, varying the parameters τ and κ and observed that the resulting models used between 250 and 400 clusters, and the number of clusters associated with an observation rarely exceeded 10. To allow some leeway, we settled on parameters $K = 500$ and $T = 20$.

After fixing the truncation parameters, we considered the learning rate parameters. As described in Section 3.3.1, the learning rate at iteration t is $\rho_t = (\tau_0 + t)^{-\kappa}$.

Therefore, κ controls the rate at which the learning rate decays, and τ_0 sets the initial rate, thus having more of an impact on the behaviour of the algorithm in the early iterations. As we are considering the streaming setting, and want to track nonstationary behaviour in the data, we expect that κ should be small, so the learning rate does not decay too quickly, and that τ_0 should be large so that early iterations do not have a far greater impact than later iterations. The effect of changing the parameters was judged qualitatively, by clustering the data and plotting the graph described in Section 3.3.2. As most of the user's time was spent at "work" and at "home", the graphs were judged by whether two large clusters were clearly visible. After examining a range of parameters, it was determined that the clustering is fairly robust to the values, and little differences in the clustering could be observed. The values were chosen to be $\kappa = 0.5$ and $\tau_0 = 128$, noting that values in the ranges $\kappa = [0.5, 0.6]$ and $\tau_0 = 8, 16, 32, 64, 128, 256$ all produced fairly similar results. This is encouraging, as it demonstrates robustness in the streaming HDP algorithm.

Finally, we compared streaming HDP with the original online HDP algorithm [7]. Both algorithms ran with the same batch size $B = 5076$ for the same number of iterations. Qualitatively (in terms of the appearance of the cluster cooccurrence graph), the results were indistinguishable.

Results

The final 8 months of data were used to evaluate the location labelling algorithm. This consisted of 540,970 measurements and 24,486 unique access points. 900 of these measurements from the 4 most recent months were labeled with a location. The streaming HDP algorithm was used to cluster the entire dataset, using the parameters $K = 500$, $T = 20$, $\kappa = 0.5$, and $\tau_0 = 128$. The batch size was chosen arbitrarily to be 5076 which corresponds to 48 hours of data, and at each iteration 1/8 of the data was replaced with new data. The algorithm took approximately 160 minutes to learn the model, and 60 minutes to assign cluster weights to the

Classifier	**Raw Signals**	**Cluster Weights**
Random Forest	87.87%	83.11%
Linear SVM	88.21%	**86.05%**
Naive Bayes (Multinomial)	**88.66%**	81.86%
Naive Bayes (Gaussian)	79.48%	78.00%
1-NN (Euclidean)	85.03%	83.56%

Table 3.2: Classification accuracy on full location dataset with 10-fold cross-validation using raw wifi measurements and cluster weights as input features.

entire dataset on a 2.6 GHz Xeon desktop. The total number of clusters found by the algorithm was 344.

First we considered the classification accuracy on the entire labeled data set using the raw wifi observations and the cluster weights as input features, using 10-fold stratified cross-validation in both cases. We tested four classifiers: random forests [124], linear SVM [125], Gaussian naive Bayes, multinomial naive Bayes, and a 1-NN using Euclidean distance. The results are in Table 3.2.

From the results in Table 3.2, we see that for raw signals, the multinomial naive Bayes algorithm outperforms much more complex classifiers, thus further reinforcing our claims in Section 3.1.1.3 that a multinomial distribution is an appropriate distribution for modelling wifi measurements. We also observe that classifying based on the much lower-dimensional cluster weight representation only leads to a slight decrease in performance. Note that the raw signal vector has 24,486 components, while the cluster weight vector has 344. However, both vectors are sparse, and so the effective reduction in storage requirements is not substantial. In this labeled data set, on average cluster weight vectors had approximately 12 nonzero entries, while raw wifi measurements had approximately 15 nonzero entries.

Note that the results in Table 3.2 are computed using cross-validation, and therefore the training set on each of the 10 iterations contains 90% of the labeled data. Ideally, we would like the user to provide at most one or two labels for each location. We trained a the same set of classifiers as above, using only one randomly selected labeled training example per class, repeating the experiment 100 times

Classifier	**Accuracy (%)**
Naive Bayes (multinomial), Raw Signals	$60.31 \pm 8.60\%$
Naive Bayes (multinomial), Cluster Weights	$60.51 \pm 9.03\%$
Graph-based Algorithm	$\mathbf{69.55 \pm 8.22}\%$

Table 3.3: Classification accuracy using one labeled training example. Results show mean and one standard deviation from 100 trials.

and averaging the results. For both the cluster weights and raw vector representations, the multinomial naive Bayes classifier outperformed the other classifiers, and we only report these results.

Table 3.3 shows the results mean classification accuracy and standard deviation when only one training example per class is available, averaged over 100 runs. We see that the graph-based classifier outperforms the other classifiers, though the variance for all three classifiers is large. As the graph-based algorithm maintains only counts of cluster observations, it can be updated online using both labeled and unlabelled data. Nodes in the graph are annotated with labels, and thus labeled examples can be discarded after the graph is updated. As we will see in the following chapter, the graph also serves as a particularly useful visualisation tool.

Table 3.4 shows a confusion matrix for the graph-based algorithm when trained with one labeled example per class. We see that nearly every error corresponds to a mislabelling between rooms that are within 10m of each other, many, such as the "Seminar Room" and the "Lounge", are separated only by a wall. We also note that many of the errors correspond to mislabelling of locations that are visited infrequently, and thus are unlikely to have been assigned unique clusters. An example is the location labeled "Supervisor's Office", which is visited only three times, and get labeled as the more frequently visited "Prof Lounge", which is happens to be adjacent to. Another example is "Restaurant C", which is labeled as "Bus Stop B", which is visited much more frequently, and is 15m away.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	← classified as
57														1					a = Bus Stop A
	78																		b = Bus Stop B
			7		4														c = Home - Bathroom
			78	7	20														d = Home - Bedroom
			45	3	14														e = Home - Kitchen
			30	5	117														f = Home - Living Room
						3													g = Restaurant A
							2												h = Restaurant B
	6																		i = Restaurant C
									9										j = Restaurant D
										8	4				10		1		k = Work - Bathroom
											16								l = Work - Cafeteria
												48	2						m = Work - Classroom
											1		41				6		n = Work - Lounge
											1	9	3	182	4	2	10		o = Work - My Office
										3				1	6				p = Work - Office 111
																8			q = Work - Prof. Lounge
													7						r = Work - Seminar Room
																2			s = Work - Supervisor's Office

Table 3.4: Confusion matrix for graph-based classifier with one training example per class.

3.5 Conclusion

This chapter describes a novel system for wifi location detection that works well indoors and outdoors. We present an approach that avoids the positioning step typical in most localization systems, and learns to label locations directly from wifi measurements. In the course of this work, we have analyzed approximately 3.4 million wifi measurements from over 40 individuals (this includes the data considered in the following chapter).

The central idea is modelling wifi observations using a multinomial distribution, and we demonstrate that this model is more appropriate than the Gaussian models that are typically used. We present a modification to the online HDP algorithm that learns a nonparametric multinomial mixture model from 8 months of data in less than four hours on a standard desktop computer. A semi-supervised graph-based location classifier is presented that outperforms existing approaches when trained on a large set of unlabelled data and one labeled example per location, and is able to accurately predict labels for nearby indoor locations.

This work differs from typical localization approaches in that we consider the problem of predicting the label that a user would assign to their current position, rather than placing a user on a physical map. We focus on personalized location detection and build classifiers unique to individuals. Thus, we avoid the costly global calibration step required by most localization systems, but forfeit the ability to identify locations the user has not previously labeled. However, there is nothing inherent in our approach that prevents the cluster model and classifiers to be shared between users.

Chapter 4

Generating Storylines from Data

Between January and April, 2012, we participated in an event organized by Nokia© called the Nokia© Mobile Data Challenge (MDC) [126]. Nokia engaged in a significant data collection campaign between between October 2009 and March 2011, collecting data from 200 subjects. People in the city of Lake Geneva region of Switzerland were offered a free Nokia mobile phone in exchange for their participation in the data collection process. These subjects were recruited using viral marketing, and thus many of the participants had existing social connections with each other. The mobile phones contained logging software that collected data from the GPS, GSM, and Bluetooth radios, the accelerometer sensors, as well as usage data consisting of incoming and outgoing calls and text messages, camera and media player usage, as well as application usage.

In an effort to involve the general research community, Nokia organized the MDC, wherein researchers could submit proposals for a set of dedicated challenges, which consisted of a set of competitive tasks, and the open challenge, in which researchers could propose their own tasks. We participated in the open challenge; the title of our proposal was "Generating Storylines from Sensor Data". The goal of our work was to provide intelligibility by producing from the raw sensor data a set of English-language descriptions of the important events captured by the data. In this chapter, we review the natural language processing tools that we use, we describe the MDC dataset and some visualization tools that were built, and we report the results of our work.

4.1 Background

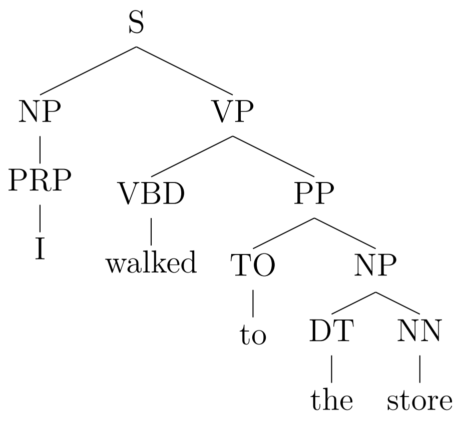
The work in this chapter builds on a number of techniques from the field of natural language processing (NLP). Here we describe the problems of parsing, named entity recognition (NER), and statistical machine translation (SMT). This is followed by a discussion of common themes in the approaches to these three tasks.

4.1.1 Parsing

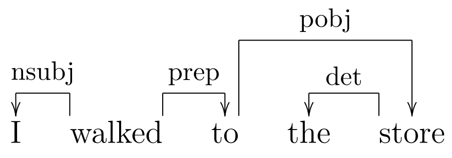
Parsing is the process of forming a structured representation from a string of text. The most typical structure is that of a hierarchy of grammatical relations, referred to as a *parse tree*. Two types of parse trees are commonly used in natural language processing applications: *constituency-based* and *dependency-based*.

Constituency-based parse trees represent the syntactic structure of a sentence where interior nodes correspond to phrases and terminal nodes (i.e., leaves) correspond to words. Such trees are built by recursively partitioning a sentence into its constituent phrases, until each phrase consists only of a single word. Figure 4.1a shows an example of a constituency-based parse tree for the sentence “I walked to the store”, generated by the Stanford Parser [2]. The tags used for the interior nodes are from the Penn Treebank tagset [127], the de facto labels for part-of-speech tagging.

In the example in Figure 4.1a, the root of the tree is the sentence, which is made up of a noun phrase and a verb phrase. The noun phrase is made up of a preposition which corresponds to the word “I”. The verb phrase is further decomposed into the past-tense verb “walked” and a prepositional phrase. The prepositional phrase is



(A) Constituency-based parse tree.



(B) Dependency-based parse tree.

FIGURE 4.1: Two parse trees for the sentence "I walked to the store", generated by the Stanford Parser [2].

made up of the word "to" and a noun phrase, which is made up of the determiner "the" and the singular noun "store".

A dependency-based parse tree depicts grammatical relationships between the words. All nodes are viewed as terminal, with labeled edges corresponding to the relationships, and they are commonly depicted as in Figure 4.1b. Figure 4.1b shows the dependency-based parse tree for the same sentence, "I walked to the store", also generated by the Stanford Parser. The root of the tree is the word "walked", which can be seen as the functional core of the sentence, which is about walking. The subject of the sentence is a noun, "I", which states who is walking, and object is contained in the preposition rooted at the word "to". The proper object in the sentence is "store", and which particular store is indicated by the determiner "the". We see that reading the dependency tree in this manner gives a clear picture of the roles of the words in the sentence, and for simple sentence the verb, object, and subject can be clearly identified.

Both constituency and dependency-based parse trees are formed from an input string and a *grammar*. Unsurprisingly, constituency-based trees are generated by *constituency grammars* (sometimes referred to as *phrase structure grammars*), and dependency-based trees are generated by *dependency grammars*. Constructing grammars is a large and active area of research. Grammars can be constructed by hand, based on knowledge of the rules of grammar for a language, or from data.

The Stanford Parser, which we use in this chapter, builds probabilistic context-free grammars from large annotated text corpora. Probabilistic context-free grammars are a particularly attractive form of representation, as they enable efficient parsing and can be learned using straightforward maximum likelihood approaches [2]. The Stanford Parser is distributed with a constituency grammar learned on a large annotated text corpora [2] and a hand-crafted dependency grammar [128].

4.1.2 Named Entity Recognition

The Stanford Named Entity Recognizer is a toolkit for NER, and can recognize the names of people, organizations, locations, dates, and numeric entities such as times, money, and percentages. Their approach is based on linear chain conditional random fields [129], a state-of-the-art probabilistic sequence labelling and segmentation algorithm. These models are trained using approximate inference (i.e., Gibbs sampling) in order to scale to large datasets [130]. Fortunately for the end user, models trained on large annotated corpora are distributed with the software.

4.1.3 Statistical Machine Translation

The goal of SMT is to build statistical translation models whose parameters are learned on large bilingual text corpora [131]. To translate the string f in the *source language* to a string e in the *target language*, a probability model $p(e|f)$ is learned, and then the translation:

$$\tilde{e} = \operatorname*{argmax}_{e \in E} p(e|f), \tag{4.1}$$

is computed, where E represents the set of all strings in the target language. By Bayes theorem, $p(e|f) \propto p(f|e)p(e)$, so learning the translation model is generally decomposed into two tasks: learning the *translation model* $p(f|e)$ and learning the *language model* $p(e)$.

The translation model $p(f|e)$ is typically learned from a *parallel corpus*, which consists of a pairs of documents in the source language, and their corresponding translated versions in the target language. There is a one-to-one correspondence between the documents in the two languages, but often there is not a direct one-to-one mapping between sentences in the documents. It is common in translated text for a sentence to be split into multiple sentences, or for multiple sentences to be merged into a single sentence. The Gale-Church algorithm [132] is a common approach to aligning sentences in parallel corpora, and relies on the insight that even when split or merged, corresponding sentences tend to have similar length. Thus, a scoring function is posited based on the sentence lengths, and a dynamic programming algorithm is used to find the optimal alignment.

Once sentences have been aligned, the translation model $p(f|e)$ can be learned by counting cooccurrences of words or ngrams in pairs of aligned sentences. Similarly, the language model $p(e)$ is typically estimated by word or ngram counts, but as it only requires sentences in the target language, training data is much easier to come by, and it is often computed on a much larger corpora.

Given translation and language models, finding the most likely translation in Equation 4.1 remains a challenge, as the set E is generally large. Exhaustively enumerating the possible translations is not feasible, and the purpose of a *decoder* is to implement heuristics to efficiently find the most likely translation. Examples of decoders include those that use search techniques such as A* (referred to as stack-based decoding in the SMT literature) [133], greedy methods [134], and the approach of Knight [135] which expresses the optimal translation as the solution to a travelling salesman problem and approximates the solution using integer programming.

4.1.4 Summary

A common theme in NLP is the shift towards statistical models. In order to support more complex language models, such as higher-order ngram models, more data are required. This has been facilitated by recent widespread availability of digitized text, largely due to the Internet.

Increased availability to multilingual text has been a boon for statistical machine translation. Sites such as Wikipedia[1] contain articles in over 250 languages, and for many languages there exist hundreds of thousands or millions of articles. The formation of the European Union has resulted in large volumes of multilingual corpora, as all EU proceedings are translated into its 23 official constituent languages and made public. In particular, a sentence-aligned corpus based on the translation of the equivalent of 53 million English words to each of 11 popular languages, called the EUROPARL corpus [136], is distributed in a format particularly suited for SMT systems[2]. Google's translation system[3] is considered to represent the state of the art, and is reportedly trained on a large proprietary multilingual corpora retrieved from the United Nations and Eurpean Union proceedings[4]. Models for parsing, which only require annotated text in a single language, are generally trained on text extracted from news organizations such as the Wall Street Journal and Reuters and annotated by hand[5].

[1]`http://www.wikipedia.org`

[2]Available online at: `http://www.statmt.org/europarl/`.

[3]`http://translate.google.com/`.

[4]Based on information from Reuters: `http://www.reuters.com/article/2007/03/28/us-google-translate-idUSN1921881520070328` (Retrieved July 30, 2012).

[5]Many such corpora are available from the Linguistic Data Consortium (`http://www.ldc.upenn.edu/`).

One of the caveats of data-driven systems is the problem of domain adaptation. Much of the available multilingual text comes from parliamentary proceedings, and thus machine translation systems perform well on that particular domain. However, it has been noted that generalization to other types of text is difficult, and often results reported on the same domain do not well represent performance on other domains [131]. With the digitization of books, many of which have been professionally translated, comes another source of multilingual text data [137]. It is likely that translation performance will improve as more diverse multilingual corpora become available.

4.2 The MDC Dataset

The dataset provided to open challenge participants consisted of data from 38 participants. The following are some statistics from the data:

- Total # Records: 32,095,200 (840k / user)
- Total # Outgoing Calls: 28,655
- Total # Incoming Calls: 18,373
- Total # Outgoing Texts: 14,565
- Total # Incoming Texts: 20,099
- Total # Wifi Measurements: 8,087,843
- Total # GPS Readings: 1,553,192
- Total # Bluetooth Observations: 4,509,322
- Total # GSM Measurements: 8,029,426

We used the MongoDB database system[6] to store and index the data, requiring a total of 148GB. MongoDB was chosen as it allows for distributing the data across multiple machines (i.e., sharding), and supports map-reduce queries [6] for efficient

[6]`http://www.mongodb.org/`

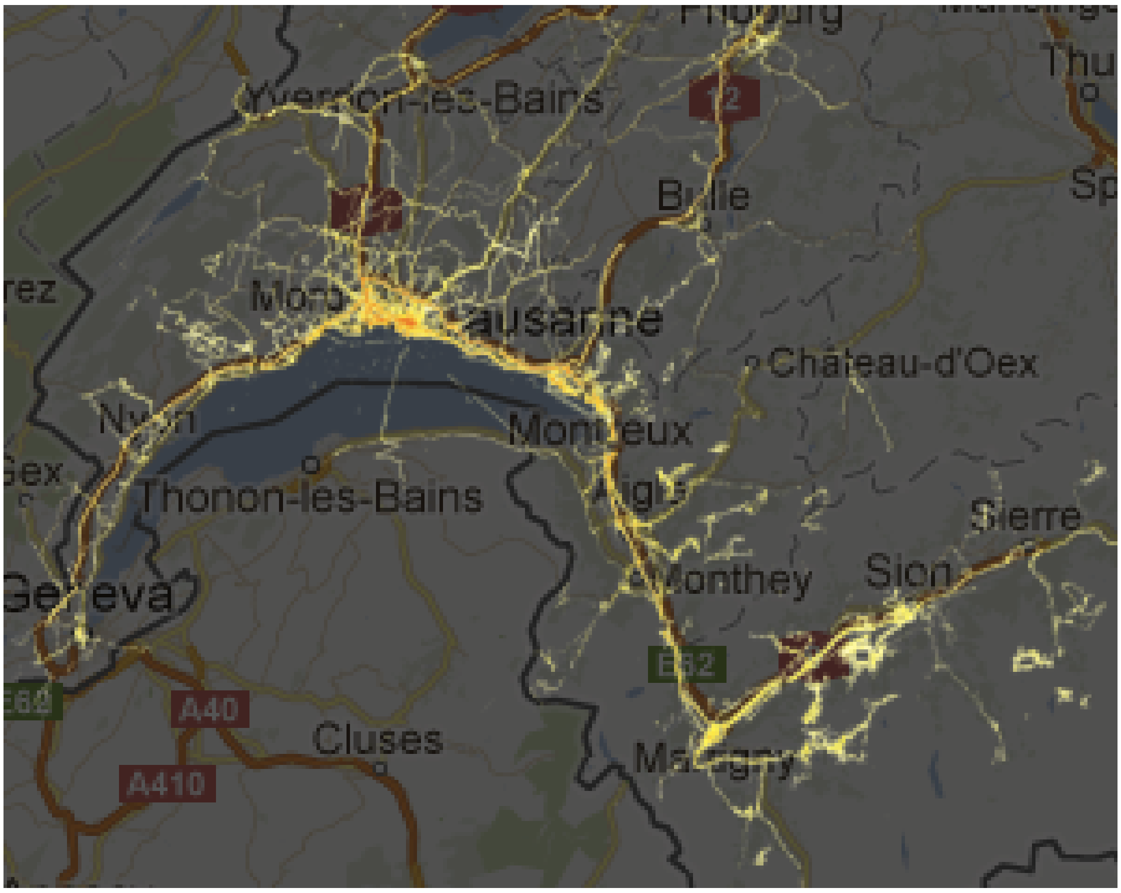


Figure 4.2: A portion of the Lake Geneva region overlaid with a location-entropy heatmap computed from GPS data. Red regions indicate high entropy, white regions indicate low entropy.

parallel data processing of the data. The data was stored on four machines, each with 8 GB RAM and a 2.4 GHz processor.

To get familiar with the location data provided by the GPS readings, we computed the location-entropy. First, we divided the physical region covered by the GPS data into a 4096 by 4096 grid. For each grid cell in which at least one subject was present, we count the number of times each subject was recorded in the cell, and then use these counts to estimate a probability distribution for the cell. For each subject i and each cell c, let n_{ci} be the number of times subject i is recorded in c. We computed the probability $p_c(i) = n_{ci} / \sum_u n_{cu}$, and then assigned to grid cell c the entropy of p_c. Using the map-reduce interface, the location-entropy was computed from over 1.5 million GPS readings in approximately one minute.

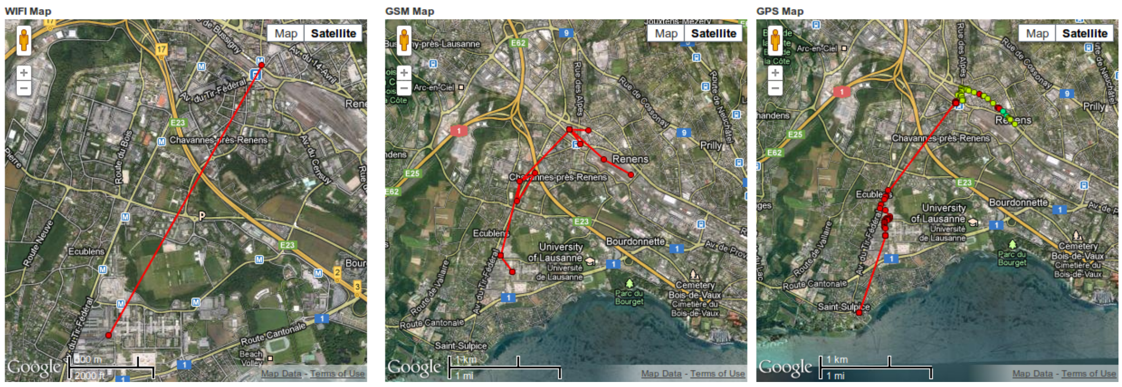


Figure 4.3: Maps generated from Wifi, GSM, and GPS data, from left-to-right respectively. Red and green points indicate position estimates for the user, with subsequent points connected by lines. Green points indicate positions where the user is moving quickly.

Figure 4.2 shows the location-entropy map overlaid on the map of the Lake Geneva region. High entropy regions represent areas where many people go, such as the Lausanne city centre, and are coloured red. Whiter areas indicate lower entropy regions where only a few people spend time, such as residential areas. We also computed hourly location-entropy maps, allowing us to visualize activity patterns at different times throughout the day. As expected, subjects tended to congregate in the downtown region during the day, and return to suburban regions outside the city centre at night.

We built a web application for browsing and visualizing the entire MDC dataset. Similar visualization tools have been developed for other datasets, most notably the Reality Mining dataset from MIT [138] The application allowed a user to browse the data by subject, and for each subject by day. Maps were generated from GPS, Wifi, and GSM data. For GSM measurements, data from the OpenCellID project[7] was used to plot the approximate position of the cellular tower to which the phone was actively communicating. For Wifi data, data provided by Nokia was merged with data retrieved from the Google Maps API[8] and a weighted average (weighted by RSS measurements) of nearby access points was used to estimate the position.

[7]`http://www.opencellid.org/`
[8]`https://developers.google.com/maps/`

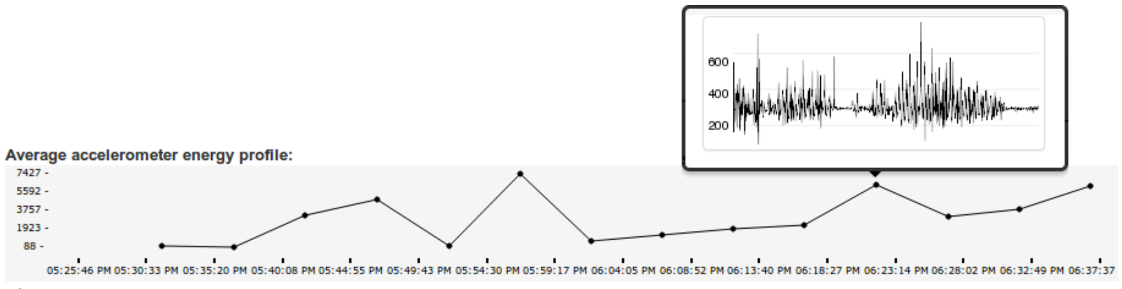


FIGURE 4.4: Accelerometer data for a particular subject. The vertical axis represents the energy in each sample, and the details of the sample can be viewed by hovering the mouse over the sample, as shown.

An example of the three maps is shown in Figure 4.3. We see that the points on the map roughly correspond to each other. As the GSM map is generated from the closest cellular tower (roughly speaking), the points jump around more. The GPS data is the most accurate, when available, but we also see that the southern-most point has jumped far from the true location. It turns out that this is an artefact caused by the anonymisation of the data by Nokia, wherein the precise location where the user works has been obscured by removing the least significant digits of the latitude and longitude data.

From Figure 4.3, we see that wifi positioning is only sparsely available when the user is in transit, however when it is available it is roughly as accurate as the GPS data. The two locations visited during the time period are the metro station and the final destination at the university, both of which are visible on the wifi map.

The web application displays a list of incoming and outgoing calls and text messages, and displays application usage events, such as checking the calendar or listening to a song. Events such as plugging in and unplugging the phone are also useful, as they generally indicate the a user is remaining in a particular location. Data from the accelerometer is visualized by a plot depicting the average energy in a segment, and hovering the mouse over points on the plot causes a window plotting the corresponding data to appear, as in Figure 4.4.

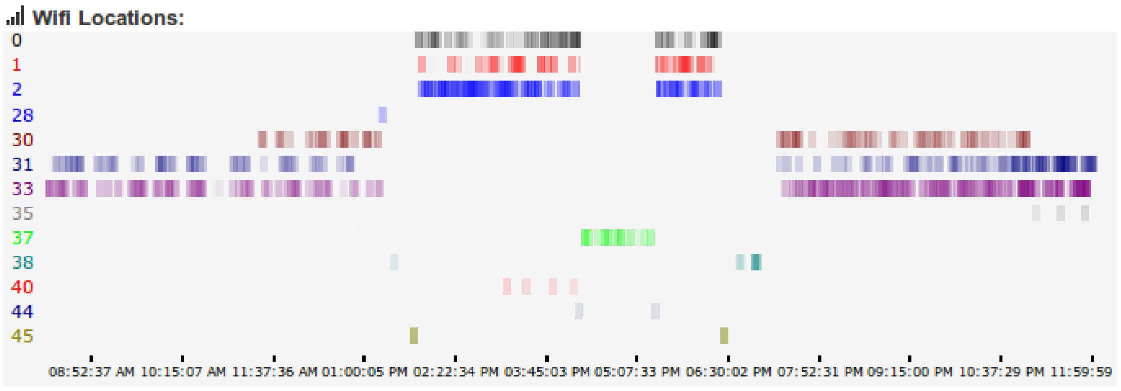


FIGURE 4.5: Wifi cluster indicators depicting the locations visited by the subject on a particular day. Each row corresponds to a cluster, and the opacity of the coloured bars indicate the cluster weights at a particular time. This image has been edited for display, and some rows have been removed for clarity.

We clustered all of the wifi data, independently for each user, using the same approach and parameters as in Section 3.4. An example of how this data is displayed to the user is in Figure 4.5. After spending time looking at the data for this particular subject, one can infer the story told by this data. Clusters 31 and 33 tend to have high weight whenever the user is at home. Cluster 30 likely corresponds to a particular room in the subject's home that is not the bedroom, as it tends to not be present in the middle of the night. Cluster 28 corresponds to the bus stop near the subject's home, and clusters 38 and 45 correspond to the two metro stations visited during the daily commute. Clusters 0, 1, and 2 correspond to the subject's office, and cluster 37 likely corresponds to a lecture hall (the subject works or studies at EPFL), as it is visited at regular times and the subject spends roughly an hour or so in that location.

What is immediately clear is how detailed a picture of a subject's behaviour can be inferred from the data, with only a small amount of effort. Combining the maps with the plots of the wifi clusters, one can quickly identify correspondences between clusters and physical locations, after which one can effectively "read" the story of a subject's day from just the wifi cluster plots. In Section 4.4 we describe how we used this visualization tool to produce detailed stories for the users.

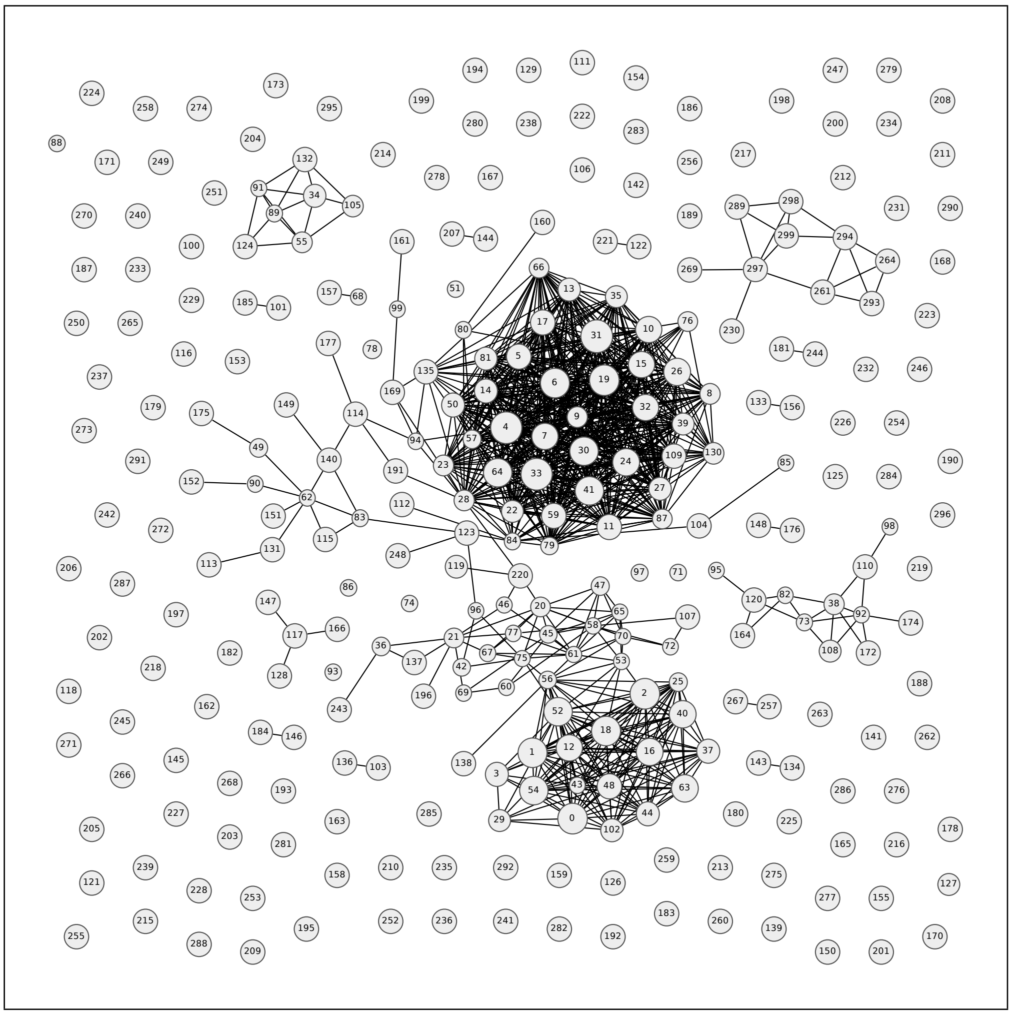


Figure 4.6: Wifi clusters for an MDC participant. The largest cluster represents the subject's home, the second-largest represents the subject's workplace.

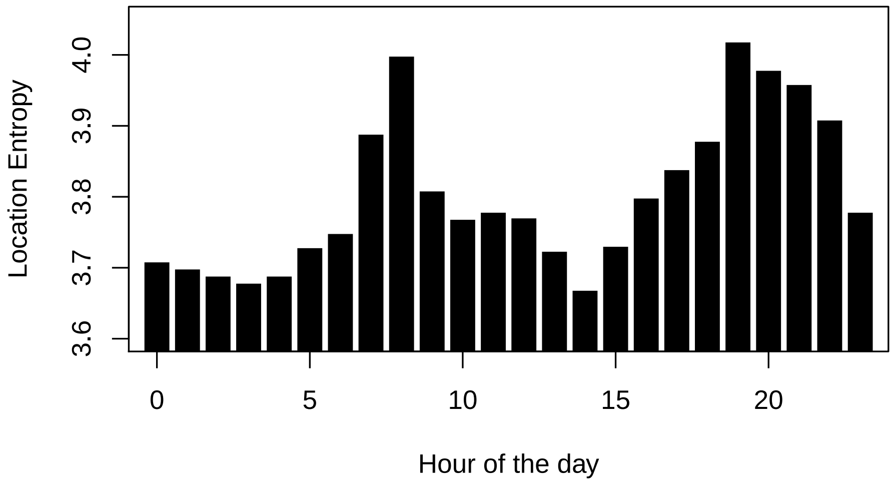


Figure 4.7: Entropy of location-cluster probabilities for each hour of the day for a particular subject.

Finally, we used the graph-based algorithm from Section 3.3.2 to visualize the wifi cluster data. This algorithm generates a graph of the wifi cluster data by representing each cluster with a node, and connecting two nodes if the corresponding cluster weights cooccurred (i.e., had nonzero weight) in a single wifi measurement. Figure 4.6 depicts the wifi cluster graph for a randomly selected subject, laid out using the Kamada-Kawai force-directed layout algorithm [60].

Using the wifi clusters and the graph depicted in Figure 4.6, we can generate animations of the subject's movements throughout the day, as well as aggregate data from multiple days and look for patterns. Figure 4.7 depicts the entropy of the location distribution (the probability that the user is in a particular location) for each hour of the day for a randomly selected subject. As expected, the entropy is low at night and in the middle of the day, when the user is predictably at home or at work, respectively, and is greatest between 08:00 and 09:00 when the user is commuting to work, and in the evenings.

4.3 Generating Storylines

In this section, we present the main contribution of this chapter: a framework for translating between mobile sensor data and English-language descriptions of events. In contrast to the typical SMT setting, where a pair of parallel corpora are used as training data, the source and target languages differ greatly in our setting. For training the translator, we use as input English-language descriptions of events, such as "I left home and went to my office, arriving at 5:00PM". The output consists of raw sensor data, for example, a GPS sensor reading such as:

$$\{\text{time: } 1272222153, \text{lat: } 46.527, \text{lon: } 6.5831, \ldots\}.$$

Alignment is particularly challenging as the main assumption of the Gale-Church algorithm [132], that sentences are of roughly equal length, is clearly violated.

Additionally, the sensor data consist of a large number of values that can be ignored. For instance, the battery level may be reported every minute, but in the description of a user's day, one would not want mentioned every battery level reading. As our goal is to present a concise summary of the events that occurred, the first step is to preprocess the data in order to find events of interest.

The events of interest depend greatly on the data modality. For instance, for call log data, every event is likely of interest, while for data pertaining to location, one might only be interested in when the user moves from one location to another. In the following section, we discuss how we extract location-based events of interest. For many other modalities, such as application usage events, call and text messaging events, and status event (e.g., ringer disabled, phone plugged in, etc.), it is assumed that all entries in the sensor data are of interest, although performing some additional preprocessing would likely be valuable. To ease our notation, we will refer to the English-language description of events as *event descriptions*, and the set of events of interest as the *event data*.

Four assumptions are made about the event descriptions on which the translator is trained. First, it is assumed that every sentence contains at least one reference to the time at which the event being described occurs. Second, a sentence can be taken as a stand-alone description of at least one event, and does not reference from other sentences (e.g. no pronouns refer to nouns in other sentences). Third, it is assumed that every sentence has some sensor data associated with it, that is there are no sentences describing events that could not be deduced from the available sensor data. By restricting ourselves to a simpler subset of English, good results can be achieved with few event descriptions. As training data is expensive to collect, and requires accurate recollections of events, we believe that this tradeoff is justified. Finally, it is assumed that there is a large amount of event data, but a much smaller number of event descriptions.

We begin by considering each sentence to be an event description. An annotated parse tree of each sentence is generated using the Stanford English PCFG parser (Version 2.0.1) [2] and Stanford Named Entity Recognizer (Version 1.2.4) [130]. Through the annotations, the number of time references in each sentence can be determined. If there is more than one reference to a time, for example in the sentence *"I left my office at 12:07PM and walked to the cafe, arriving at 12:11PM."*, a simple splitting rule is used to split the sentence by finding the deepest common ancestor x in the parse tree of the words referencing the times, typically a conjunction, and returning two trees, each with one of the children of x removed. In our example sentence, we would get two sentences "*I left my office at 12:07PM.*", and "*I walked to the cafe, arriving at 12:11PM.*" A final tree transformation is applied to automatically split sentences such as "*I emailed Jane and left my office at 12:30PM*", into two sentences "*I emailed Jane at 12:30PM*" and "*I left my office at 12:30PM*". This is done automatically using the tree transformation[9]:

```
(S NP (VP VP1 CC VP2) .) → (S NP VP1 .),(S NP VP2 .).
```

Next, features are extracted from each sentence using a set of heuristics that were tailored to this particular domain and the limited subset of the English language being considered. The verb at the root of the dependency tree for each sentence is extracted, as it generally represents the *event action* The dependent subject and object are extracted, as they may, for example, correspond to the caller and callee, or text message sender and recipient. Location-related actions, indicated by the verbs *leave*, *drive*, *walk*, *take*, *go*, *return*, and *arrive*, are treated as a single *location* action. For sentences describing location-related actions the parse tree is searched for a noun phrase with a preposition parent, as such a phrase often contains the label for the location to which the action applies. Named entities that represent times are extracted, indicating the time at which the action occurred, and in some cases the duration of the action being described (e.g., the length of a call).

[9]See [131, chapter 11] for an explanation of this syntax

Let A denote the set of all event actions observed in the training sentences. Each event description is represented as a tuple (t, a, ϕ), where t is the timestamp, $a \in A$ denotes the action, and ϕ denotes the features (whose domain varies depending on the value of a, as described previously). Each sensor datum is represented as a tuple (t, e, θ), where t is the timestamp, e is the event type, and θ denote the features (whose domain depends on e). The event types and corresponding features are described in [126].

As previously discussed, one of the main challenges is aligning event descriptions with the appropriate event data. It is assumed that the event times given in the event descriptions are accurate to within one minute. For each event description, a window of the event data in the range of one minute prior to and one minute after the event time is retrieved. Each event datum has an associated type (e.g., `gps`, `calllog`, `application`, etc.). From this data the maximum likelihood distribution, $p(e|a)$, for event types conditioned on the event action is computed by counting the number of times each event type e is associated with event action a and dividing by the number of times action a occurs.

Next, for each event type that appears in the window, a distribution $p(\theta|e, a, \phi)$ is computed. The most general form for this latter distribution can be computed by taking the cross product of the feature values θ and ϕ and computing conditional distributions $p(\theta_i|\phi_j)$ for each pair $(i, j) \in \{1, \ldots, |\theta|\} \times \{1, \ldots, |\phi|\}$.

However, the large number of parameters would necessitate a large amount of training data. It is reasonable to assume that some knowledge of the semantics of the event data features ϕ is available, and this knowledge can be used to construct distributions for each event type by comparing only the appropriate pairs of features. For instance, for call log data, where

$$\theta = (\mathit{length}, \mathit{description}, \mathit{direction}, \mathit{number}),$$

and

$$\phi = (\textit{object}, \textit{subject}, \textit{duration}),$$

we have:

$$\begin{aligned} &p(\theta|\phi, e = \texttt{calllog}, a = \texttt{call}) = \\ &\quad \delta(\textit{direction} = \texttt{incoming})\delta(\textit{object} = \texttt{me})\times \\ &\quad \delta(|\textit{length} - \textit{duration}| < 60)p(\textit{number}|\textit{name} = \texttt{subject})+ \\ &\quad \delta(\textit{direction} = \texttt{outgoing})\delta(\textit{subject} = \texttt{me})\times \\ &\quad \delta(|\textit{length} - \textit{duration}| < 60)p(\textit{number}|\textit{name} = \texttt{object}), \end{aligned}$$

where δ is the Kronecker delta function and $p(\textit{number}|\textit{name})$ can be estimated from the training data and potentially augmented with information from the user's address book.

Due to the time constraints set forth by the MDC, we did not develop a complete translation system, but focused on location-based events, as detected using the wifi location detection algorithm from Chapter 3. Translating from location data to human readable descriptions is called *reverse geociting* [139]. As described in the previous section, all wifi data was clustered, so we have:

$$\theta = (\textit{cluster weights}), \phi = (\textit{location label}),$$

and $p(\theta|e = \texttt{wifi}, a = \texttt{location}, \phi)$ corresponds to $p(c|l)$ in the graph-based algorithm from Section 3.3.2.

For a sequence of event data, most likely locations for each wifi observation are computed and then simple smoothing is performed to eliminate oscillations by replacing occurrences of the sequence $l_1l_2l_1$ with $l_1l_1l_1$. Events of interest are represented as the points in time when the location changes. Arrival events are generated when a pattern $l_2l_1l_1$ is observed, and leaving events are generated when a pattern $l_1 l_1 l_2$ is observed. Event descriptions corresponding to arrival and leaving events were generated using simple templates. Despite the simplicity of these rules, the performance is quite good, as is demonstrated in the following section.

4.4 Results and Discussion

The MDC dataset did not contain event descriptions with which to train the translation model. Using the visualization tool described in Section 4.2, we fantasized two weeks worth of journal entries for one particular user (chosen randomly). We invented names for people that the user communicated with, as the data was did not include this information, and invented place names based on visual inspection of the wifi location clusters, and by picking the names of nearby establishments from Google maps when GPS data was available. We were very specific, and included the names of bus and metro lines that the user used to get to places, which we could deduce from the GPS data and the Lausanne public transit website[10]. This process required approximately 40 hours of work. A short segment of a journal entry is included:

I left Home at 7:32AM. I walked to Perrelet Bus Station, arriving at 7:35AM. I checked my Calendar at 7:36AM. I checked my messages at 7:37AM. I took the 7 Bus to Renens-Gare Nord Bus Station, arriving at 7:47AM. I walked to Renens-CFF Metro Station, arriving at 7:50AM. I took the M1 to EPFL Metro Station, arriving at 8:01AM. I walked to My Office, arriving at 8:06AM. I missed a call from Gary at 8:53AM. I plugged my phone in at 10:12AM. I checked my calendar at 10:15AM. I unplugged my phone at 1:01PM.

We clustered the wifi locations using the same parameters as were used for the dataset in Section 3.4 and learned a translation model from the synthesized journal

[10]`http://www.t-l.ch/`

entries. The translation model was used to translate the location data for the week following the last journal entry. A short segment is included:

I left home at 04:20. I arrived at auditorium at 11:48. I left auditorium at 11:50. I arrived at lounge at 11:52. I left lounge at 12:38. I arrived at my office at 12:46. I left my office at 19:51. I arrived at library at 19:53. I left library at 19:57. I arrived at epfl metro station at 20:03. I left epfl metro station at 20:06. I arrived at renens-gare nord bus station at 20:19. I left renens-gare nord bus station at 20:25. I arrived at perrelet bus station at 20:29. I left perrelet bus station at 20:33. I arrived at home at 20:34.

From visual inspection, the location events detected by our algorithm corresponded perfectly with the events that we observed in the wifi data, and the location labels were all correct. Of the 62 location events detected by the algorithm, 6 locations were marked as unknown, and visual inspection of the data confirmed that these constituted clusters that were not visited and did not co-occur with clusters that were visited during the period covered by the training data. However, 4 of these locations were from geographic regions that the user did visit, and so it is possible that these are errors due to deficiencies in the clustering algorithm. Of course, given the nature of this work, and the fact that we generated the training data, a proper quantitative evaluation is impossible.

4.5 Related Work

Most similar to this work is that on grounded language modelling [140]. Grounding language in events looks at the problem of finding correlations between time series data and human-language transcriptions for the purpose of creating machines that can learn the meaning of words in terms of their world view. This technology was applied to learning correspondences between features extracted from video of baseball games and transcriptions of the play-by-play commentary in order to learn the meaning of events for the purpose of searching large corpora using natural language. Given that most television video is accompanied by closed caption transcriptions, large amounts of training data are available, and the performance of the system is impressive. Much of the work on grounded language modelling, namely the hierarchical temporal patterns and intermediate concept layer would likely be useful tools in moving from low level events, as we consider in this chapter, to higher level descriptions of behaviour. This presents an exciting avenue for future work.

4.6 Conclusion

In this chapter we proposed a framework for generating storylines from sensor data using human-generated journal entries as training data. This line of research has strong implications for personal privacy, a common theme in the work in this thesis, and one we discuss further in Chapter 5.

As is common for practical applications of machine translation, we relied on heuristics and domain knowledge to accommodate a small training set. We focus on a simple grammar and a restricted subset of the English language, making strong assumptions about the content of the event descriptions. Given more training data, these assumptions could likely be lifted. Additionally we intend to learn a language model from the event descriptions, and to use the learned model to generate storylines that more closely match the style and prose of the training data.

This chapter describes the first steps in building a translator between sensor data and human-readable descriptions of events. Enhancing the location events with data from the GPS and GSM sensors is an area of future work, as is incorporating events related to phone calls, text messages, system events such as plugging in and unplugging the phone, and application usage such as accessing the calendar or surfing the Internet.

As in many tasks involving natural language, performing a quantitative evaluation is difficult. Qualitatively, the storylines generated by our system captured all of the significant location-based events. However, we intend to perform a more thorough quantitative evaluation which will require us to collect data accompanied by user-provided event descriptions, as opposed to generating the event descriptions ourselves. The quality of the generated storylines could then be evaluated by the users themselves.

This work [141] was awarded 2nd place in the MDC open challenge. Organizing this event required a tremendous effort on the part of the research team at Nokia. The motivation behind the event was to provide the research community with a large real-world dataset that most researchers would not have the resources to collect. Sharing this data required a careful balancing act between user privacy and providing as much detail as is required to extract meaningful results. This event was a great success, and we sincerely hope that it serves as a model for other companies and organizations that have interesting data.

Chapter 5

Conclusion

Sensors spread throughout the environments we inhabit and the devices we interact with and carry on our person provide opportunities for better understanding of human behaviour on both an individual and a group level. Although we pursued three distinct threads, activity recognition, location detection, and generating storylines, when considered together they form the building blocks of a system that can sense, analyze, and report information about human behaviour. We conclude this thesis by summarizing the principal contributions presented in the preceding chapters, by discussing the privacy issues inherent in this work, by describing some potential applications of this work, and finally by presenting directions for future work.

5.1 Summary of Contributions

Throughout the preceding chapters, we have developed a set of tools and techniques for analyzing data captured from wearable sensors and mobile devices. We have demonstrated that rich information about human behaviour, such as what activities we perform and where we spend our time, can be inferred from sensor data readily available on modern smartphones, and can be automatically transcribed into human-language descriptions of events. Throughout this thesis, there is an emphasis on empirical evaluation, and in total we analysed over 250 GB of real data.

The geometric template matching (GeTeM) algorithm, presented in Chapter 2, while motivated by the goal of performing activity recognition from accelerometer data in real time on an embedded device, is a general time-series classification technique. GeTeM was evaluated on a set of benchmark time series tasks, ranging from classification to clustering, where it was shown to outperform state of the art techniques.

For location detection, we presented the subjective location labelling problem and an algorithm suited for real time operation a mobile phone. We provide strong empirical evidence suggesting a multinomial model for wifi signal strengths, and present an algorithm for clustering wifi signals using this model. Our approach is based on an efficient stochastic optimization algorithm for the hierarchical Dirichlet process (HDP) model, which has an advantage over prior approaches in that it is not agglomerative but does not require the number of clusters to be prespecified. A semi-supervised learning algorithm is presented for incorporating unlabelled wifi measurements to improve the performance of a location classifier. This approach is evaluated on over a year's worth of wifi measurements, and shows superior performance when compared to existing approaches.

Finally, we consider how the results of these types of analysis can be presented to a user. A tool was developed for visualizing a large dataset provided by Nokia Research as a part of the 2012 Nokia© Mobile Data Challenge (MDC) challenge, and a framework was presented for translating between sensor data and human language descriptions of events. Trained on data synthesized using the visualization tool, a translator was able to produce legible, accurate descriptions of location-based events from a collection of wifi measurements.

5.2 Privacy Issues

Research involving data from human subjects must be done with an understanding of the ethical and privacy-related implications. Technological advances have long been at odds with privacy, dating back to the development of photography in the late 1800s [142]. In the 1960s and 1970s, advances in electronics enabled governments to increase the amount of data that could be collected and processed, leading to concerns about *information privacy* [143]. In an effort to balance the utility of data collection with the privacy issues, governments established policies governing fair information practices, such as the influential US Privacy Act of 1974. The principles of fair information practices underlying most privacy legislation are based on the work of Alan Westin [144] and can be summarized as follows [143]:

- **Openness and transparency:** Data collection and record keeping should not be kept secret. The public has a right to be aware of what personal information is being collected and recorded.
- **Individual participation:** A person should have the right to see and correct any records of their personal data.
- **Collection limitation:** The amount of data collected should not exceed what is necessary, and should not be stored longer than is necessary.
- **Use limitation:** The purpose of the data collection should be clearly explained, and the data should only ever be used for the stated purpose.
- **Data quality:** Data should be kept up to date, and should be relevant for the specific purpose for which they are collected.
- **Reasonable security:** Safeguards should be in place, and the effort put into keeping the data secure should reflect the sensitivity of the data.

- **Accountability:** Record keepers must be accountable for compliance with these principles.

The MDC presents an interesting case study in balancing the sensitivity of a large set of personal information with a desire to make available such data to the wider research community [126]. Beginning with the data collection effort, fair information practices were strictly adhered to. The nature of the data being collected was clearly explained to the participants, their consent was obtained, tools for deleting recorded data post hoc were provided, and participants could opt out at any time. The data were securely stored by Nokia, and researchers who had access to the data were required to also store it in a prescribed, secure manner. A commitment to respecting and preserving the privacy of the data was obtained prior to distribution, and to ensure accountability data shared with researchers were uniquely watermarked and thus traceable to the particular research group.

To protect user privacy, personally identifying information such as phone numbers were hashed such that they remain unique but not identifiable. An interesting approach was taken to anonymize GPS data wherein the precision with which coordinates were recorded degraded in regions where few people resided. Coordinates in busy areas such as city centres were precisely recorded, while in rural areas they were less precise and thus the location of a person's home could not be accurately identified, for example. By being open about the tradeoffs made between providing useful data to researchers and preserving the privacy of the subjects, the MDC was an example of how companies, which often have the capacity to collect interesting data, can collaborate with the research community. We hope that this serves as a model for future endeavours.

A significant hindrance to informed debate about privacy of personal information is the lack of knowledge about what information can be extracted from sensor data. For instance, none of the users surveyed by Klasjna et al. [145] expressed any concerns about accelerometer data being recorded, and this matches, anecdotally, our experience presenting our work at conferences and workshops. When we explain to participants in our demonstrations that we can tell, for example, who is carrying the phone, when they change their shoes, the speed that they're walking, etc., we generally see a shift in their opinions. Recent work by Wang et al. [146] has demonstrated significant improvements in dead-reckoning from accelerometer data by grounding estimates using signatures from the magnetometer, allowing for accurate indoor localization. It is possible that future work may allow for accurate localization from accelerometer and magnetometer data alone. The fact is that research in this area may be too preliminary to adequately characterize data modalities as truly innocuous, and the consequences of sharing such data cannot be fully understood.

As a concrete example, the Google Android permissions system is set up to describe precisely what access is granted, and allow the user to make an informed decision before installing an application. The two permissions, `ACCESS_COARSE_LOCATION` and `ACCESS_FINE_LOCATION`, represent access to the location-based service (LBS) provided by the operating system, and allow the application to determine the location of the user with varying levels of accuracy. When requested, the user is notified with a clear explanation of what information the application will have access to. However, as we demonstrate in this thesis, the `ACCESS_WIFI_STATE` and `CHANGE_WIFI_STATE` permissions are sufficient for an application to localize a user with high precision, while the user will only perceive the application as having access to the wifi network information.

The research in this thesis can be used to enhance privacy by directly addressing the *collection limitation* described above. We present techniques that can run at the source of the data, distilling only the desired information, such as the activity or the location label, and thus avoiding the need to store the raw data. By only collecting the *information* that is important, one can mitigate unexpected information disclosure that could be possible if the raw data were stored.

The approaches described in this thesis operate in isolation from external services. Traditional LBSs rely on sending one's location data to a central service that then responds with information, such as a map or nearby locations of interest. In contrast, the approach we describe does not require disclosure of any location data, at the expense of being unable to identify locations that the user has not previously labeled. While this may limit the user experience, it can enable location-based services while satisfying the requirements of more privacy-conscious users.

Education is one of the most important aspects of privacy. Without fully understanding the consequences of the use of technology, one cannot make informed decisions. Work in the ubiquitous computing field can benefit society by raising awareness about these issues[1]. We see our work on generating storylines as contributing to the education of users by making clear, in natural language, what kind of information can be extracted from their mobile phone usage. Perhaps by being exposed to these summaries, users can better comprehend the privacy tradeoffs made by incorporating mobile devices into their daily lives.

5.3 Potential Applications

The work presented in this thesis is primarily of practical value, with numerous potential applications. The motivation behind the work in Chapter 2 was activity and gait recognition, which have applications in fitness, security, and health care, for example.

Many products for tracking personal physical fitness are available, generally relying on self-reporting of activity, or in some cases bundled with hardware such as step counters and GPS receivers[2]. The nature of the activity is not considered, only

[1]Provided they are accurately reported, of course. See [147, Section 5] for an interesting discussion on the effect of popular media in reporting privacy-related issues.

[2]For example the Nike+© (`http://nikeplus.nike.com/plus/`) and fitbit© (`http://www.fitbit.com/`) families of products.

the distance travelled or number of steps taken, and using the work in Chapter 2 to infer the physical activity taking place can improve the estimates of the amount of calories burned, for example, and avoid the necessity of self-reporting. Given the ability of GeTeM to detect novel activities (see Section 2.5.1), such a system could be bootstrapped with a few activities, and learn new activities throughout the lifetime of the system.

In the medical community, accelerometry-based measures are frequently used for assessing physical activities of patients in recovery following a stroke [148], and changes in gait have been used to classify different stages of Huntington's disease and cerebral palsy [149]. Most work in this area uses specialized sensors, and thus allowing for widespread, early detection of symptoms using commodity mobile phones would be valuable.

As in the work on grounding events in language [140, 150], learning the correspondence between events in a data stream and the words describing them can enable natural language-based search. For example translating and indexing location-based events would allow one to answer questions such as "*when was the last time I visited the zoo?*" or "*how long, on average, do I spend waiting for the bus?*", for example. A searchable record of one's activities would be an interesting and useful application.

Both gait and presence can be used in multifactor authentication systems. Gait is particular to an individual and hard to mimic, and thus forms a useful biometric marker for the purpose of identification. Location can also be used for authorization, by ensuring that a user is physically present when, for example, they attempt to log onto a computer or enter a secure area. Vendors have recently begun to use systems that verify the presence of a customer's mobile phone during credit card and banking transactions[3].

[3] `http://www.loc-aid.com/`, for example.

The overarching motivation behind the work in this thesis was an automated reverse journal. In such a system, journal entries are automatically generated from data collected from a smartphone. A reverse journal would give insight into behaviours and allow for deep introspection and understanding of one's routines. This is similar to the goals of the Life Log project from the MIT Media Lab [151].

5.4 Future Work

We conclude with a discussion of interesting opportunities for future work, suggested by the work in this thesis.

5.4.1 Automated Parameter Selection for GeTeM

Manual effort is required to select parameters for the time-delay embedding models used in GeTeM. The standard approaches such as examining the spectral properties [38] and the method of false nearest neighbours [39] require manual inspection. Although GeTeM has been shown to be robust to the time-delay embedding parameters, it is still important that these are set correctly. Since a time-delay embedding consists of the raw data, transforming between a model built with one set of parameters to a model with a different set of parameters is trivial, and so the parameters can be updated online with negligible computational cost. Thus, there is a potential for online parameter tuning, which would be useful for long-running systems. Additionally, since GeTeM is computationally efficient, a system could evaluate multiple models with different parameters in parallel, and compare their performance.

5.4.2 Noisy Time-Delay Embedding

There is limited theory on the effect of noise on a time-delay reconstruction of a dynamical system. Work by Jaeger and Kantz [152, 153] showed that least-squares fits of noisy time-delay reconstructions have systematic bias, and demonstrate that total least-squares minimization removes the bias and multi-step reconstruction error minimization methods improve performance considerably. In our work, we did not fit parametric models to the time-delay reconstructions, and instead used them as nonparametric models. Our approach to handling noise is to average across nearest neighbours, and in practice this works well. It would be valuable to consider fitting parametric models using a total least-squares cost function in order to determine whether parametrized time-delay models could better handle noisy data.

5.4.3 Multivariate Time-Delay Embedding

Embedding multivariate data is not considered in this thesis, but would be valuable for modelling multiple sensory input streams in conjunction. Recent work on modelling GPS data using time-delay embeddings [154] simply concatenates the inputs and uses traditional univariate delay embeddings (as in [155]), but this is unsatisfactory, as it is equivalent to embedding the two inputs independently and simply concatenating the models. Thus, correlations between inputs are only superficially considered. There is recent work in the statistical physics literature on multivariate time-delay embedding [156], and incorporating these techniques with the GeTeM algorithm would be interesting to explore.

5.4.4 Location Prediction

The algorithm presented in Chapter 3 for producing a graph of the location clusters was particularly useful for visualizing the mobility patterns of the subjects in the MDC data challenge, as discussed in Chapter 4. It is clear from manually inspecting traces of the subject moving between clusters throughout the day that the trajectories through the cluster space are predictable. It would be interesting to consider two prediction problems, *static prediction* and *dynamic prediction*.

Static prediction is the problem of predicting where a user would be at a specific time of day. From Figure 4.7, it is clear that location of the particular subject typically has low entropy for much of the day. We have observed similar patterns in all of the other subjects. Therefore, one would expect that simply conditioned on the hour of the day, one would be able to predict with fairly high accuracy the location (in cluster-space) of the user. Correlating cluster occupancy with available GPS data would allow grounding of the clusters in physical space, thus allowing for location prediction with respect to physical coordinates.

Dynamic prediction incorporates temporal context, that is the goal is to predict where the user will be conditioned on their previous location, and possibly the time of day as well. Again, we have observed that trajectories are fairly consistent, and thus we expect that dynamic prediction could also be performed with high accuracy.

Both of these problems were considered using GSM cell-occupancy data [157, 158], and GPS data [157], and it would be interesting to extend this work to finer-resolution location data available both indoors and outdoors, as provided by our approach.

5.4.5 Reverse Journal

Finally, as described in Section 5.3, much of the work in this thesis was inspired by the notion of a reverse journal, or a persistent transcription of daily events. In Chapter 2 we demonstrate the ability to detect what activities a person is engaged in, and in Chapter 3 we demonstrate how to detect where they are. Finally, the work in Chapter 4 describes a method for generating natural-language descriptions of the output of these algorithms. Thus, we provide a framework for generating human-readable histories from data collected by a mobile phone, forming the building blocks of a reverse journal application.

Appendix A

Android Data Collection Platform

A data collection platform for the Android operating system was developed for the purpose of collecting much of the data used in this thesis. This system has been released as an open-source project[1], under the permissive MIT license[2]. The software consists of a background service that brokers messages between input plugins that collect data from the various sensors, and output plugins that consume the data for the purpose of logging or analysis. The system is built in a modular fashion so that new plugins can easily be added to the system. With this work, we hope to develop a platform on which other researchers can build, and hopefully contribute to.

Similar systems exist for data collection and analysis such as the Darwin System from Dartmouth and Nokia that runs on the Nokia and Apple iOS platforms [159], and the Context Logger application for the iOS platform from the University of Passau[3]. However, these systems are not open-source projects, and are therefore not available outside the research groups that developed them. The OpenDataKit platform from Google and the University of Washington runs on the Android platform with an emphasis on collecting survey data from large groups of mobile

[1]Available at `https://github.com/jwf/Humansense-Android-App/`

[2]`http://opensource.org/licenses/mit-license.php`

[3]`http://wiki.esl.fim.uni-passau.de/index.php/Context_Logger_for_iPhone_OS`

users[4]. Most similar to our project is the Funf Open Sensing Framework from MIT[5], also for the Android platform. We note that the OpenDataKit and Funf systems did not exist when we began this work.

Background Service

The core of the data collection platform is the background service. The background service serves three main functions: it serves as a message broker between input and output plugins, it provides a user interface allowing plugins to specify settings and preferences that can be configured by the user, and it provides a mechanism for uploading data from the phone to a server. As the background service is intended to run as a persistent service on the phone, the overarching factor is performance. While good design patterns were mostly adhered to, namely loose coupling between plugins, and abstraction for specifying user preferences, efficiency was prioritized.

The message broker is based on the publish-subscribe pattern (pub/sub). For performance reasons, the input plugins specify their message schemas with an inner class implementing the DataPacket interface. The resulting system is more tightly coupled than typical pub/sub systems where the schemas are decoupled from the implementation.

Input plugins publish data by calling a `write` method and passing a `DataPacket`. The service broadcasts messages to all output plugins, which act as subscribers. Output plugins only need to implement the `onDataReceived` method, which gets passed every `DataPacket` that is published by the input plugins. As all messages are broadcast to all output plugins, it is the responsibility of the output plugin to filter the messages that it is interested in. New plugins must be added to the list of plugins in the `PluginFactory` class, and must be specified at compile-time. At the expense of loose coupling, plugins must be compiled with the entire system,

[4]`http://opendatakit.org/`

[5]`http://funf.media.mit.edu/`

but this is much to the benefit of performance, as the message serialization and parsing can be optimized as compile time, rather than dynamically at run time.

Plugins can specify options and settings to be presented to the user. A significant effort was put into abstracting the complex preferences API provided by the Android SDK, and a simple interface is provided for soliciting preferences using text input fields, checkboxes, lists, and graphical sliders. Additionally, each plugin is automatically assigned a setting to enable or disable the plugin, allowing the user to easily control which plugins are active, and provides a clear indication of what data is being collected.

A mechanism for transferring log and data files is provided, allowing files to be automatically uploaded to a server. Files can be uploaded over the wifi or cellular data connection, and an option is provided to allow the user to disable uploading over the cellular data connection. Data can be uploaded automatically at regular intervals, or manually via a button in the user interface.

Input and Output Plugins

Included with the platform are a number of input plugins for collecting data from the sensors and radios, and output plugins for logging and analyzing the data. Currently, plugins support data collection from the Bluetooth, GSM, GPS, and wifi radios, and the accelerometer, magnetic field, and temperature sensors. For output, plugins are available to store the data in compressed log files and to dump data to the Android logging interface (mainly for debugging purposes). Additionally, a plugin implementing the activity detection system in Chapter 2 is available for performing activity and gait recognition in real time. Both training models and classifying accelerometer data can be performed entirely on the device, with the results displayed on the screen or streamed to a separate computer for easier visualization. Finally, a prototype of a location detection algorithm is implemented which uses a variant of DBScan [119] to cluster wifi locations. This was a precursor to the work described in Chapter 3, and can perform indoor location detection with a high accuracy. However, as noted in Chapter 3, a deficiency of this approach is that over a long period of time clusters for nearby locations tend to get merged, and so the accuracy degrades over time. Implementing our alternative approach remains as future work.

Bibliography

[1] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR time series classification/clustering homepage, 2006. URL `www.cs.ucr.edu/~eamonn/time_series_data/`.

[2] D. Klein and C.D. Manning. Accurate unlexicalized parsing. In *ACL*, 2003.

[3] J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. *Lecture Notes in Computer Science*, 3968:1–16, 2006.

[4] P. Maass and M. Rajagopalan. That's not my phone, it's my tracker - NYTimes.com. *The New York Times*, July 13 2012.

[5] D. Lazer, A. Pentland, L. Adamic, S. Aral, A.-L. Barabási, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, and M. Van Alstyne. Computational social science. *Science*, 323(5915):721–723, 2009.

[6] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[7] C. Wang, J. Paisley, and D.M. Blei. Online variational inference for the hierarchical Dirichlet process. In *AISTATS*, 2011.

[8] G.E. Batista, X. Wang, and E.J. Keogh. A complexity-invariant distance measure for time series. In *Proc. of the SIAM Intl. Conf. on Data Mining*, pages 699–710, 2011.

[9] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *Proc. of the IEEE Intl. Conf. on Data Mining*, pages 273–280, 2001.

[10] D. Ge, N. Srinivasan, and S. Krishnan. Cardiac arrhythmia classification using autoregressive modeling. *BioMedical Engineering OnLine*, 1(1):5, 2002.

[11] M. Corduas and D. Piccolo. Time series clustering and classification by the autoregressive metric. *Computational Statistics & Data Analysis*, 52(4): 1860–1872, 2008.

[12] T. Sauer, J.A. Yorke, and M. Casdagli. Embedology. *Journal of Statistical Physics*, 65(3):579–616, 1991.

[13] H. Kantz and T. Schreiber. *Nonlinear time series analysis*. Cambridge University Press, 2004.

[14] M. Small. *Applied nonlinear time series analysis: applications in physics, physiology and finance*. World Scientific series on nonlinear science: Monographs and treatises. World Scientific, 2005.

[15] F. Takens. Detecting strange attractors in turbulence. *Dynamical systems and turbulence*, 898(1):365–381, 1981.

[16] J.A. Ward, P. Lukowicz, and H.W. Gellersen. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems Technology*, 2: 1–23, January 2011. ISSN 2157-6904.

[17] J.C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974.

[18] D.L. Davies and D.W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224 –227, April 1979.

[19] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning.* MIT Press, 2006.

[20] X. Xi, E. Keogh, C. Shelton, L. Wei, and C.A. Ratanamahatana. Fast time series classification using numerosity reduction. In *Proc. of the IEEE Intl. Conf. on Machine Learning*, pages 1033–1040, 2006.

[21] H. Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[22] C.A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints. In *Proc. of SIAM Intl. Conf. on Data Mining*, pages 11–22, 2004.

[23] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is ”nearest neighbor” meaningful? In *Proc. of the Intl. Conf. on Database Theory*, pages 217–235, 1999.

[24] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. *Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[25] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. of the ACM Intl. Conf. on Management of Data*, SIGMOD ’94, pages 419–429, 1994.

[26] K. Chan and A.W.C. Fu. Efficient time series matching by wavelets. In *Proc. of the Intl. Conf. on Data Engineering*, pages 126–133. IEEE, 1999.

[27] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *ACM SIGMOD Record*, volume 26, pages 289–300, 1997.

[28] C.S. Burrus, R.A. Gopinath, and H. Guo. *Introduction to wavelets and wavelet transforms: a primer*. Prentice Hall, 1998.

[29] J. Durbin. The fitting of time-series models. *Review of the Intl. Statistical Institute*, 28(3):233–244, 1960.

[30] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[31] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[32] P. Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems*, pages 648–654. MIT Press, 1997.

[33] B.H. Juang and L.R. Rabiner. A probabilistic distance measure for hidden markov models. *AT&T Technical Journal*, 64(2):391–408, 1985.

[34] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004.

[35] E.A. Heinz, K.S. Kunze, S. Sulistyo, H. Junker, P. Lukowicz, and G. Tröster. Experimental evaluation of variations in primary features used for accelerometric context recognition. In *Proc. of the European Symposium on Ambient Intelligence*, pages 252–263, 2003.

[36] N. Ravi, N. Dandekar, P. Mysore, and M.L. Littman. Activity recognition from accelerometer data. In *Proc. of the Conf. on Innovative Applications of Artificial Intelligence*, pages 1541–1546. AAAI Press, 2005.

[37] T. Huynh and B. Schiele. Analyzing features for activity recognition. In *Proc. of the Joint Conf. on Smart Objects and Ambient Intelligence*, pages 159–163. ACM, 2005.

[38] T. Buzug and G. Pfister. Optimal delay time and embedding dimension for delay-time coordinates by analysis of the global static and local dynamical behavior of strange attractors. *Physics Review A*, 45(10):7073–7084, 1992.

[39] M.B. Kennel, R. Brown, and H.D.I. Abarbanel. Determining embedding dimension for phase space reconstruction using the method of false nearest neighbors. *Physics Review A*, 45(6):3403–3411, 1992.

[40] A. Galka. *Topics in nonlinear time series analysis: with implications for EEG analysis*. World Scientific, 2000.

[41] J. Röschke and J. Aldenhoff. The dimensionality of human's electroencephalogram during sleep. *Biological Cybernetics*, 64(4):307–313, 1991.

[42] R. Esteller, G. Vachtsevanos, J. Echauz, T. Henry, P. Pennell, C. Epstein, R. Bakay, C. Bowen, and B. Litt. Fractal dimension characterizes seizure onset in epileptic patients. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, volume 4, pages 2343–2346, 1999.

[43] K. Bush and J. Pineau. Manifold embeddings for model-based reinforcement learning under partial observability. In *Advances in Neural Information Processing Systems*, volume 22, pages 189–197, 2009.

[44] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of the ACM Symposium on Theory of computing*, pages 604–613, 1998.

[45] L. Daniel. Faster retrieval with a two-pass dynamic-time-warping lower bound. *Pattern Recognition*, 42(9):2169–2180, September 2009.

[46] A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.C. Ivanov, R.G. Mark, J.E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[47] S.C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[48] J.H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236—244, 1963.

[49] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[50] B. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. *Multiple Hypotheses Testing*, pages 100–115, 1988.

[51] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, et al. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, pages 32–41, 2008.

[52] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines: and other kernel-based learning methods.* Cambridge University Press, 2000.

[53] M. Mahdaviani and T. Choudhury. Fast and scalable training of semi-supervised CRFs with application to activity recognition. In *Advances in Neural Information Processing Systems*, December 2007.

[54] M.S. Nixon, T. Tan, and R. Chellappa. *Human identification based on gait.* Springer, 2006.

[55] D. Gafurov, E. Snekkenes, and P. Bours. Gait authentication and identification using wearable accelerometer sensor. In *2007 IEEE Workshop on Automatic Identification Advanced Technologies*, pages 220–225, 2007.

[56] H.J. Ailisto, M. Lindholm, J. Mantyjarvi, E. Vildjiounaite, and S.M. Makela. Identifying people from gait pattern with accelerometers. In *Society of Photo-Optical Instrumentation Engineers Conf. Series*, volume 5779, pages 7–14, 2005.

[57] M. Bächlin, J. Schumm, D. Roggen, and G. Töster. Quantifying gait similarity: user authentication and real-world challenge. *Advances in Biometrics*, pages 1040–1049, 2009.

[58] W. Pentney and M. Meila. Spectral clustering of biological sequence data. In *Proc. of the National Conf. on Artificial Intelligence*, volume 20, pages 845–850. AAAI Press, 2005.

[59] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[60] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[61] A. Subramanya, A. Raj, J. Bilmes, and D. Fox. Recognizing activities and spatial context using wearable sensors. In *Proc. of Uncertainty in Artificial Intelligence*, 2006.

[62] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[63] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI-94 workshop on knowledge discovery in databases*, pages 229–248, 1994.

[64] J. Zhu, S. Rosset, H. Zou, and T. Hastie. Multi-class AdaBoost. Technical report, Department of Statistics, University of Michigan, 2005.

[65] Y. Freund and R. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[66] I. Mukherjee and R.E. Schapire. A theory of multi-class boosting. In *Advances in Neural Information Processing Systems*, 2010.

[67] S.J. Sheather and Jones M.C. A reliable data-based bandwidth selection method for kernel density estimation. *Royal Statistical Society, Series B*, pages 683–690, 1991.

[68] J. Hightower and G. Borriello. A survey and taxonomy of location sensing systems for ubiquitous computing. UW CSE 01-08-03, University of Washington, Department of Computer Science and Engineering, Seattle, WA, August 2001.

[69] H. Landau. Precise kinematic GPS positioning. *Journal of Geodesy*, 63: 85–96, 1989.

[70] A.H. Sayed, A. Tarighat, and N. Khajehnouri. Network-based wireless location: challenges faced in developing techniques for accurate wireless location information. *IEEE Signal Processing Magazine*, 22(4):24–40, July 2005.

[71] Federal Communications Commission et al. FCC wireless 911 requirements fact sheet, 2001.

[72] European Commision. Commission pushes for rapid deployment of location enhanced 112 emergency services. *Press Release IP/03/1122*, 2003.

[73] M. Weckström, M. Spirito, and V. Ruutu. Mobile station location. In T. Halonen, J. Romero, and J. Melero, editors, *GSM, GPRS, and EDGE*

Performance: Evolution towards 3G/UMTS, chapter 4. John Wiley & Sons, 2nd edition, 2003.

[74] G. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X.P. Costa, and B. Walke. The IEEE 802.11 universe. *IEEE Communications Magazine*, 48(1):62–70, January 2010.

[75] P. Castro, P. Chiu, T. Kremenek, and R. Muntz. A probabilistic location service for wireless network environments. In *Proc. of Ubiquitous Computing*, pages 18–34, 2001.

[76] P. Bahl and V.N. Padmanabhan. RADAR: An inbuilding RF-based user location and tracking system. In *INFOCOM*, 2000.

[77] P. Bahl, V.N. Padmanabhan, and A. Balachandran. Enhancements to the RADAR user location and tracking system. MSR-TR 2000-12, Microsoft Research, 2000.

[78] M. Youssef and A. Agrawala. The Horus WLAN location determination system. In *MobiSys*, 2005.

[79] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen. A probabilistic approach to wlan user location estimation. *Intl. Journal of Wireless Information Networks*, 9(3):155–164, 2002.

[80] A. Haeberlen, E. Flannery, A.M. Ladd, A. Rudys, D.S. Wallach, and Kavraki L.E. Practical robust localization over large-scale 802.11 wireless networks. In *Proc. of ACM Conf. on Mobile Computing and Networking (MobiCom)*, pages 70–84, 2004.

[81] Y. Ji, S. Biaz, S. Pandey, and P. Agrawal. ARIADNE: A dynamic indoor signal map construction and localization system. In *MobiSys*, 2006.

[82] K.K. Chintalapudi, A.P. Iyer, and V.N. Padmanabhan. Indoor localization without the pain. In *MobiCom*, 2010.

[83] S. Cavalieri. WLAN-based outdoor localisation using pattern matching algorithm. *Intl. Journal of Wireless Information Networks*, 14:265–279, 2007.

[84] T. Deasy and W. Scanlon. Simulation or measurement: The effect of radio map creation on indoor WLAN-based localisation accuracy. *Wireless Personal Communications*, 42:563–573, 2007.

[85] A. Tsui, Y.-H. Chuang, and H.-H. Chu. Unsupervised learning for solving rss hardware variance problem in wifi localization. *Mobile Networks and Applications*, 14:677–691, 2009.

[86] S. Sen, B. Radunovic, R.R. Choudhury, and T. Minka. You are facing the Mona Lisa: Spot localization using PHY layer information. In *Proc. of MobiSys*, 2012.

[87] F. Belloni, V. Ranki, A. Kainulainen, and A. Richter. Angle-based indoor positioning system for open indoor environments. In *Workshop on Positioning, Navigation and Communication 2009*, pages 261–265. IEEE, 2009.

[88] G. Durgin, T.S. Rappaport, and H. Xu. Measurements and models for radio path loss and penetration loss in and around homes and trees at 5.85 GHz. *IEEE Transactions on Communications*, 46:1484–1496, 1998.

[89] W.C.Y. Lee and D.J.Y. Lee. Microcell prediction in dense urban area. *IEEE Transactions on Vehicular Technology*, 47(1):246–253, 1998.

[90] H.R. Anderson. A ray tracing propagation model for digital broadcast system in urban area. *IEEE Transactions on Broadcasting*, 39(3):309–317, 1993.

[91] M.S. Grewal and A.P. Andrews. *Kalman Filtering Theory and Practice.* Prentice Hall, 1993.

[92] Cisco Systems Inc. 20 myths of wi-fi interference. Whitepaper, 2007.

[93] S.S. Shapiro and M.B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3), 1965.

[94] B. Efron. Missing data, imputation, and the bootstrap. *Journal of the American Statistical Association*, 89(426):463–475, 1994.

[95] P. Tao, A. Rudys, A.M. Ladd, and D.S. Wallach. Wireless LAN location-sensing for security applications. In *Proc. of AAAI Conf. on Aritifical Intelligence (AAAI)*, pages 11–20, 1996.

[96] Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476), 2006.

[97] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[98] P. Müller and F.A. Quintana. Nonparametric bayesian data analysis. *Statistical science*, pages 95–110, 2004.

[99] S.G. Walker, P. Damien, P.W. Laud, and A.F.M. Smith. Bayesian nonparametric inference for random distributions and related functions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3): 485–527, 1999.

[100] M.I. Jordan. Bayesian nonparametric learning: Expressive priors for intelligent systems. In R. Dechter, H. Geffner, and J. Halpern, editors, *Heuristics, Probability and Causality: A Tribute to Judea Pearl*. College Publications, 2010.

[101] M.I. Jordan. Hierarchical models, nested models and completely random measures. In M.-H. Chen, D. Dey, P. Mueller, D. Sun, and K. Ye, editors, *Frontiers of Statistical Decision Making and Bayesian Analysis: In Honor of James O. Berger*. Springer, 2010.

[102] E.B. Fox. *Bayesian Nonparametric Learning of Complex Dynamical Phenomena*. Ph.D. thesis, MIT, Cambridge, MA, 2009.

[103] T.S. Ferguson. A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973.

[104] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[105] D. Blackwell and J.B. MacQueen. Ferguson distributions via Pólya urn schemes. *Annals of Statistics*, 1:353–355, 1973.

[106] D.J. Aldous. Exchangeability and related topics. *Lecture notes in Mathematics*, 1117:1–198, 1985.

[107] Y.W. Teh, K. Kurihara, and M. Welling. Collapsed variational inference for HDP. *Advances in neural information processing systems*, 20(20):1481–1488, 2008.

[108] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[109] D.M. Blei and M.I. Jordan. Variational methods for the Dirichlet process. In *Proc. of the Intl. Conf. on Machine learning*, page 12, 2004.

[110] A. Asuncion, M. Welling, P. Smyth, and Y.W. Teh. On smoothing and inference for topic models. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 27–34, 2009.

[111] M.A. Sato. Online model selection based on the variational bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[112] L. Bottou and N. Murata. Stochastic approximations and efficient learning. *The Handbook of Brain Theory and Neural Networks, Second edition*, 2002.

[113] M.D. Hoffman, D.M. Blei, and F. Bach. Online learning for latent Dirichlet allocation. *Advances in Neural Information Processing Systems*, 23:856–864, 2010.

[114] R.M. Neal. *Handbook of Markov Chain Monte Carlo*, chapter 5: MCMC Using Hamiltonian Dynamics. CRC Press, 2011.

[115] R.M. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–741, 2003.

[116] M.D. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *ArXiv e-prints*, November 2011. URL `http://arxiv.org/abs/1111.4246v1`.

[117] M. Wahabzada and K. Kersting. Larger residuals, less work: Active document scheduling for latent Dirichlet allocation. In *Machine Learning and Knowledge Discovery in Databases*, volume 6913, pages 475–490, 2011.

[118] J. Paisley, D.M. Blei, and M.I. Jordan. Variational Bayesian inference with stochastic search. In *Proc. of the Intl. Conf. on Machine Learning*, 2012.

[119] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 1996, pages 226–231, 1996.

[120] F. Khan, D. Phung, and S. Venkatesh. Robust wifi localization using received signal strength. Technical report, Department of Computing, Curtin University of Technology, 2007.

[121] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[122] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.

[123] J.H. Kang, W. Welbourne, B. Stewart, and G. Borriello. Extracting places from traces of locations. *Mobile Computing and Communications Review*, 9 (3):58–68, July 2005.

[124] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[125] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[126] J. K. Laurila, D. Gatica-Perez, I. Aad, J. Blom, O. Bornet, T. Do, O. Dousse, J. Eberle, and M. Miettinen. The mobile data challenge: Big data for mobile computing research. In *Proc. Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf.. on Pervasive Computing*, Newcastle, UK, June 2012.

[127] A. Bies, M. Ferguson, K. Katz, R. MacIntyre, V. Tredinnick, G. Kim, M.A. Marcinkiewicz, and B. Schasberger. Bracketing Guidelines for Treebank II Style Penn Treebank Project. Technical report, University of Pennsylvania, 1995.

[128] M.C. De Marneffe, B. MacCartney, and C.D. Manning. Generating typed dependency parses from phrase structure parses. In *Proc. of the Intl. Conf. on Language Resources and Evaluation*, volume 6, pages 449–454, 2006.

[129] J. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the Intl. Conf. on Machine Learning*, pages 282–289, 2001.

[130] J.R. Finkel, T. Grenager, and C.D. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*, 2005.

[131] P. Koehn. *Statistical machine translation.* Cambridge University Press, 2010.

[132] W.A. Gale and K.W. Church. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1):75–102, 1993.

[133] Y.Y. Wang and A. Waibel. Decoding algorithm in statistical machine translation. In *Proc. of the European chapter of the Association for Computational Linguistics*, pages 366–372, 1997.

[134] U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Yamada. Fast decoding and optimal decoding for machine translation. In *Proc. of the Association for Computational Linguistics*, pages 228–235, 2001.

[135] K. Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, December 1999.

[136] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, pages 79–86, 2005.

[137] J. Uszkoreit, J.M. Ponte, A.C. Popat, and M. Dubiner. Large scale parallel document mining for machine translation. In *Proc. of the Intl. Conf. on Computational Linguistics*, pages 1101–1109, 2010.

[138] Z. Shen and K. Ma. MobiVis: A visualization system for exploring mobile data. In *Proc. of the IEEE PacificVis Symposium*, pages 175–182, 2008.

[139] D. Feldman, A. Sugaya, and D. Rus. An effective coreset compression algorithm for large scale sensor networks. In *Proc. ACM/IEEE Conf. on Information Processing in Sensor Networks*, 2012.

[140] M. Fleischman and D. Roy. Grounded language modeling for automatic speech recognition of sports video. In *Proc. of Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 121–129, 2008.

[141] J. Frank, S. Mannor, and D. Precup. Generating storylines from sensor data. In *Proc. Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf.. on Pervasive Computing*, Newcastle, UK, June 2012.

[142] S. Warren and L. Brandeis. The right to privacy. *Harvard Law Review*, 4: 193–220, 1890.

[143] M. Langheinrich. Privacy by design: principles of privacy-aware ubiquitous systems. In *Proc. of Ubiquitous Computing*, pages 273–291. Springer, 2001.

[144] A.F. Westin. *Privacy and Freedom.* New York: Athenum, 1967.

[145] P. Klasjna, S. Consolvo, T. Choudhury, and Beckwith. R. Exploring privacy concerns about personal sensing. In *Proc. of Pervasive*, May 2009.

[146] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R.R. Choudhury. No need to war-drive: Unsupervised indoor localization. In *Proc. of the Intl. Conf. on Mobile Systems, Applications, and Services*, pages 197–210, 2012.

[147] R. Beckwith and S. Mainwaring. Privacy: Personal information, threats, and technologies. In *Proc. of the Intl. Symposium on Technology and Society*, pages 9–16. IEEE, 2005.

[148] N. Gebruers, C. Vanroy, S. Truijen, S. Engelborghs, and P.P. De Deyn. Monitoring of physical activity after stroke: a systematic review of accelerometry-based measures. *Archives of physical medicine and rehabilitation*, 91(2):288–297, 2010.

[149] A. Opheim, J.L. McGinley, E. Olsson, J.K. Stanghelle, and R. Jahnsen. Walking deterioration and gait analysis in adults with spastic bilateral cerebral palsy. *Gait & Posture*, 2012.

[150] M.B. Fleischman. *Grounding Language in Events.* Ph.D. thesis, MIT, Cambridge, MA, 2008.

[151] N. Eagle and A. Pentland. The organization life log. In *Proc. of the IEEE Intl. Symposium on Wearable Computing*, pages 256–257, 2003.

[152] L. Jaeger and H. Kantz. Unbiased reconstruction of the dynamics underlying a noisy time series. *Chaos*, 6, 1996.

[153] H. Kantz and L. Jaeger. Improved cost functions for modelling of noisy chaotic time series. *Physica D*, 109:59–69, 1997.

[154] M. De Domenico, A. Lima, and M. Musolesi. Interdependence and predictability of human mobility and social interactions. In *Proc. Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf.. on Pervasive Computing*, Newcastle, UK, June 2012.

[155] L. Cao, A. Mees, and K. Judd. Dynamics from multivariate time series. *Physica D: Nonlinear Phenomena*, 121(1-2):75–88, 1998.

[156] S.P. Garcia and J.S. Almeida. Multivariate phase space reconstruction by nearest neighbor embedding with different time delays. *Phys. Rev. E*, 72: 027205, August 2005.

[157] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from GPS traces using hierarchical conditional random fields. *The Intl. Journal of Robotics Research*, 26(1):119–134, 2007.

[158] M.A. Bayir, M. Demirbas, and N. Eagle. Discovering spatiotemporal mobility profiles of cellphone users. In *IEEE Intl. Symposium on a World of Wireless Mobile and Multimedia Networks*, pages 1–9, June 2009.

[159] E. Miluzzo, C.T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A.T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *Proc. of the Intl. Conf. on Mobile Systems, Applications, and Services*, pages 5–20. ACM, 2010.