DISSERTATION

DATA MINING TECHNIQUES FOR TEMPORAL POINT PROCESSES APPLIED TO INSURANCE CLAIMS DATA

Submitted by

Todd Ashley Iverson

Department of Statistics

In partial fulfillment of the requirements

for the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2008

COLORADO STATE UNIVERSITY

June 19, 2008

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY TODD ASHLEY IVERSON ENTITLED DATA MINING TECHNIQUES FOR TEMPORAL POINT PROCESSES APPLIED TO INSURANCE CLAIMS DATA BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Asa Ben-Hur (Co-adviser)

Hariharan K. Iyer (Co-adviser)

Ryan T. Elmore

Haonan Wang

Jean Opsomer (Acting Chair)

ABSTRACT OF DISSERTATION

DATA MINING TECHNIQUES FOR TEMPORAL POINT PROCESSES APPLIED TO INSURANCE CLAIMS DATA

We explore data mining on databases consisting of insurance claims information. This dissertation focuses on two major topics we considered by way of data mining procedures. One is the development of a classification rule using kernels and support vector machines. The other is the discovery of association rules using the Apriori algorithm, its extensions, as well as a new association rules technique.

With regard to the first topic we address the question – can kernel methods using an SVM classifier be used to predict patients at risk of type 2 diabetes using three years of insurance claims data? We report the results of a study in which we tested the performance of new methods for data extracted from the MarketScan® database. We summarize the results of applying popular kernels, as well as new kernels constructed specifically for this task, for support vector machines on data derived from this database. We were able to predict patients at risk of type 2 diabetes with nearly 80% success when combining a number of specialized kernels.

The specific form of the data, that of a timed sequence, led us to develop two new kernels inspired by dynamic time warping. The Global Time Warping (GTW) and Local Time Warping (LTW) kernels build on an existing time warping kernel by including the timing coefficients present in classical time warping, while providing a solution for the diagonal dominance present in most alignment methods. We show that the LTW kernel performs significantly better than the existing time warping kernel when the times contained relevant information.

With regard to the second topic, we provide a new theorem on closed rules that could help substantially improve the time to find a specific type of rule. An insurance claims database contains codes indicating associated diagnoses and the resulting procedures for each claim. The rules that we consider are of the form *diagnoses imply procedures*. In addition, we introduce a new class of interesting association rules in the context of medical claims databases and illustrate their potential uses by extracting example rules from the MarketScan® database.

Todd Ashley Iverson
Department of Statistics
Colorado State University
Fort Collins, Colorado 80523
Summer 2008

ACKNOWLEDGEMENTS

I would like to thank my advisors, Asa Ben-Hur and Hari Iyer for all of their support and guidance. I came to CSU with the goal of finding a good advisor, and I was lucky to end up with two of the best! In particular, I would like to thank Asa for his patience and guidance. I started this project with little knowledge in the area of Computer Science, and Asa did an excellent job guiding me to the tools needed to succeed.

I would also like to thank Caterpillar Inc. and Dr. Syamala Srinivasan for the opportunity to start working on this great project. Without Dr. Srinivasan's inspiration, none of this would have happened.

Finally, I would like to thank Thomson Medstat for the opportunity to do such exciting research based on their MarketScan® database.

DEDICATION

To my parents, for all of their support and guidance; and especially to Jennifer and Cora, for giving my life purpose.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

This dissertation is about mining useful information from insurance claims databases. The problem was motivated by Caterpillar Inc., who provided us with the Thomson Medstat MarketScan® insurance claims database [1]; and tasked us with finding methods for extracting useful information. The amount of information in the database is vast, as it contains up to five years worth of insurance claims for over 9.6 million patients. As health care cost is an important current topic, the availability of data and the computational capabilities of modern computers make this an excellent and relevant topic for current research.

It is not hard to imagine a multitude of questions that can be asked: Pick any disease of interest. Can we use claims data to identify individuals who are at risk of this disease? How early can the risk of the disease be detected? What can we learn about the treatment of the disease? Can we identify different treatments for this disease and understand why they were selected? Did the different treatments differ in their cost? These are the questions that we addressed.

In response to the first two questions, we developed methods that allow for identifying patients at risk of type 2 diabetes using only their insurance claims and investigated the methods' ability to detect risk of type 2 diabetes well before its onset. To answer the last two questions, we give results that will allow one to tailor existing association rule mining techniques to claims data, as well as redefine an "interesting" rule to account for the cost in different procedures used for the same set of diagnoses.

1.1 Diabetes, Kernels and Support Vector Machines

Diabetes is a major source of this nation's medical expense [2] that differs from many health issues in two ways. First, type 2 diabetes is preceded by a pre-diabetes phase that allows for early detection of those at risk of type 2 diabetes [3]. Second, the onset of the disease can be delayed or completely prevented through life-style changes in the form of diet and exercise [3]. On the other hand, a failure to detect and treat diabetes can lead to a progression of the disease. A recent study [4] showed that more than 25% of untreated patients with type 2 diabetes experienced worsening glycemic control.

It has been shown that detecting those at risk of diabetes can be done in a very accurate way (0.859 area under the ROC curve) using age, sex, ethnicity, blood pressure, body mass index, parental or sibling history of diabetes, and various measurements on the patient's blood [5]. The tests used to make these blood measurements is time consuming, costly, and inconvenient [5]. We wish to determine if a patient is at risk of type 2 diabetes using only the patient's insurance claims data. We have developed methods that allow accurate prediction of patients at risk of type 2 diabetes based on claims data alone. These methods can be used as an initial screening, cutting down on the number of individuals that need the costly test and helping identify patients that might not have otherwise been screened. We know of no other attempts to solve this specific problem.

We use kernel methods and support vector machines to classify patients at risk of type 2 diabetes. Kernel methods provide ways to combine different forms of information and can be applied even when the data is not a fixed length vector. Some background information on support vector machines, kernels, and specific existing methods relevant to these data is given in Chapter 2. A new kernel we designed specifically with these data in mind is discussed in Chapter 3. Experiments to assess their performance when predicting type 2 diabetes are found in Chapter 4.

1.2 Mining Association Rules from Claims Data

The second task involved exploring the database and identifying consistent, as well as unusual, patterns. Insurance databases contain much information on medical diagnoses, procedures, and expenditures. Health care cost is a current topic of national concern and any information on how this money is being spent is of interest.

One method for learning relationships from a database of claims is through mining association rules. There are a few specific needs that are not addressed by current rule-mining algorithms. First, we need a more efficient method for mining rules that are specifically of the form *diagnosis implies procedure.* Also, the standard definition of an interesting rule does not allow for the inclusion of cost information. Finally, current implementation of these algorithms leaves it up to the user to collect sets of related rules.

We give a result that will help speed future algorithms when looking for rules of the form *diagnosis implies procedure* to address the first need. To address the second and third need, we develop a new definition of interesting families of rules that incorporate cost information. Details can be found in Chapter 5

1.3 Insurance Claims Data

Before proceeding to the main body of work, we would like to familiarize the reader with the insurance claims database. First, we will give specific information on the MarketScan® database. Then we point out the properties that make working with these data an interesting and research-worthy activity.

1.3.1 Description of the Database

All the proposed research was inspired by and conducted on Thomson Medstat's MarketScan® database of health insurance claims from 1999 to 2003. The database consists of private sector health claims collected from over 100 employers and includes over 10 million employees and dependents. The data contains the medical

service uses and medical expenditures for insured employees, early retirees, COBRA continuations, and dependents covered under the employees' health plan.

For each person included in the database, referred to as an enrollee, individual health insurance claims are split by type: inpatient claims, outpatient claims, prescription drug claims, and enrollment information. The inpatient claims include services associated with a hospital admission. The outpatient claims are related to services that occur at a doctor's office, emergency room, or other outpatient facility.

All inpatient and outpatient claims contain extensive information on the associated expenditures. The inpatient and outpatient claims also contain associated diagnosis and procedure codes. The diagnosis codes come in the form of ICD-9-CM coded variables. The International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM) is the recognized method for coding diagnoses and procedures in United States hospitals. The procedure codes are coded either using the ICD-9-CM, CPT-4, or HCPCA protocols. The Current Procedural Terminology, 4th Edition (CPT-4) is used most frequently, with ICD-9-CM procedure codes the second most common and HCFA Common Procedural Coding System (HCPCA) the least prevalent.

Partial or complete drug information is available for a large number of enrollees. These include both mail-order and card program prescription drug claims. These claims contain all relevant information about that particular prescription, including drug brand, dosage, number of days covered by the prescription, and cost information.

Enrollment records include enrollment information as well as general demographic information about the enrollee. These includes age, gender, region, and employment information.

1.3.2 Interesting Aspects of Claims Data

The data that you find in a claims database comes in an interesting and challenging form. For each enrollee, we observe claims happening at various points throughout

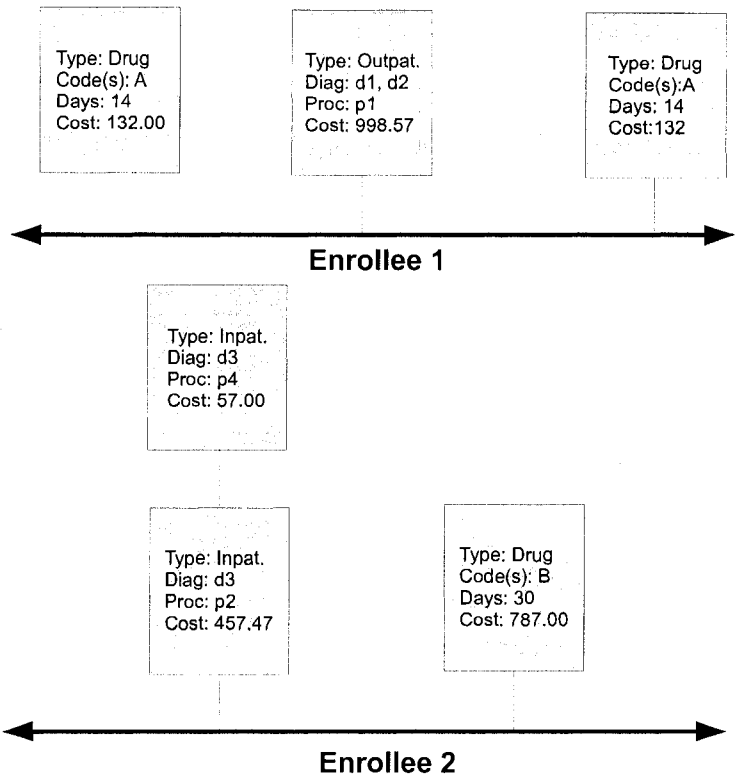


Figure 1.1: Illustration of claims data. Claims data consists of timed sequence of events of different types. Each type of claim has different variables associated with it. The number of codes present also varies from claim to claim.

their claim history. We can think of these as discrete events happening in continuous time, and refer to this type of data as a *timed sequence.*

Furthermore, the events that we see are of different types, in this case inpatient, outpatient, and prescription drug claims. Each type of claim is associated with a different set of variables. Also, the number of drug, diagnosis, and procedure codes can vary from claim to claim. All of these facts make it very hard to use standard multivariate techniques that are designed for data that is described by fixed length vectors.

The final point of interest is the timing involved in the claims. Suppose that we have two enrollee's that are affected by the same condition. At any point in time,

those enrollees could be at a different point in the progression of the condition. It may also be the case the enrollees progression will occur at different rates. For these reasons, we believe that the timing of the claims relative to one another may be an important feature of the data.

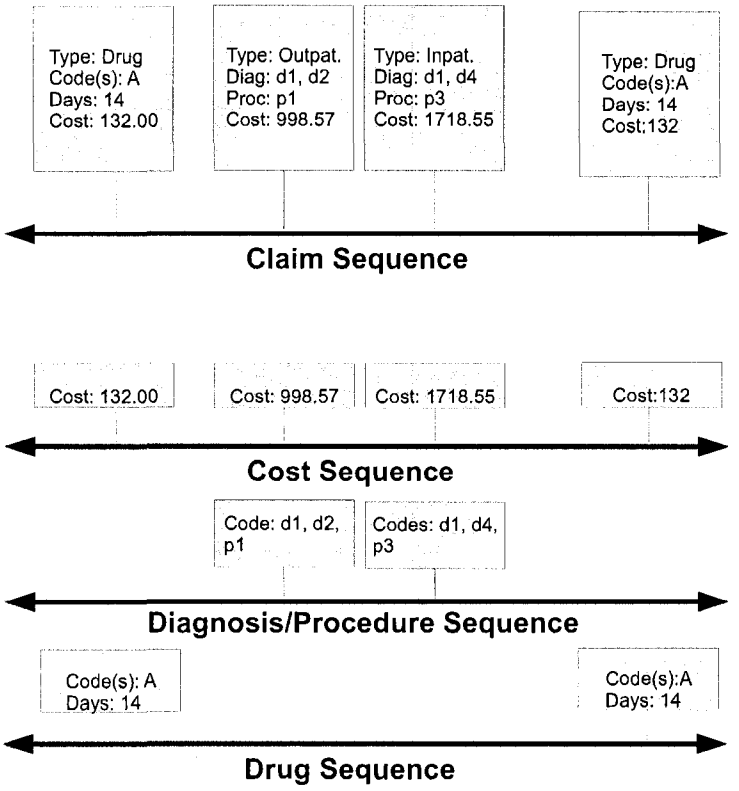


Figure 1.2: Separating claims data into three sequences. There is one for each type of variable: cost, diagnosis/procedure codes, and drug information.

Our general strategy for dealing with these data is to split the data into several timed sequences, all with the same type of information. Figure 1.2 illustrates the approach. We make a separate sequence for the cost information, the diagnosis and procedure codes, and the prescription drug information.

1.4 Dissertation Layout

The layout of the dissertation is as follows: We provide some background on classification methods, support vector machines, and kernels in Chapter 2. Chapter 3 presents two new kernels that were inspired by the diabetes problem. We address the task of predicting subjects at risk of type 2 diabetes based solely on their insurance claims in Chapter 4 and an overview of how standard association rules mining can be applied to claims data, as well as a new definition of an "interesting" rule is given in Chapter 5.

Chapter 2

BACKGROUND

In this chapter, we provide some background material on the two class classification problem, classical methods for classification using sequence data, support vector machines (SVMs), and kernels. We also summarize some kernels designed specifically for sequences.

2.1 Common Classification Methods

In this section, we will review some of the common methods for handling a classification problem and motivate our choice of kernel methods. A thorough presentation of the general theory of classification problems and most of the following methods can be found in Hastie et. al. [6].

The specific problem that we face is a two class classification problem. For each observation, we denote the (known) features as $\boldsymbol{x}$ and use the variable y to denote the class associated with the observation. The task is to find a classifier that takes the features $\boldsymbol{x}$ as input and outputs a prediction of the class $\hat{y}$. We hope to find a classifier that predicts the class of a new observation with a high rate of success.

There are many methods for building classifiers. The statistical approach is as follows. Suppose that the data for each class is generated from a probability distribution $P(\boldsymbol{x}|c)$, where c is the class. We need to define a *loss function*, $l(y, \hat{y})$, that assigns a penalty for a misclassification. A good classifier would have a small expected loss $E(l(Y, \hat{Y}))$.

Suppose that we choose the zero-one loss, which assigns a loss of 1 for a misclassification and 0 for a correct classification. Then the classifier that minimizes the expected prediction error assigns the label of the most probable class given the features $\boldsymbol{x}$, i.e. $\hat{y} = argmax_c P(C = c|\boldsymbol{x})$ [6]. This classifier is known as the *Bayes classifier*, and the associated (minimum) value of the expected prediction error is called the *Bayes risk*. Note that minimizing the zero-one loss is equivalent to maximizing the success rate. Thus, knowing the class distributions and the prior distributions for each class allow us to classify with the greatest success rate.

Many methods either estimate the Bayes classifier for some assumed model or approximate this classifier by estimating the class distributions non-parametrically. The discriminant function obtained from the well known *linear discriminant analysis* (LDA) leads to the Bayes classifier when the class distributions are normal and the two class distributions have the same covariance matrices, assuming we use the true class priors. The discriminant function from *Quadratic discriminant analysis* is related to the Bayes classifier when the class distributions are normal and both class distributions have possibly different covariance matrices. The *Naive Bayes* classifier assumes that the marginal distributions of the features $\boldsymbol{x}$ are independent and estimates the class distributions accordingly.

Suppose that the classes labeled $+1$ and -1 and the probability of a random observation coming from each class is π_{+1} and π_{-1}, respectively. The function

$$lo(\boldsymbol{x}) = \log\left(\frac{P(\boldsymbol{x}|C = +1)}{P(\boldsymbol{x}|C = -1)}\right)$$

is known as the log-odds, and it leads to the Bayes classifier when using the correct rule; namely $lo(x) > \log(\pi_{-1}/\pi_{+1})$. Another classifier based on log-odds is logistic regression, a method that assumes the log-odds is a linear function for all classes and the best model is fit using the maximum likelihood criterion.

There are many classifiers that don't rely on the Bayes theory in their development. Some methods are based on classification using distance measures. For

example, the k-nearest neighbors (KNN) classifier is one such classifier. For each point, we find the k nearest points using some distance measure and the predicted class is the most frequent class amongst these points.

There is a whole group of tree based classifiers based on building a set of classification rules. If the features are continuous, the rules are obtained by splitting the feature space into regions. A classification rule is assigned to each possible combination of regions of the feature vectors. These methods include CART [7], C4.5 [8], and Random Forests [9].

Support vector machines use large margin hyperplanes in conjunction with a kernel to generate a non-linear classification boundary. This is an example of a method that employs a trick known as "the kernel trick" (see section 2.1.3.1) to adapt and add greater flexibility to linear and simple distance based classification methods. Other methods that use the kernel trick include kernel LDA and the kernel k-nearest neighbors classifier.

2.1.1 Methods for Sequence Data

Sequence data arise in many other situations as well. These include speech and sound data, DNA and protein sequences, text documents, network traffic flows, and in our case, insurance claims data. The book by Sankoff and Kruskal [10] gives an overview of many early methods of dealing with sequences of different lengths.

In some applications, the prediction variables used in developing a classifier are vectors of a given dimension, say d, and the classification problem can be tackled using traditional multivariate analysis techniques. However, in the diabetes problem, the predictors are not vectors of a fixed size, but rather a timed sequence of claims. Therefore, standard multivariate classification methods are not applicable here.

There are two main methods of dealing with this type of data. The first involves summarizing the sequences with a fixed number of statistics and employing

Figure 2.1: Two alignments between *abbca* and *abb*.

standard multivariate methods. One example is the "bag of words" technique from text classification [11].

The second involves comparing the sequences directly through some form of alignment. Here are two examples of an alignment. Consider the strings *abbca* and *abb*. The alignments are shown in Figure 2.1. In the first example, we have aligned first a in *abbca* with the first b in *abb* and then second b in *abbca* with second b in *abb*. In the second alignment, we have the first three letters in *abbca* with the first three letters in *abb*.

Not all alignments are intuitively pleasing. In this example, the second alignment is probably a better way to compare the sequences than the first. For this reason, alignment methods develop a scoring function that measures either the distance between, or similarity of, the sequences when using a particular alignment.

To define an alignment scoring method, first define the collection of alignments that are being considered. The next step is to define a scoring function, or *alignment score.* Finally, all possible alignment scores are combined into one score, either by taking the minimum distance, maximum similarity, or taking a sum over all the scores. Examples of well-known methods that use this approach are the Needleman-Wunsch

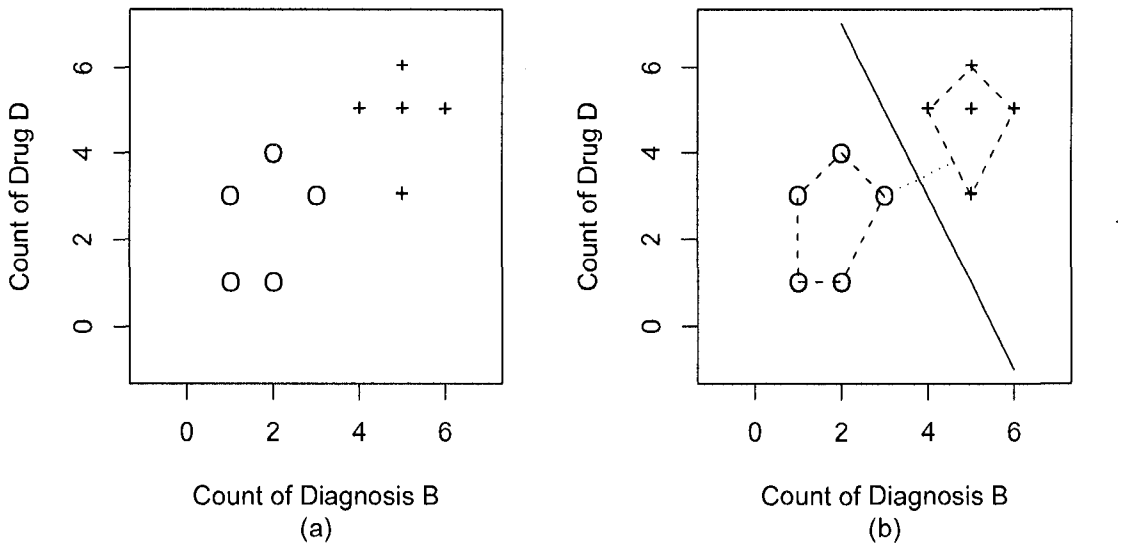


Figure 2.2: Example of a classification problem. (a) The two dimensional prediction variables are plotted in the plane. Examples of the positive and negative classes are represented with + and O respectively. (b) The optimal separating plane for this example.

score [12], the Smith-Waterman score [13], the local alignment kernel [14], and the time warping score [15]

2.1.2 Support Vector Machines

In the following sections we will provide an overview of support vector machines (SVMs) [16]. First we show how classification can be performed through optimal separating planes [17]. Then, we show how these methods are generalized through a soft-margin classifier [18].

2.1.2.1 Linear Classification with Large Margin Hyperplanes

First, we look at the problem of finding a large margin hyperplane in the case when the data are linearly separable [17]. That is, there exists a hyperplane that completely separates the two groups (see Figure 2.2).

We will use a representation for a hyperplane (a line if the data are represented in a two-dimensional Euclidean space) that will be convenient for discussions contained

in subsequent sections. Notice that a hyperplane $w_1\ x_1 + w_2\ x_2 + \ldots + w_d\ x_d + b = 0$ can be written as

$$\{\boldsymbol{x} |\ \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0\},$$

where $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ denotes the inner product of the vectors $\boldsymbol{w} = (w_1, \ldots, w_d)$ and $\boldsymbol{x} = (x_1, \ldots, x_d)$. We call the function $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ the *discriminant function* and the hyperplane $f(x) = 0$ the *decision boundary*. The side of the hyperplane that a point lies on can be determined by $\text{sign}(f(\boldsymbol{x}))$, as all point on the same side of the hyperplane will share the same sign. We will use $\hat{y} = \text{sign}(f(\boldsymbol{x}))$ as our prediction for the class of y.

Consider the example from the last section in Figure 2.2(a). Here we have a case where the data are linearly separable, that is the two groups can be completely separated by a line.

As there are lots of lines that separate the two groups, consider the question: Which line is the "optimal" line? There are many ways that we could define optimality one being the line that has the largest margin. The *margin* is the distance between the line and its closest point. Figure 2.2(b) on page 12 shows the line with largest margin. Geometrically, this line can be found by:

1. Drawing the Convex Hull around each group.
2. Finding the pair of points, one point on each hull, that are closest to each other.
3. Finding the perpendicular bisector of the line segment connecting the points in 2.

The line from 3 is the optimal separating plane, that is it has the largest margin.

The optimization is reworked in the following way: re-scale $\boldsymbol{w}$ and b such that the margin is unit length. Suppose that $\boldsymbol{x}_{pos}$ is the closest point on positive side and $\boldsymbol{x}_{neg}$ is the closest point on the negative side. The rescaling of $\boldsymbol{w}$ and b gives

$\langle \boldsymbol{w}, \boldsymbol{x}_{pos} \rangle + b = 1$ and $\langle \boldsymbol{w}, \boldsymbol{x}_{neg} \rangle + b = -1$. This implies that $\langle \boldsymbol{w}, (\boldsymbol{x}_{pos} - \boldsymbol{x}_{neg}) \rangle = 2$ and thus $\left\langle \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, (\boldsymbol{x}_{pos} - \boldsymbol{x}_{neg}) \right\rangle = \frac{2}{\|\boldsymbol{w}\|}$. This shows that a large margin is the result of a small $\boldsymbol{w}$. Thus, the optimization problem can be written

$$\begin{aligned} &\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{subject to } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 \text{ for all } i = 1, \ldots, n \end{aligned} \tag{2.1}$$

A constrained optimization problem like this can be solved using *Lagrangian Multipliers* (see [19, page 166]). The *Lagrangian* for this problem is

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) - 1), \tag{2.2}$$

where $\alpha_i \geq 0$ for all $i = 1, \ldots, n$. The Karush-Kuhn-Tucker(KKT) conditions tell us that a solution must occur at a saddle-point where L is maximized with respect to $\boldsymbol{\alpha}$ and minimized with respect to $(\boldsymbol{w}, b)$. Thus

$$\frac{\partial}{\partial b} L = 0 \text{ and } \frac{\partial}{\partial \boldsymbol{w}} L = 0$$

which implies

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.3}$$

and

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x_i}. \tag{2.4}$$

The KKT conditions also tell us that $\alpha_i(y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) - 1) = 0$ for all $i = 1, \ldots, n$. Thus, either $\alpha_i = 0$ and it doesn't matter what value $y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b)$ takes or $y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) - 1 = 0$ which indicates that $\boldsymbol{x}_i$ is on the margin. Points on the margin are called *support vectors.*

The *dual* optimization problem is constructed by substituting 2.3 and 2.4 into 2.2 which gives a new form for the optimization problem that no longer depends on $\boldsymbol{w}$

and b. The result is

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{2.5}$$

$$\text{subject to } \alpha_i \geq 0 \text{ for all } i = 1, \ldots, n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0.$$

This is a quadratic optimization problem for which there are many known algorithms.

2.1.2.2 Soft Margin SVMs

Thus far we have assumed that the data is linearly separable. We can use a so-called *soft margin* method [18] to deal with linearly non-separable cases by introducing a penalty for points on the wrong side of the decision boundary. Rewrite (2.2) as

$$\min_{\boldsymbol{w}\in\mathbb{R}^d, \boldsymbol{\xi}\in\mathbb{R}^d} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$$

$$\text{subject to } \begin{cases} y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i \text{ and} \\ \xi_i \geq 0 \text{ for all } i = 1, \ldots, n \end{cases}$$

where C is a given cost parameter associated with the slack variables $\boldsymbol{\xi}$. Applying Lagrangian multipliers and the KKT conditions leads to a similar dual problem as (2.6)

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$

$$\text{subject to } 0 \leq \alpha_i \leq \frac{C}{n} \text{ for all } i = 1, \ldots, n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0.$$

Once again, the solution space for this optimization problem depends only on the number of observations and not the dimension of $\boldsymbol{x}$. The parameter C is a tuning parameter that can be chosen using cross-validation.

Now it is time to make two very important points about SVMs. First, notice that we are optimizing on α and need to find n values, where n is the number of observations in the data set. Thus, the size of the search space for optimizing an SVM only depends on the number of observations, not the dimension of $\boldsymbol{x}$. This makes SVM a great method for high-dimensional data.

Second, the only information that we need about the vectors $\boldsymbol{x}_i$ are the inner products $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. This leads to the *kernel trick*. Suppose we have a function k such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ for all $i, j = 1, \ldots, n$. Then we need not store $\boldsymbol{x}_i$ but only be able to evaluate k. As we will see in the next section, this leads to a nice method for developing non-linear classification methods, and also allows us to apply the machinery to non-vector data.

2.1.3 Kernels

In this section, we review how the capabilities of large margin classifiers can be greatly expanded through the use of kernels. We illustrate how kernels can be used to make non-linear decision boundaries. Then, the properties of kernels are discussed.

2.1.3.1 Linear Classifiers and the Kernel Trick

Consider the example data shown in Figure 2.3(a) where there are again two groups of points. It is clear that there is no line that is able to successfully separate the two groups of points, that is, the data are not linearly separable. Suppose we apply the map ϕ that takes (x_1, x_2) to (x_1^2, x_2^2). Figure 2.3(b) shows the points in the new space, which are now linearly separable. The line that is shown is the result of applying a SVM classifier. The points on this line can be translated back to the original space, as shown in Figure 2.3(c). Notice that the net result is a non-linear classifier.

Recall that the only values needed from the space (x_1^2, x_2^2) are the inner products, which are given by the function $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle = x_1^2 x_2^2$. We call such a function a *kernel*. In practice, we never actually do the transformation from (x_1, x_2) to (x_1^2, x_2^2). Instead we simply evaluate $k(x_1, x_2)$.

This example illustrates the *kernel trick*. We imagine a mapping ϕ from the data $\boldsymbol{u}$ to some vector $\boldsymbol{x}$ in a Hilbert Space $\mathcal{H}$. For technical reasons, we need the mapping to land in a Hilbert Space, which is a vector space with an inner product in which

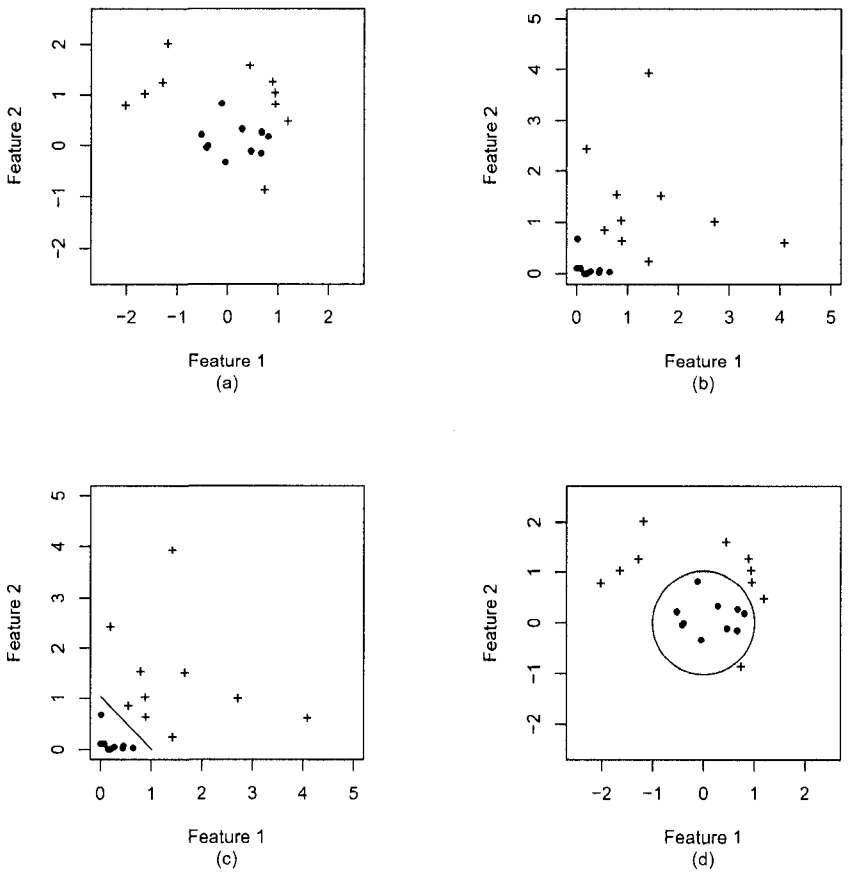


Figure 2.3: A non-linear example. (a) Two Groups • and +. (b) Plot of Square of Each Feature. (c) Optimal Separating Plane in the Square-Square Scale. (d) Optimal Separating Plane Translated In Original Scale.

every Cauchy sequence has a limit that belongs to the space. Notice that the original data $\boldsymbol{u}$ need not be a vector. This allows Kernel Methods to be applied to data that can be represented by a graph or some other non-vector format.

The kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ is constructed and used to train a linear classifier in the new space $\mathcal{H}$. The section above illustrated how the kernel trick could be applied to a SVM. There are many other linear methods that can take advantage of the kernel trick, including Linear Discriminant Analysis, Principal Component Analysis, and Ridge Regression.

Clearly, a kernel exists for any mapping to a Hilbert Space. It is defined by $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2) \rangle$. The next section answers the question: Given a function

k, is there a mapping ϕ from the data to some Hilbert Space $\mathcal{H}$ such that $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2) \rangle$? If we can answer this question, we can simply use the kernel knowing that there is a corresponding mapping and Hilbert space without ever needing to construct this mapping. This will allow us to, for instance, consider mappings into infinite dimensional spaces.

2.1.3.2 Kernel Properties and Popular Kernels

What properties must a function k have for there to exist a mapping ϕ into a Hilbert Space $\mathcal{H}$ such that $k(\boldsymbol{u}_i, \boldsymbol{u}_j) = \langle \phi(\boldsymbol{u}_i), \phi(\boldsymbol{u}_j) \rangle$? For a given set of observations $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$, we construct a matrix K whose i, jth component is $k(\boldsymbol{u}_i, \boldsymbol{u}_j)$, which is called the *Gram matrix.*

The necessary and sufficient conditions for k to be the kernel for some mapping ϕ into some Hilbert space $\mathcal{H}$ is: for every set of observations $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ the Gram matrix K is positive definite [19]. Recall that a matrix is positive definite if it has all positive eigenvalues.

Here are some examples of popular kernels. These kernels are all defined for data that is in vector form. The most basic kernel is the Euclidean inner product and is called a linear kernel:

$$k_l(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j . \rangle \tag{2.6}$$

Another popular kernel is the polynomial kernel of degree d. It corresponds to the feature space of monomial terms of the observations.

$$k_p(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + a)^d, \tag{2.7}$$

Where a and d are parameters whose value needs to be specified. We refer to such model parameters as *tuning parameters.* Typically, cross-validation is used to choose tuning parameters.

Our last example of a common kernel is the Gaussian Kernel. It too has a tuning parameter, in this case σ. Notice the resemblance to the Gaussian, or normal, pdf.

This gives us some intuition into the workings of this kernel.

$$k_g(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \tag{2.8}$$

The Gaussian kernel is very sensitive to normalization and the choice of σ. We need for values of the exponent to generally be between -3 and 3 for this kernel to perform satisfactorily. The Gaussian kernel is an example of a kernel that represents a mapping into an infinite dimensional Hilbert space.

The Gaussian kernel is an example of a *universal kernel* [20], a continuous kernel that is dense in the space of all continuous functions. To see why this is important, recall the set up for a two-class classification problem. Suppose that $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is a training set generated from some probability distribution P, where $x \in \mathcal{X}$ are the observations and $y \in \{-1, +1\}$ are the class labels. A classifier $\mathcal{C}$ uses a set of training examples T to build a decision function $f_{\mathcal{C},T} : \mathcal{X} \to \{-1, +1\}$ that is used to classify new observations with unknown labels. If we are using the zero-one loss function, the expected loss for $f_{\mathcal{C},T}$ is given by $\mathcal{R}(f_{\mathcal{C},T}) = P(\{(x, y)|f_{\mathcal{C},T}(x) \neq y\})$.

Recall that the Bayes classifier is the classifier that minimizes the zero-one loss function, with this minimum value referred to as the Bayes risk, which we denote $\mathcal{R}_{Bayes}$. A classifier is called a *consistent* if

$$\lim_{|T|\to\infty} \mathcal{R}(f_{\mathcal{C},T}) = \mathcal{R}_{Bayes}$$

holds in probability. A classifier that is consistent for all distributions is called *universally consistent.*

Steinwart [20] was able to show that the SVM classifier is universally consistent provide we use a universal kernel. Thus, the SVM classifier using a Gaussian kernel is an asymptotically optimal classifier with respect to the zero-one loss.

A important tool frequently used when using kernel methods is that of kernel construction. One method of constructing a kernel is to define the associated feature

map, $\phi(\boldsymbol{x})$. The kernels value when evaluated on $\boldsymbol{x}$ and $\boldsymbol{x}'$ is $\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$. There are other properties of kernels that are useful in kernel construction. Suppose that $a > 0$ is a constant and K_1 and K_2 are kernels on $\mathcal{X} \times \mathcal{X}$. Then, in each of the following cases, K is also a kernel on $\mathcal{X} \times \mathcal{X}$ [21].

1. $K(x, y) = a * K_1(x, y)$
2. $K(x, y) = a + K_1(x, y)$
3. $K(x, y) = K_1(x, y) + K_2(x, y)$
4. $K(x, y) = K_1(x, y) * K_2(x, y)$

Consequently, these operations can be used to construct new kernels. We will illustrate this fact in the next section.

2.1.3.3 Kernel Construction - An Example

Two of the useful qualities of kernel methods are the ability to build kernels for objects that are not fixed length vectors and the ability to combine kernels using the closure properties given in the last section. We will illustrate these points by constructing an example that will allow us to compare triangles and simple convex quadrilaterals [1].

We start by building a kernel to compare two triangles. We use the following notation for a triangle. The angles of the triangle are denoted γ, β, and α, with γ taking the value of the largest angle and β the value of the second largest angle. The length of the sides are denoted by a, b, and c, where c is the longest side and a is the shortest.

---

[1] A simple quadrilateral is a four-sided polygon that is not self-interesting. Pick any two points from the edges of the quadrilateral. We say a quadrilateral is convex if the line connecting these points is guaranteed to be contained inside the quadrilateral.

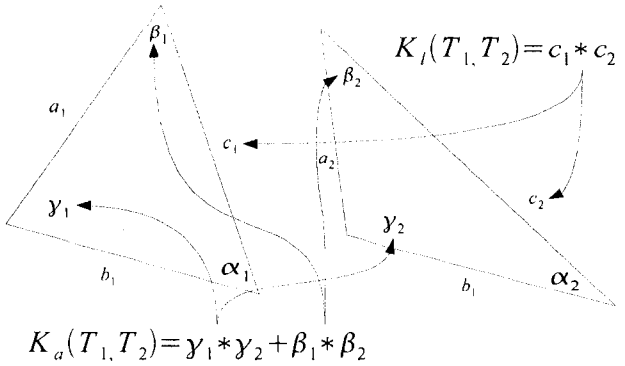
One way to construct a kernel is to build a fixed number of features for the object. It is a well known fact that a triangle is completely determined when provided three pieces of information, say two of the angles and the length of a side. We will choose to use the two largest angles, γ and β, and the length of the longest side c to represent a triangle.

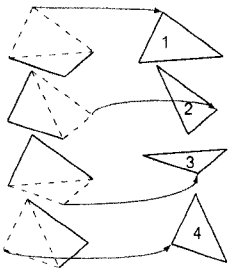
When constructing kernels, it is useful to compare "like things". Thus, we start by defining a kernel for comparing the angles of two triangles. We do this by constructing the vector $(\gamma, \beta)^T$ for each triangle and using the inner product as our kernel The second kernel is constructed to compare the lengths of the sides of the triangle. It is the inner product of two one-dimensional vectors, each containing the length of the longest side of the respective triangle. Figure 2.4(a) illustrates both of these kernels.

To combine these kernels, we can either add the kernels or multiply them. Suppose that we have two triangles T and T'. Adding kernels is related to the "or" operation, in the sense that the resulting kernel will be relatively large if T or T' is large. Multiplying the kernels, on the other hand, is like an "and" operation, in that resulting kernel will be relatively large if T and T' are large. In this example we will combine the kernels using addition.

Note that these two properties of a triangle, angles and length, might be of very different sizes or importance. To allow the user to scale the values, assign a relative importance to each, or simply tune the performance of the kernel, we introduce a tuning parameter $0 \leq \lambda \leq 1$ and add the kernels for angle and length using a weighted sum. Figure 2.4(a) shows the final form of the triangle kernel, $K_{\triangle}^{(\lambda)}$.

Now suppose that we want to construct a kernel that can be used to compare any combination of triangles and quadrilaterals. One useful way of compare items that are of different dimensionality is to break each item into "parts" and compare the parts. We already know how to compare triangles, so let's convert a quadrilateral into a number of triangles and do comparisons using our triangle kernel. Each quadrilateral





$$K_{\triangle}^{(\lambda)}(T_1, T_2) = \lambda * K_a(T_1, T_2) + (1 - \lambda) K_l(T_1, T_2) \qquad Q = \{T_1, T_2, T_3, T_4\}$$

(a) (b)

Figure 2.4: A kernel for comparing two triangles. (a) K_a compares the angles of the triangles by adding the inner products of the largest angles and the second largest angles, respectively. K_l compares the length of the longest side of each triangle. The kernel for comparing triangles, $K_{\triangle}^{(\lambda)}$, is a weighted sum of K_a and K_l. We introduce the tuning parameter $0 \leq \lambda \leq 1$ to allow the user to specify the relative importance of angles and lengths or tune for better performance. (b) We think of a quadrilateral as four triangles, each consisting of one of the interior angles, the two sides adjacent to said angle, and the appropriate diagonal.

is broken into into four triangles, each consisting of one of the interior angles, the two sides adjacent to said angle, and the appropriate diagonal. That is, we represent a quadrilateral as a set of triangles, i.e. $Q = \{T_1, T_2, T_3, T_4\}$. This process is illustrated in Figure 2.4(b).

We are now ready to build a kernel that can compare any two objects O and O', each being either a triangle or a quadrilateral. Let $\mathcal{T}$ be the space of all regular convex triangles, $\mathcal{Q}$ be the space of all regular convex quadrilaterals, and $\mathcal{O} = \mathcal{T} \cup \mathcal{Q}$. First, we define some sets that unify our types of objects. Let

$$S_O = \begin{cases} \{O\} & O \in \mathcal{T} \\ Q & O \in \mathcal{Q} \end{cases}.$$

The kernel used for comparing $O, O' \in \mathcal{O}$ is given by

$$K_O^{(\lambda)}(O, O') = \sum_{T \in S_O, T' \in S_{O'}} K_\triangle^{(\lambda)}(T, T').$$

Figure 2.5 illustrates the computation of this kernel for various combinations of objects.

In summary, we first constructed a kernel for comparing a relatively simple object, a triangle, using the properties of kernels. We then conceptualized a more complex object, a quadrilateral, as being composed of simpler parts (triangles). This allowed us to compare different types of objects(triangles and quadrilaterals), provided that they could be broken into the correct type of simpler parts(triangles). This illustrates a common method for kernel construction that was formalized by Haussler [22], known as a *convolution kernel.* The local alignment kernel, which is discussed in the next section, is an example of such a kernel.

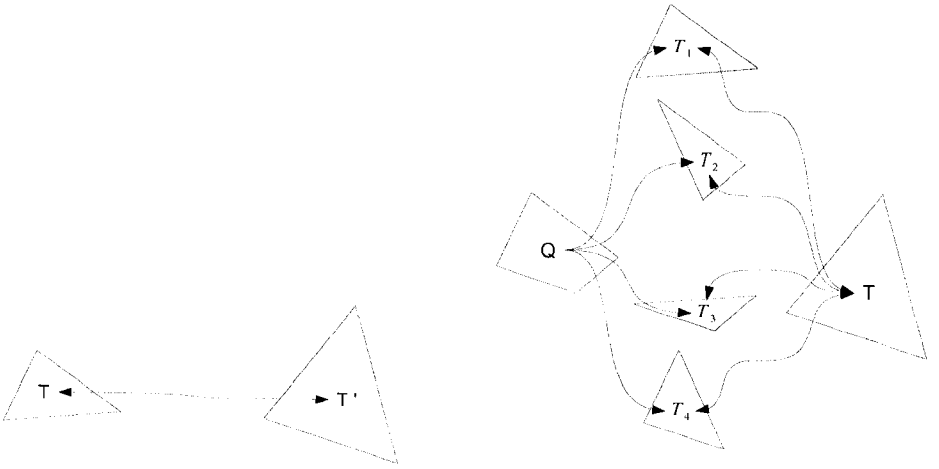
2.1.4 Methods for Sequences Data

In this section, we will look at various existing methods for dealing with data that are sequences of different lengths.

2.1.4.1 Hidden Markov Models and the Fisher Kernel

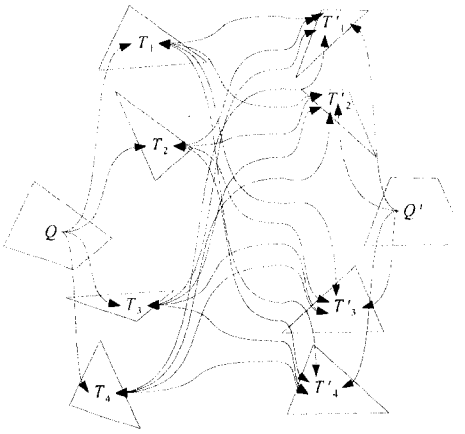
First, we give an overview of hidden Markov models (HMM) [23]. The general idea behind an HMM is that the data is being generated from a probabilistic model that is always in one of a given set of states. The states are hidden from the observer, but we are able to observe an emitted signal that depends on the current state.

When we first observe the system, we assume that the process is in some random state. Suppose that we have a finite number of states $\{m_1, m_2, \ldots, m_N\} = \mathcal{M}$. The starting probability is denoted $p_{m_1}, \ldots, p_{m_N}$, where p_{m_i} is the probability that the process starts in state i. The simplest implementation assumes that a random state change occurs at each discrete time interval. We denote the transition probabilities



$K_O^{(\lambda)}(T, T') = K_\triangle^{(\lambda)}(T, T')$ (A)

$K_O^{(\lambda)}(T, Q) = \sum_{j=1}^{4} K_\triangle^{(\lambda)}(T, T_j)$ (B)



$K_O^{(\lambda)}(Q, Q') = \sum_{i=1}^{4} \sum_{j=1}^{4} K_\triangle^{(\lambda)}(T_i, T'_j)$ (C)

Figure 2.5: A kernel for comparing triangles and/or quadrilaterals. (A) When comparing two triangles, we simply apply $K_\triangle^{(\lambda)}$. (B) When comparing a quadrilateral, Q to a triangle T, we compare the triangle T to each of the sub-triangles of Q using $K_\triangle^{(\lambda)}$ and sum the results.(C) When comparing the quadrilaterals Q and Q', we compare each sub-triangle of Q to each of the sub-triangles of Q' using $K_\triangle^{(\lambda)}$ and sum the results.

$P_t(m_i|m_j)$ for all states $m_i, m_j \in \mathcal{M}$, where $P_t(m_i|m_j)$ is the probability of a transition from state i to state j.

At each time step, the system will emit a random signal. The distribution of the signals depends only on the current state. Suppose that the set $\mathcal{A}$ contains all of the possible signals. We define the emission probability $P_e(a|m_i)$ for all $m_i \in \mathcal{M}$ and $a \in \mathcal{A}$, where $P_e(a|m_i)$ is the chance of state i emitting the signal a.

2.1.4.2 The Fisher Kernel

The Fisher kernel[24] is a method that enables us to use a probabilistic generating function with kernel methods. Suppose that the sequence $\boldsymbol{x}$ is generated by some parametric model, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ are the k parameters. The *likelihood function* $l(\boldsymbol{\theta}|\boldsymbol{x})$ is the result of plugging the observed values of $\boldsymbol{x}$ into the pmf/pdf and thinking of it as a function of the parameters $\boldsymbol{\theta}$. The *score vector* is the vector

$$s(\boldsymbol{\theta}|\boldsymbol{x}) = \left(\frac{\partial l(\boldsymbol{\theta}|\boldsymbol{x})}{\partial \theta_1}, \ldots, \frac{\partial l(\boldsymbol{\theta}|\boldsymbol{x})}{\partial \theta_p} \right)^T,$$

where we make the appropriate assumptions about l such that the derivatives exist [2].

The score vector was used in a classic statistical hypothesis test, the Rao score test [25], which decides whether the data comes from one of two specified models. Consider the following hypotheses

$$H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$$

$$H_a : \boldsymbol{\theta} = \boldsymbol{\theta}_a$$

where H_0 and H_a are the null and alternate hypothesis, respectively. The parameters $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_a$ are the parameters for the specified models, called the *null model* and *alternate model*, respectively. Before showing the relationship between the score vector,

---

[2] Taking $s(\boldsymbol{\theta}|\boldsymbol{x})/\sqrt{n}$ gives the score vector as originally defined by Fisher. We will stick to the definition presented in [24].

the Fisher kernel, and the Rao score test, we must define the Fisher information. The Fisher information is given by

$$I(\boldsymbol{\theta}) = -E\left[\frac{\partial^2 \ln l(\boldsymbol{\theta}|\boldsymbol{X})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}\middle|\,\boldsymbol{\theta}\right]$$

Then the Rao score statistic is

$$s(\boldsymbol{\theta}_0|\boldsymbol{x})^T I(\boldsymbol{\theta}_0)^{-1} s(\boldsymbol{\theta}_0|\boldsymbol{x}),$$

which can be shown to have an asymptotic chi-square distribution with k degrees of freedom [26].

One definition for the Fisher kernel is

$$s(\boldsymbol{\theta}_0|\boldsymbol{x})^T I(\boldsymbol{\theta}_0)^{-1} s(\boldsymbol{\theta}_0|\boldsymbol{y}),$$

which seems to be inspired by the Rao score statistic. More commonly, the kernel is defined as

$$s(\boldsymbol{\theta}_0|\boldsymbol{x})^T s(\boldsymbol{\theta}_0|\boldsymbol{x}),$$

which saves us from having to compute the Fisher information matrix. Refer to Appendix A for the explicit features related to a Fisher kernel based on an HMM.

2.1.4.3 The Local Alignment Kernel

The Local Alignment(LA) kernel [14] was developed to adapt the popular Smith Waterman algorithm for local sequence alignment into a kernel. To properly define the LA kernel, we first need to define an alignment for strings.

Classically, one method of comparing sequences of different lengths used a "similarity" score based on a system of aligning sequences and scoring said alignments. These scores can be classified as *global*, they measure the complete alignment of the sequences from start to finish; or *local*, measuring only the "similarity" between parts of each sequence. The Needleman-Wunsch algorithm gives one such score that is considered a global measurement for sequences. Another global comparison, this time for

Figure 2.6: Two alignments between *abbca* and *abb* including gaps. To help in scoring the alignment, we introduce a gap in one sequence, denoted with an "-", whenever the alignment skips a letter in the other sequence. Both the Needleman-Wunsch and Smith-Waterman scores penalize gaps.

time series with continuous measurements, is Dynamic Time Warping (DTW) [15]. A popular local scoring method for sequences is the Smith-Waterman score. All of these methods find the alignment with the minimum distance, and report this value as the score.

Let's define an alignment. Let $\mathcal{A}$ be the *alphabet* of characters. The string $\boldsymbol{x}$ of length n is the concatenation of n letters from $\mathcal{A}$. We let $\mathcal{X}$ be the class of all strings.

An *alignment* of p positions between $\boldsymbol{x}, \boldsymbol{y}$ is a pair of p-tuples:

$$\pi = \Big(\big(\pi_1(1), \ldots, \pi_1(p)\big), \big(\pi_2(1), \ldots, \pi_2(p)\big)\Big) \in \mathbb{N}^{2p}$$

that satisfies

$$\begin{aligned} 1 \leq \pi_1(1) \leq \ldots \leq \pi_1(p) \leq |\boldsymbol{x}| \\ 1 \leq \pi_2(1) \leq \ldots \leq \pi_2(p) \leq |\boldsymbol{y}|, \end{aligned}$$

And let $\Pi(|\boldsymbol{x}|, |\boldsymbol{y}|)$ be the collection of all such alignments. These are the alignments that are used by both the Needleman-Wunsch and Smith-Waterman algorithms.

We have shown the alignments given in the example section 2.1.1 again in Figure 2.6, this time with gaps. To aid in understanding the computation of the score,

we introduce a gap in one sequence, denoted with an "-", whenever the alignment skips a letter in the other sequence. Note that the first alignment is given by $((1,2),(3,3))$. It may not be the best way of aligning these strings. The second alignment, $((1,1),(2,2),(3,3))$, will align the substrings *abb* that are found at the beginning of each string, and may be a more appealing way of aligning these particular strings. In that light, we need to develop a method for measuring the quality of each alignment for a specific pair of strings. The Smith-Waterman score will rate the similarity between the two strings aligned in a specific way. In particular, it is the local alignment score used by the Smith-Waterman algorithm, which ignores gaps before the first aligned position or after the last aligned postion. Suppose that we have a similarity function S for aligned letters and a gap function g for which assign a penalty to gaps according to the length of the gap. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two strings. The local alignment score for alignment π is given by

$$s_\pi(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{|\pi|} S(\boldsymbol{x}_{\pi_1(i)},\boldsymbol{y}_{\pi_2(i)}) - \sum_{i=1}^{|\pi|-1} [g(\pi_1(i+1)-\pi_1(i)) + g(\pi_2(i+1)-\pi_2(i))].$$

Notice that the score is the sum of the similarities between aligned symbols minus the penalty we accrued for gaps.

The Smith-Waterman score is defined as

$$s_{SW}(\boldsymbol{x},\boldsymbol{y}) = \max_\pi s_\pi(\boldsymbol{x},\boldsymbol{y}),$$

that is, it is the score for the alignment that results in the highest similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$ [3]. One might ask "Is this function a kernel?". While there is at least one trivial combination of S and g that make this a kernel, other combinations have not produced a positive definite function [14].

---

[3]The Needleman-Wunsch global score is almost identical, the difference being that the global score penalizes gaps before the first alignment and after the second alignment, where as the Smith-Waterman score does not.

The LA kernel is defined in the following way:

$$k_{LA}^{(\beta)}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\pi \in \Pi(|\boldsymbol{x}|, |\boldsymbol{y}|)} \exp(\beta s_\pi(\boldsymbol{x}, \boldsymbol{y})).$$

Instead of employing the maximum of the scores it uses a "soft max", the sum of the exponentiated score for each alignment. This score has the property that $\lim_{\beta \to \infty} \frac{1}{\beta} \log k_{LA}^{(\beta)}(\boldsymbol{x}, \boldsymbol{y}) = s_{SW}(\boldsymbol{x}, \boldsymbol{y})$, while retaining information about all the alignment scores.

This kernel is designed for aligning a string of symbols. The diabetes data set consists of a timed sequences of codes that happened at specific points in time. This leads to difficulties adapting these data to the LA kernel. In particular, there is no obvious way to incorporate the information about the timing of the claims in a string alignment setting.

Having said that, the process by which the LA kernel converts the Smith Waterman score into a kernel inspired us to attempt to do the same for an alignment method that does work for timed sequences: time warping.

Chapter 3

TIME WARPING KERNELS

3.1 Introduction

Two methods based on alignments were presented in the last chapter, namely the Smith-Waterman algorithm and the local alignment kernel. Both of these methods are designed for comparing un-timed sequences. Another alignment method, called *time warping* [27], is designed instead for comparing timed sequences. This method was made popular in the area of spoken word recognition. The general idea is to align two time series by "warping" time. The quality of a given alignment is assessed by an alignment score. The best score out of all possible alignments is used as the final measure of distance between the two sequences.

Since claims data is also a timed sequence, this alignment method is of great interest. Consider two patients with the same disease. When comparing their claims history, it must be noted that (a) their diseases may not be progressing at different rates, (b) the patients may be at different points in the disease's progression at a given point in time, and (c) not all of the claims in a patient's history need be related to the disease. Because of (a), we believe using a method that incorporates time warping is justified. Due to (b) and (c) the use of a local alignment method seems to be warrented.

To fully understand the need for our work, it is necessary to understand that the original definition of time warping included time coefficients used to weigh the severity of the time warping needed for a particular alignment (more details are given

in Section 3.2). In the last few years, there have been attempts to adapt dynamic time warping (DTW) for use with SVMs (see [28], [29], [30], and [31]). Of the attempts to adapt DTW to the SVM framework, only Shimodaira et. al. [28] have included these time coefficients. Only Cuturi et. al. [31] proved that their kernel is positive definite. All of these methods are global alignment methods. It is our contention that building a kernel that includes these time coefficients will lead to a better similarity score when the waiting times between observations are important.

In this chapter, we extend the work of Cuturi et. al. [31] in the following ways: We introduce both local and global DTW kernels that include information about the time intervals between measurements. First, we review the definitions of classical time warping in section 3.2. In section 3.3.1, we motivate and define the global DTW (GTW) kernel. Section 3.3.2 introduces the local DTW (LTW) kernel. An efficient algorithm for each of these kernels is given in Section 3.3.3. We test our hypothesis that the new kernels are superior at discriminating between observations when the time between measurements is important in Section 3.3.4. We illustrate the performance of the LTW kernel when used in the diabetes problem in Section 4.3.3.6.

3.2 Classical time warping

Suppose that the vector $\boldsymbol{x} = (x_1, \ldots, x_m)$ contains a sequence of measurements from a space with a distance measure w. These measurements are taken at real-valued times $t_x(1) = 0, \ldots, t_x(m)$, with $t_x(i) < t_x(j)$ when $i < j$. Let $\mathcal{X}_m$ be the class of all such vectors of length m and $\mathcal{X} = \bigcup_{k=1}^{\infty} \mathcal{X}_m$.

The classical definition of a discrete (global) time warping (see Kruskal and Liberman [15]) is as follows: Let $k = 1, \ldots, K$ index the time warping alignments $(i(k), j(k))$. To represent a valid (global) time warping, $i(k)$ must be a function outputting natural numbers such that $i(k) \leq i(k+1) \leq i(k) + 1$, $i(1) = 1$, and $i(K) = m$. Similarly, $j(k)$ must be a non-decreasing function of the natural numbers

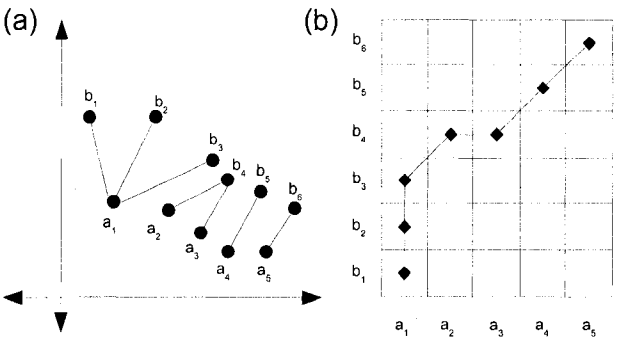


Figure 3.1: Illustration of a time warping alignment. (a) The time warping $(i(k), j(k))$ aligns the points of $\boldsymbol{a}$ to $\boldsymbol{b}$. Here a_1 is aligned with b_1, b_2, and b_3; a_2 and a_3 are aligned with b_4; and so forth. (b) A matrix representation of a time warping.

such that $j(k) \leq j(k+1) \leq j(k)+1$, $j(1) = 1$, and $j(K) = n$. We write $\boldsymbol{\pi} = ((i(1), j(1)), \ldots, (i(K), j(K))$.

In the example illustrated in Figure 3.1, we see that a_1 is grouped/aligned with b_1, b_2 and b_3. The term *time warping* comes from the fact that we are imagining expanding/compressing[4] time in an attempt to match the sequences. In this case, the time that passed between measurements b_1 and b_3 is compressed so that b_1, b_2, and b_3 happen at the same instant as a_1.

It is common to talk in terms of the allowable changes from one pair to the next. The conditions given above allow for either adding one to $i(k-1)$, adding one to $j(k-1)$, or adding one to both $i(k-1)$ and $j(k-1)$. We refer to these as $(1,0)$, $(0,1)$, and $(1,1)$ moves respectively.

An equivalent expression of the time warping can be constructed as follows: we write $\Delta i(k) = i(k) - i(k-1)$ and $\Delta j(k) = j(k) - j(k-1)$ for $k = 2, \ldots, K$. As the time warping is forced to start at $i(1) = j(1) = 1$, it does not matter how $\Delta i(1)$ is defined. For convenience, we define $\Delta i(1) = \Delta j(1) = 1$. Finally, we use the notation

[4]This is a slightly different take on alignments than that taken by methods designed for strings of symbols. For example, the idea behind the Needleman-Wunch algorithm is to insert and delete symbols to match two strings.

$\Delta\boldsymbol{\pi} = ((\Delta i(1), \Delta j(1)), \ldots, (\Delta i(K), \Delta j(K))$ to refer to this alternate representation of $\boldsymbol{\pi}$. This representation explicitly gives which move was employed at each step: $(1,0)$, $(0,1)$, or $(1,1)$.

Suppose that we define the time coefficient function T as

$$T_{\boldsymbol{\pi}}(t, t', k) = \frac{t(i(k)) - t(i(k-1)) + t'(j(k)) - t'(j(k-1))}{2}. \tag{3.1}$$

Kruskal and Liberman [15] give the distance between $\boldsymbol{x} \in \mathcal{X}_m$ and $\boldsymbol{y} \in \mathcal{X}_n$ according to the time warping $\boldsymbol{\pi}$ as

$$d_{\boldsymbol{\pi}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k=1}^{K} w(a_{i(k)}, b_{j(k)}) T_{\boldsymbol{\pi}}(t_x, t_y, k),$$

and the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ as

$$D(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{\pi}} d_{\boldsymbol{\pi}}(\boldsymbol{x}, \boldsymbol{y}).$$

The Kruskal and Liberman [15] definitions were motivated by a desire to make the scores symmetric in time. We are not concerned with this property and make no claims about the time symmetry of our method.

3.3 The Problem with Classical Time Warping

First, we give our intuition of how a time warping distance should behave. Suppose that while aligning two timed sequences, we align two measurements x_1 and x_2 that were taken at times t_1 and t_2 respectively. A time warping distance between the sequences should increase if we (1) change x_1 and x_2 so that they are further apart or (2) change t_1 and t_2 so that they are further apart. Equivalently, we could consider the similarity (i.e. inner product) between the values. In this case, a time warping "similarity" should increase when if we (1) make x_1 and x_2 closer in value or (2) make t_1 and t_2 closer in time.

The classical definition of time warping is the product of two distances, one for time and one for observations. The problem is that the product of two distances will

be small when either of the distances is small. A solution is to define the similarity between two observations as the product of two similarities. The product of similarities between two observations will only be large if both of the similarities are large, which is more desirable.

3.3.1 The Proposed Global Time Warping Kernel

In this section, we will construct a kernel that is inspired by time warping. We do this by constructing a similarity between two time series based on time warping. We then show that this is a kernel. This work is an extension of the proof by Cuturi et. al. [31] in that it allows for variable lengths of time between observations.

3.3.1.1 Kernels for Measurement/Time Pairs

Let (x_1, t_1) and (x_2, t_2) be pairs such that x_i is the measurement that was taken and t_i is the time since the last measurement. Suppose that S is a kernel for the measurements x and T is a kernel for the time values t. We define two squared distances between measurement-time pairs based on these kernels

$$d_1((x_1, t_1), (x_2, t_2)) = S(x_1, x_1) - 2S(x_1, x_2) + S(x_2, x_2) \tag{3.2}$$

and

$$d_2((x_1, t_1), (x_2, t_2)) = T(t_1, t_1)S(x_1, x_1) - 2T(t_1, t_2)S(x_1, x_2) + T(t_2, t_2)S(x_2, x_2). \tag{3.3}$$

Recall that the product of two kernels is also a kernel. Both of these distance functions are conditionally positive definite by construction (for both of these facts, see Schölkopf and Smola [19]). These squared distances are converted to kernels using a kernel similar to the Gaussian kernel

$$\chi_1^{\beta,\sigma}((x_1, t_1), (x_2, t_2)) = e^{\frac{-d_1((x_1,t_1),(x_2,t_2)) - \ln(\beta)}{\sigma}} \tag{3.4}$$

and

$$\chi_2^{\beta,\sigma}((x_1, t_1), (x_2, t_2)) = e^{\frac{-d_2((x_1,t_1),(x_2,t_2)) - \ln(\beta)}{\sigma}}. \tag{3.5}$$

We wish to show that $\chi_1^{\beta,\sigma}$ and $\chi_2^{\beta,\sigma}$ are kernels. First, it is easy to show that if $k(x,y)$ is conditionally positive definite, then so is $k(x,y)+c$ for any real constant c. It can also be shown that $\exp(-k(x,y))$ is a kernel if and only if $k(x,y)$ is conditionally positive definite [19]. Thus, $\chi_1^{\beta,\sigma}$ and $\chi_2^{\beta,\sigma}$ are positive definite since $(d_1((x_1,t_1),(x_2,t_2))+\ln(\beta))/\sigma$ is conditionally positive definite.

We are using two parameters to keep the size of final time warping kernel values in check. The parameter $\sigma > 0$ plays the role of scaling the distances to an appropriate level. The parameter $\beta > 1$ will ensure the convergence of the time warping kernel by forcing $\chi_1^{\beta,\sigma} < 1$ for all inputs and has the added benefit of controlling the diagonal dominance[5] of the kernel, which was a problem encountered by Cuturi et. al.

3.3.1.2 Similarity for a Single Time Warping

Let $\boldsymbol{\pi} \in \Pi_G(n,m)$ with $\boldsymbol{\pi} = ((i(1),j(1)),\ldots,(i(K),j(K))$. Suppose that $x_1,\ldots,x_n$ are measurements taken at times $t_1,\ldots,t_n$ with $t_1 = 1$ and $x'_1,\ldots,x'_m$ are measurements taken at times $t'_1,\ldots,t'_m$ with $t'_1 = 1$.

We define the vectors

$$\boldsymbol{v}(\boldsymbol{x},\boldsymbol{t}) = \big((x_{i(1)},1),(x_{i(2)},t_{i(2)}-t_{i(1)}+1),\ldots,(t_{i(K)}-t_{i(K-1)}+1)\big) \tag{3.6}$$

and

$$\boldsymbol{v}'(\boldsymbol{x}',\boldsymbol{t}') = \big((x'_{j(1)},1),(x'_{j(2)},t'_{j(2)}-t'_{j(1)}+1),\ldots,(t'_{j(K)}-t'_{j(K-1)}+1)\big) \tag{3.7}$$

Where there is no confusion, we refer to each of these simply as $\boldsymbol{v}$ and $\boldsymbol{v}'$; and use $\boldsymbol{v}(k)$ and $\boldsymbol{v}'(k)$ as shorthand for $\boldsymbol{v}(\boldsymbol{x},\boldsymbol{t})(k)$ and $\boldsymbol{v}'(\boldsymbol{x},\boldsymbol{t})(k)$, the kth component of these vectors, respectively. We define the similarity between $(\boldsymbol{x},\boldsymbol{t})$ and $(\boldsymbol{x}',\boldsymbol{t}')$ based on an alignment $\boldsymbol{\pi}$ as

---

[5]Suppose that K is a kernel and we make a matrix G for which the i,jth element is $K(x_i,x_j)$ for some collection of inputs $x_1,\ldots,x_n$. The kernel K is termed diagonally dominant when the values on the diagonal of G are much larger than the off diagonal elements.

$$k_{\boldsymbol{\pi}}^{\beta,\sigma}((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}')) = \prod_{k=1}^{K} \frac{\chi_2^{\beta,\sigma}(\boldsymbol{v}(k),\boldsymbol{v}'(k))}{1-\chi_1^{\beta,\sigma}(\boldsymbol{v}(k),\boldsymbol{v}'(k))}, \tag{3.8}$$

where $\beta > 1$ and $\sigma > 0$ are tuning parameters mentioned above. The use of this particular form will allow us to construct a positive definite kernel.

3.3.1.3 A Kernel Over All Time Warping Alignments

We define the Global Time Warping kernel $K_{GTW}^{\beta,\sigma}$ as

$$K_{\mathrm{GTW}}^{\beta,\sigma}((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}')) \;=\; \sum_{\boldsymbol{\pi}\in\Pi_G(n,m)} k_{\boldsymbol{\pi}}^{\beta,\sigma}((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}')). \tag{3.9}$$

The following theorem states that $K_{\mathrm{GTW}}^{\beta,\sigma}$ is indeed a kernel under natural assumptions.

Theorem 3.3.1 *Let $\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}$ be a collection of measurements taken at times $\boldsymbol{t}$ and $\boldsymbol{t}'$, respectively. Suppose that $S : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is a symmetric positive definite function. Then $K_{GTW}^{\beta,\sigma}((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}'))$ is a symmetric positive definite function for all $\beta > 1$ and $\sigma > 0$.*

We start with an overview of the proof. First, we construct a feature space of vectors in a way that mirrors the rules for time warping alignment functions (i.e. $i(k) \le i(k+1) \le i(k)+1$, $i(1) = 1$, and $i(K) = n$). Next, we define a kernel on this feature space. The proof is concluded by showing this kernel is equal to $K_{\mathrm{GTW}}^{\beta,\sigma}((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}'))$. Suppose that $\boldsymbol{v}$ is a vector. We will frequently use the either v_i or $v(i)$ to refer to the ith component of $\boldsymbol{v}$.

Proof: Symmetry is clear from the definition. Let $\beta > 1$ and $\sigma > 0$ be given. First, we construct a feature space of replicated vectors. Suppose that $\mathcal{X}$ is a space of measurements that are taken at times $t \in \mathbb{R}$. Let $(\mathcal{X}\times\mathbb{R})^* = \bigcup_{i=1}^{\infty}(\mathcal{X}\times\mathbb{R})^i$. For each $(\boldsymbol{x},\boldsymbol{t}) \in (\mathcal{X}\times\mathbb{R})^*$ and $\boldsymbol{a} \in \mathbb{N}^{|\boldsymbol{x}|}$ we define the following mappings:

$$\begin{aligned}
\boldsymbol{x_a} &= (\underbrace{x_1,\ldots,x_1}_{a_1 \text{ times}},\underbrace{x_2,\ldots,x_2}_{a_2 \text{ times}},\ldots,\underbrace{x_n,\ldots,x_n}_{a_n \text{ times}}) \\
\boldsymbol{t_a} &= (\underbrace{t_1,\ldots,t_1}_{a_1 \text{ times}},\underbrace{t_2,\ldots,t_2}_{a_2 \text{ times}},\ldots,\underbrace{t_n,\ldots,t_n}_{a_n \text{ times}}) \\
\boldsymbol{d_a} &= (1,\ \underbrace{0,\ldots,0}_{a_1-1 \text{ times}},1,\ \underbrace{0,\ldots,0}_{a_2-1 \text{ times}},\ldots,1,\ \underbrace{0,\ldots,0}_{a_n-1 \text{ times}})
\end{aligned} \tag{3.10}$$

Finally, construct the vector $\Delta \boldsymbol{t}_a$ of the same length as $\boldsymbol{x_a}$ such that $\Delta t_a(1) = 1$ and for all $i \neq 1$

$$\Delta \boldsymbol{t}_a(i) = \begin{cases} \boldsymbol{t}_a(i) - \boldsymbol{t}_a(i-1) + 1 & d_a(i) = 1 \\ 1 & \text{otherwise} \end{cases}.$$

These are the appropriate time warping time differences $t_{i(k)} - t_{i(k-1)} + 1$ seen, for example, in Equation 3.6. The feature map, defined as $\phi(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{a}) = (\boldsymbol{x}_a, \Delta \boldsymbol{t}_a)$, maps to the replicated measurements and the time difference vectors.

The functions d_1 and d_2 are defined in Equations 3.2 and 3.3. Both of these functions are clearly conditionally positive definite(CPD)[19].

We define κ as

$$\kappa((\boldsymbol{x}, \boldsymbol{t}), (\boldsymbol{x}', \boldsymbol{t}')) = \prod_{k=1}^{m} \chi_2^{\beta,\sigma}((x_k, t_k), (x'_k, t'_k)) \tag{3.11}$$

which is positive definite, because it is a product of positive definite functions [19].

We can think of κ as a kernel on the triples $(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{a})$. We extend κ to

$$\kappa^*((\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{a}), (\boldsymbol{x}', \boldsymbol{t}', \boldsymbol{a}')) = \begin{cases} \kappa((\boldsymbol{x}_a, \Delta \boldsymbol{t}_a), (\boldsymbol{x}'_{a'}, \Delta \boldsymbol{t}'_{a'})) & |\boldsymbol{x}| = |\boldsymbol{a}|, |\boldsymbol{x}'| = |\boldsymbol{b}|, |\boldsymbol{x}_a| = |\boldsymbol{x}'_{a'}| \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbb{N}^* = \cup_{i=1}^{\infty} \mathbb{N}^i$. Then define

$$\begin{aligned}
K((\boldsymbol{x}, \boldsymbol{t}), (\boldsymbol{x}', \boldsymbol{t}')) &= \sum_{\boldsymbol{a} \in \mathbb{N}^*} \sum_{\boldsymbol{a}' \in \mathbb{N}^*} \kappa^*((\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{a}), (\boldsymbol{x}', \boldsymbol{t}', \boldsymbol{a}')) \\
&= \sum_{\substack{\boldsymbol{a} \in \mathbb{N}^n, \boldsymbol{a}' \in \mathbb{N}^m \\ |\boldsymbol{x_a}| = |\boldsymbol{x}'_{\boldsymbol{a}'}|}} \kappa((\boldsymbol{x}_a, \Delta \boldsymbol{t}_a), (\boldsymbol{x}'_{a'}, \Delta \boldsymbol{t}'_{a'})),
\end{aligned} \tag{3.12}$$

which is also a kernel, provided that the sum converges. This is due to a well known fact about the closure of kernels with respect to summation [19]. To finish the proof, we show this kernel is the same as $K^{\beta,\sigma}_{\mathrm{GTW}}$.

We wish to show the relationship between time warpings and these replicated alignments, but to accomplish this we need to define similar mappings for replicating a time warping $\boldsymbol{\pi}$.

Let $\boldsymbol{\pi} \in \Pi_G(n,m)$ with $|\boldsymbol{\pi}| = K$ and $\boldsymbol{c} \in \mathbb{N}^K$. Recall that $\pi(k) = (i(k), j(k))$ and $\Delta\pi(k) = (\Delta i(k), \Delta j(k))$. Then we define

$$\begin{aligned} \boldsymbol{\pi}_c &= (\underbrace{\pi(1),\ldots,\pi(1)}_{c_1 \text{ times}},\ldots,\underbrace{\pi(K),\ldots,\pi(K)}_{c_K \text{ times}}) \\ \Delta\boldsymbol{\pi}_c &= (\Delta\pi(1),\ \underbrace{0,\ldots,0}_{c_1-1 \text{ times}}\ ,\ldots,\Delta\pi(K),\ \underbrace{0,\ldots,0}_{c_K-1 \text{ times}}\ ) \end{aligned} \tag{3.13}$$

Now we show the relationship between the replicated vectors and the replicated time warpings. Suppose that we have $\boldsymbol{u},\boldsymbol{v} \in (\mathcal{X} \times \mathbb{R})^*$ and $\boldsymbol{a},\boldsymbol{b} \in \mathbb{N}^*$ such that $|\boldsymbol{a}| = |\boldsymbol{v}|$, $|\boldsymbol{b}| = |\boldsymbol{u}|$, and $|\boldsymbol{u}_{\boldsymbol{a}}| = |\boldsymbol{v}_{\boldsymbol{b}}|$. Each pair of $(\boldsymbol{a},\boldsymbol{b})$ that fit this description is called a $\boldsymbol{u},\boldsymbol{v}$ *couple*. Let $\boldsymbol{\pi} \in \Pi_G(n,m)$ with $|\boldsymbol{\pi}| = K$ and $\boldsymbol{c} \in \mathbb{N}^K$. We call such a pair $(\boldsymbol{\pi},\boldsymbol{c})$ a $\boldsymbol{\pi}$ *replicate*. It was noted in Cuturi et. al. [31] that there is a one-to-one and onto mapping from the set of $\boldsymbol{u},\boldsymbol{v}$ couples to the set of $\boldsymbol{\pi}$ replicates.

We will also construct replicated versions of the vectors seen in Equation 3.8. Let

$$\begin{aligned} \boldsymbol{v}_c &= \left((\boldsymbol{x}_a(k), \Delta\boldsymbol{t}_a(k)\right)_{k=1}^{|\boldsymbol{x}_a|} \\ \boldsymbol{v}'_c &= \left((\boldsymbol{x}'_{\boldsymbol{a}'}(k), \Delta\boldsymbol{t}'_{\boldsymbol{a}'}(k)\right)_{k=1}^{|\boldsymbol{x}'_{\boldsymbol{a}'}|}. \end{aligned} \tag{3.14}$$

In fact, these vectors illustrate the equivalence between $\boldsymbol{u}, \boldsymbol{v}$ couples to the set of $\boldsymbol{\pi}$ replicates. Therefore

$$\begin{aligned} K((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}')) &= \sum_{\substack{\boldsymbol{a}\in\mathbb{N}^n,\boldsymbol{a}'\in\mathbb{N}^m \\ |\boldsymbol{x}_{\boldsymbol{a}}|=|\boldsymbol{x}'_{\boldsymbol{a}'}|}} \kappa((\boldsymbol{x}_{\boldsymbol{a}},\Delta\boldsymbol{t}_{\boldsymbol{a}}),(\boldsymbol{x}'_{\boldsymbol{a}'},\Delta\boldsymbol{t}'_{\boldsymbol{a}'})) \\ &= \sum_{\substack{\boldsymbol{a}\in\mathbb{N}^n,\boldsymbol{a}'\in\mathbb{N}^m \\ |\boldsymbol{x}_{\boldsymbol{a}}|=|\boldsymbol{x}'_{\boldsymbol{a}'}|}} \prod_{k=1}^{|\boldsymbol{x}_{\boldsymbol{a}}|} \chi_2^{\beta,\sigma}((\boldsymbol{x}_{\boldsymbol{a}}(k),\Delta\boldsymbol{t}_{\boldsymbol{a}}(k)),(\boldsymbol{x}'_{\boldsymbol{a}'}(k),\Delta\boldsymbol{t}'_{\boldsymbol{a}'}(k))) \\ &= \sum_{\boldsymbol{\pi}\in\Pi_G(n,m)} \sum_{\boldsymbol{c}\in\mathbb{N}^{|\boldsymbol{\pi}|}} \prod_{k=1}^{|\boldsymbol{\pi}_{\boldsymbol{c}}|} \chi_2^{\beta,\sigma}(\boldsymbol{v}_{\boldsymbol{c}}(k),\boldsymbol{v}'_{\boldsymbol{c}}(k)) \end{aligned}$$

Let $I = \{i|\Delta\pi_c(i) = (0,0)\}$ and $I^C = \{1,2,\ldots,|\Delta\boldsymbol{\pi}_{\boldsymbol{c}}|\}/I$. I^C is the set of positions of the first instance of each unique element of an alignment and I is the set of positions for subsequent replications. Thus $i \in I$ implies $\Delta\pi(i) = (0,0)$, in which case $\chi_2^{\beta,\sigma}(\boldsymbol{v}_{\boldsymbol{c}}(k),\boldsymbol{v}'_{\boldsymbol{c}}(k)) = \chi_1^{\beta,\sigma}(\boldsymbol{v}_{\boldsymbol{c}}(k),\boldsymbol{v}'_{\boldsymbol{c}}(k))$.

$$\begin{aligned} K\left((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}')\right) &= \sum_{\boldsymbol{\pi}\in\Pi_G(n,m)} \sum_{\boldsymbol{c}\in\mathbb{N}^{|\boldsymbol{\pi}|}} \prod_{i=1}^{|\boldsymbol{\pi}_{\boldsymbol{c}}|} \chi_2^{\beta,\sigma}(\boldsymbol{v}_{\boldsymbol{c}}(k),\boldsymbol{v}'_{\boldsymbol{c}}(k))) \\ &= \sum_{\boldsymbol{\pi}\in\Pi_G(n,m)} \prod_{k=1}^{|\boldsymbol{\pi}|} \chi_2^{\beta,\sigma}(\boldsymbol{v}(k),\boldsymbol{v}'(k)) \sum_{\boldsymbol{c}\in\mathbb{N}^{|\boldsymbol{\pi}|}} \prod_{i\in I} \chi_1^{\beta,\sigma}(\boldsymbol{v}_{\boldsymbol{c}}(i),\boldsymbol{v}'_{\boldsymbol{c}}(i)) \\ &= \sum_{\boldsymbol{\pi}\in\Pi_G(n,m)} \prod_{k=1}^{|\boldsymbol{\pi}|} \chi_2^{\beta,\sigma}(\boldsymbol{v}(k),\boldsymbol{v}'(k))(1+\chi_1^{\beta,\sigma}(\boldsymbol{v}(k),\boldsymbol{v}'(k)) + \chi_1^{\beta,\sigma}(\boldsymbol{v}(k),\boldsymbol{v}'(k))^2 + \ldots \\ &= \sum_{\boldsymbol{\pi}\in\Pi_G(n,m)} \prod_{k=1}^{|\boldsymbol{\pi}|} \frac{\chi_2^{\beta,\sigma}(\boldsymbol{v}(k),\boldsymbol{v}'(k))}{1-\chi_1^{\beta,\sigma}(\boldsymbol{v}(k),\boldsymbol{v}'(k))} \\ &= K_{\mathrm{GTW}}^{\beta,\sigma}(\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}') \end{aligned} \tag{3.15}$$

To get the equivalence between lines 2 and 3 of Equation 3.15, we switch the inner-most sum and inner-most product. Line 4 of this equation is the consequence of the sum of the geometric series in line 3. Convergence of the series

$1 + \chi_1^{\beta,\sigma}(\boldsymbol{v}(k), \boldsymbol{v}'(k)) + \chi_1^{\beta,\sigma}(\boldsymbol{v}(k), \boldsymbol{v}'(k))^2 + \ldots$ is guaranteed since $\chi_1 < 1$ by construction.

3.3.2 Local Time Warp Kernel

We now present the local time warping (LTW) kernel. A global alignment is an alignment similar to those seen in the Needleman-Wunch algorithm. They force the entire sequences to be aligned from start to finish. The classical definition of time warping also imposes this condition. In some cases, however, the two sequences will show similarity only in part of the sequence. Local time warping, analogously to the Smith Waterman algorithm, allows us to align only portions of each sequence; with the unaligned portions of the sequence contributing nothing to the final score.

3.3.2.1 Local Time Warping

We define a *local time warping* as a collection of K ordered pairs $\boldsymbol{\pi} = \{(i(k), j(k))\}_{k=1}^{K}$ such that $i(k) \leq i(k+1) \leq i(k)+1, i(1) \geq 1, i(K) \leq n$ and $j(k) \leq j(k+1) \leq j(k)+1, j(1) \geq 1, j(K) \leq m$. Notice that we have loosened the requirements for $i(1) = 1$, $i(K) = m$, $j(1) = 1$, and $j(K) = n$. This allows a time warping to match two sequences locally. Let $\Pi_L(n, m)$ be the class of all local time warpings from vectors of length n to vectors of length m. This is a much larger collection of alignments than the collection of global alignments $\Pi_G(n, m)$. A local time warping is illustrated in Figure 3.2.

3.3.2.2 Local Time Warping Kernel

Suppose that $\boldsymbol{\pi} \in \Pi_L(n, m)$ with $\boldsymbol{\pi} = ((i(1), j(1)), \ldots, (i(K), j(K))$. As before, $x_1, \ldots, x_n$ are measurements taken at times $t_1, \ldots, t_n$ with $t_1 = 1$ and $x'_1, \ldots, x'_m$ are measurements taken at times $t'_1, \ldots, t'_m$ with $t'_1 = 1$. The Local Time Warping (LTW) kernel is defined analogously to the GTW kernel summing over all local time warpings, that is

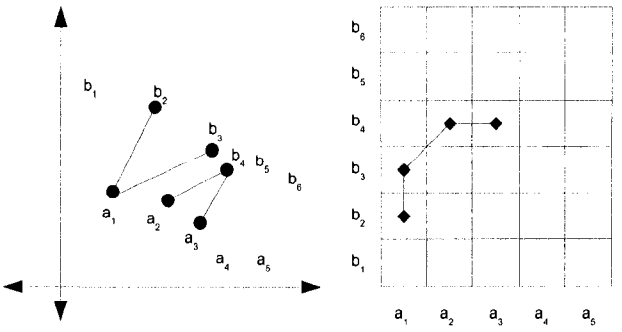


Figure 3.2: Illustration of a local time warping alignment. (a) This local time warping aligns a_1 to b_2 and b_3, and a_2 and a_3 are aligned with b_4. Notice the local time warping, unlike the global time warping, is not required to start by aligning the first elements in the sequence, a_1 and b_1, nor is it required to end by aligning the last element in each sequence, a_5 and b_6. This allows for only part of each sequence to be aligned. (b) A matrix representation of this local time warping.

$$K_{\mathrm{LTW}}^{\beta,\sigma}((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}')) \;=\; \sum_{\boldsymbol{\pi}\in\Pi_L(n,m)} k_{\boldsymbol{\pi}}^{\beta,\sigma}((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}')). \tag{3.16}$$

The following theorem states that $K_{\mathrm{LTW}}^{\beta,\sigma}$ is indeed a kernel under natural assumptions.

Theorem 3.3.2 *Let $\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}$ be a collection of measurements that were taken at times $\boldsymbol{t}$ and $\boldsymbol{t}'$, respectively. Suppose that $S : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is a symmetric positive definite function. Then $K_{LTW}^{\beta,\sigma}((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}'))$ is a symmetric positive definite function for all $\beta > 1$ and $\sigma > 0$.*

Proof: This proof is similar to the proof for the global time warping kernel and we will only highlight the differences.

The first difference is that we need to account for the starting and ending positions of the alignment. Let $\boldsymbol{s} \in \mathbb{N}^2$ such that $1 \le s_1 \le s_2 \le |\boldsymbol{x}|$, and $\boldsymbol{a} \in \mathbb{N}^{s_2-s_1+1}$; we replace the triple $(\boldsymbol{x},\boldsymbol{t},\boldsymbol{a})$ seen in the last proof with the four tuple $(\boldsymbol{x},\boldsymbol{t},\boldsymbol{a},\boldsymbol{s})$.

The term $\boldsymbol{s}$ gives the starting and stopping position of the replicated vector, Thus the replicated vector definitions become

$$\boldsymbol{x}_{a,s} = (\underbrace{x_{s_1},\ldots,x_{s_1}}_{a_1 \text{ times}},\underbrace{x_{s_1+1},\ldots,x_{s_1+1}}_{a_2 \text{ times}},\ldots,\underbrace{x_{s_2},\ldots,x_{s_2}}_{a_{s_2-s_1+1} \text{ times}}) \tag{3.17}$$

$$\boldsymbol{t}_{a,s} = (\underbrace{t_{s_1},\ldots,t_{s_1}}_{a_1 \text{ times}},\underbrace{t_{s_1+1},\ldots,t_{s_1+1}}_{a_2 \text{ times}},\ldots,\underbrace{t_{s_2},\ldots,t_{s_2}}_{a_{s_2-s_1+1} \text{ times}}) \tag{3.18}$$

$$\boldsymbol{d}_{a,s} = (1,\underbrace{0,\ldots,0}_{a_1-1 \text{ times}},1,\underbrace{0,\ldots,0}_{a_2-1 \text{ times}},\ldots,1,\underbrace{0,\ldots,0}_{a_{s_2-s_1+1}-1 \text{ times}}) \tag{3.19}$$

$$\tag{3.20}$$

The definition of the vector $\Delta\boldsymbol{t}_{a,s}$ is defined the same as before, only starting and stopping at the new positions.

The replicated terms for $\boldsymbol{\pi}$, namely $\boldsymbol{\pi_c}, \Delta\boldsymbol{\pi_c}, \boldsymbol{v_c}$, and $\boldsymbol{u_c}$, remain unchanged. To simplify the notation, for each $\boldsymbol{x}$, we define the set of acceptable replication pairs $(\boldsymbol{a}, \boldsymbol{s})$ as

$$S_{\boldsymbol{x}} = \left\{(\boldsymbol{a},\boldsymbol{s}) | \boldsymbol{s} \in \mathbb{N}^2, \boldsymbol{a} \in \mathbb{N}^*, 1 \leq \boldsymbol{s}_1 \leq \boldsymbol{s}_2 \leq |\boldsymbol{x}|, |\boldsymbol{a}| = \boldsymbol{s}_2 - \boldsymbol{s}_1 + 1\right\}$$

Again we have an equivalence between $\boldsymbol{u}, \boldsymbol{v}$ couples to the set of $\boldsymbol{\pi}$ replicates. To see this, we simply modify the define of a $\boldsymbol{u}, \boldsymbol{v}$ couple as follows: Suppose that we have $(\boldsymbol{x}, \boldsymbol{t}), (\boldsymbol{x}', \boldsymbol{t}') \in (\mathcal{X} \times \mathbb{R})^*$, $(\boldsymbol{a}, \boldsymbol{s}) \in S_{\boldsymbol{x}}$, and $(\boldsymbol{a}', \boldsymbol{s}') \in S_{\boldsymbol{x}'}$ We define a $\boldsymbol{u}, \boldsymbol{v}$ *couple* as the set $(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{a}, \boldsymbol{a}')$.

Then,$(\boldsymbol{\pi}, \boldsymbol{c})$ is equivalent to $((\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{s}, \boldsymbol{a}), (\boldsymbol{x}', \boldsymbol{t}', \boldsymbol{s}', \boldsymbol{a}'))$ provided

1. $\pi(1) = (s_1, s_1')$
2. $\pi(K) = (s_2, s_2')$
3. $\Delta\pi_c(k) = (d_{a,s}(k), d'_{a',s}(k)), \forall k$

Once we have established this equivalence, the only additional changes from the first proof come in the form of changes to the conditions related to sums and indicator functions.

For example, the new definition of κ^* is

$$\kappa^*((\boldsymbol{x},\boldsymbol{t},\boldsymbol{a},\boldsymbol{s}),(\boldsymbol{x}',\boldsymbol{t}',\boldsymbol{a}',\boldsymbol{s}')) = \begin{cases} \kappa((\boldsymbol{x}_{\boldsymbol{a},\boldsymbol{s}},\Delta\boldsymbol{t}_{\boldsymbol{a},\boldsymbol{s}}),(\boldsymbol{x}'_{\boldsymbol{a}',\boldsymbol{s}'},\Delta\boldsymbol{t}'_{\boldsymbol{a}',\boldsymbol{s}'})) & (\boldsymbol{a},\boldsymbol{s}) \in S_{\boldsymbol{x}}, (\boldsymbol{a}',\boldsymbol{s}') \in S_{\boldsymbol{x}'} \\ & |\boldsymbol{x}_{\boldsymbol{a},\boldsymbol{s}}| = |\boldsymbol{x}'_{\boldsymbol{a}',\boldsymbol{s}'}| \\ 0 & \text{otherwise} \end{cases}$$

where the old condition for a non-zero element, $|\boldsymbol{x}| = |\boldsymbol{a}|, |\boldsymbol{x}'| = |\boldsymbol{b}|, |\boldsymbol{x}_a| = |\boldsymbol{x}'_b|$ is replaced with a condition incorporating the start and stop positions. K is defined to be summed over all replicated local time warpings, thus

$$\begin{aligned} K((\boldsymbol{x},\boldsymbol{t}),(\boldsymbol{x}',\boldsymbol{t}')) &= \sum_{\boldsymbol{s},\boldsymbol{s}'\in\mathbb{N}^2} \sum_{\boldsymbol{a},\boldsymbol{a}'\in\mathbb{N}^*} \kappa^*((\boldsymbol{x},\boldsymbol{t},\boldsymbol{a},\boldsymbol{s}),(\boldsymbol{x}',\boldsymbol{t}',\boldsymbol{a}',\boldsymbol{s}')) \\ &= \sum_{\substack{(\boldsymbol{a},\boldsymbol{s})\in S_{\boldsymbol{x}} \\ (\boldsymbol{a}',\boldsymbol{s}')\in S_{\boldsymbol{x}'} \\ |\boldsymbol{x}_{\boldsymbol{a},\boldsymbol{s}}| = |\boldsymbol{x}'_{\boldsymbol{a}',\boldsymbol{s}'}|}} \kappa((\boldsymbol{x}_{\boldsymbol{a},\boldsymbol{s}},\Delta\boldsymbol{t}_{\boldsymbol{a},\boldsymbol{s}}),(\boldsymbol{x}'_{\boldsymbol{a}',\boldsymbol{s}'},\Delta\boldsymbol{t}'_{\boldsymbol{a}',\boldsymbol{s}'})) \\ &= \sum_{\substack{(\boldsymbol{a},\boldsymbol{s})\in S_{\boldsymbol{x}} \\ (\boldsymbol{a}',\boldsymbol{s}')\in S_{\boldsymbol{x}'} \\ |\boldsymbol{x}_{\boldsymbol{a},\boldsymbol{s}}| = |\boldsymbol{x}'_{\boldsymbol{a}',\boldsymbol{s}'}|}} \prod_{k=1}^{|\boldsymbol{x}_{\boldsymbol{a},\boldsymbol{s}}|} \chi_2^{\beta,\sigma}((\boldsymbol{x}_{\boldsymbol{a},\boldsymbol{s}}(k),\Delta\boldsymbol{t}_{\boldsymbol{a},\boldsymbol{s}}(k)),(\boldsymbol{x}'_{\boldsymbol{a}',\boldsymbol{s}'}(k),\Delta\boldsymbol{t}'_{\boldsymbol{a}',\boldsymbol{s}'}(k))) \\ &= \sum_{\boldsymbol{\pi}\in\Pi_L(n,m)} \sum_{\boldsymbol{c}\in\mathbb{N}^{|\boldsymbol{\pi}|}} \prod_{k=1}^{|\boldsymbol{\pi}_c|} \chi_2^{\beta,\sigma}(\boldsymbol{v}_c(k),\boldsymbol{u}_c(k)) \end{aligned}$$

The remainder of the proof is identical to the end of the first proof.

3.3.3 Efficient Computation

In this section, we will show that the entries in both time warping kernels can be computed in $O(|\boldsymbol{x}||\boldsymbol{y}|)$ time using dynamic programming techniques. First, the dynamic programming algorithm is presented. Then, the proof that this algorithm indeed computes the value of the kernel is given and a proof is sketched.

Theorem 3.3.3 *Let $T_G(i,j)$ be defined recursively as shown in Algorithm 1. Then $K^{\beta}_{GTW}(\boldsymbol{x},\boldsymbol{y}) = T_G(|\boldsymbol{x}|,|\boldsymbol{y}|)$.*

Algorithm 1 Dynamic Global Time Warping Kernel

Initialize:

$H(0,0) = T(0,0) = 0$

for $i = 1, m$ **do**

$H(i,0) = T(i,0) = 0$

end for

for $j = 1, m$ **do**

$H(0,j) = T(0,j) = 0$

end for

Main Loop:

for $i = 1, m$ **do**

for $j = 1, n$ **do**

$C1 = \exp((1/\sigma) * (S(x_i, x_i) + S(y_j, y_j) - 2S(x_i, y_j) + \ln(\beta))$

$C2_{1,1} = \exp((1/\sigma) * (\Delta t_x(i)\Delta t_x(i)S(x_i, x_i) + \Delta t_y(j)\Delta t_y(j)S(y_j, y_j) - 2\Delta t_x(i)\Delta t_y(j)S(x_i, y_j) + \ln(\beta))$

$C2_{1,0} = \exp((1/\sigma)*(\Delta t_x(i)\Delta t_x(i)S(x_i, x_i)+S(y_j, y_j)-2\Delta t_x(i)S(x_i, y_j)+\ln(\beta))$

$C2_{0,1} = \exp((1/\sigma)*(S(x_i, x_i)+\Delta t_y(j)\Delta t_y(j)S(y_j, y_j)-2\Delta t_y(j)S(x_i, y_j)+\ln(\beta))$

$L_{1,0} = \frac{\exp(C2_{1,0})}{1-\exp(C1)}$

$L_{0,1} = \frac{\exp(C2_{0,1})}{1-\exp(C1)}$

$L_{1,1} = \frac{\exp(C2_{1,1})}{1-\exp(C1)}$

if i =1 and j = 1 **then**

$H(i,j) = L_{1,1}$

else

$H(i,j) = L_{1,0}H(i-1,j) + L_{0,1}H(i,j-1) + L_{1,1}H(i-1,j-1)$

end if

$T(i,j) = H(i,j) + T(i-1,j) + T(i,j-1) - T(i-1,j-1)$

end for

end for

Algorithm 2 Dynamic Local Time Warping Kernel

Initialize:

$H(0,0) = T(0,0) = 0$

for $i = 1, m$ **do**

$H(i,0) = T(i,0) = 0$

end for

for $j = 1, m$ **do**

$H(0,j) = T(0,j) = 0$

end for

Main Loop:

for $i = 1, m$ **do**

for $j = 1, n$ **do**

$C1 = \exp((1/\sigma) * (S(x_i, x_i) + S(y_j, y_j) - 2S(x_i, y_j) + \ln(\beta))$

$C2_{1,1} = \exp((1/\sigma) * (\Delta t_x(i)\Delta t_x(i)S(x_i, x_i) + \Delta t_y(j)\Delta t_y(j)S(y_j, y_j) - 2\Delta t_x(i)\Delta t_y(j)S(x_i, y_j) + \ln(\beta))$

$C2_{1,0} = \exp((1/\sigma)*(\Delta t_x(i)\Delta t_x(i)S(x_i, x_i)+S(y_j, y_j)-2\Delta t_x(i)S(x_i, y_j)+\ln(\beta))$

$C2_{0,1} = \exp((1/\sigma)*(S(x_i, x_i)+\Delta t_y(j)\Delta t_y(j)S(y_j, y_j)-2\Delta t_y(j)S(x_i, y_j)+\ln(\beta))$

$L_{1,0} = \frac{\exp(C2_{1,0})}{1-\exp(C1)}$

$L_{0,1} = \frac{\exp(C2_{0,1})}{1-\exp(C1)}$

$L_{1,1} = \frac{\exp(C2_{1,1})}{1-\exp(C1)}$

$L_{0,0} = \frac{\exp(C1)}{1-\exp(C1)}$

$H(i,j) = L_{1,0}H(i-1,j) + L_{0,1})H(i,j-1) + L_{1,1})H(i-1,j-1) + L_{0,0}$

$T(i,j) = H(i,j) + T(i-1,j) + T(i,j-1) - T(i-1,j-1)$

end for

end for

Theorem 3.3.4 *Let $T_L(i,j)$ be defined recursively as shown in Algorithm 2. Then $K^{\beta}_{LTW}(\boldsymbol{x},\boldsymbol{y}) = T_L(|\boldsymbol{x}|,|\boldsymbol{y}|)$.*

Proof: The proof of both theorems is typical of dynamic programming algorithms, and will only be outlined. The local version of the kernel needs two modifications to the DTW algorithm presented in Kruskal [15]. First, an additional option must be added at each step that represents starting a new alignment at that particular point. Second, replacing the minimum with a sum over exponentials leads to the sum of products in Algorithms 1 and 2. The global version only needs to replace the minimum with a sum.

3.3.4 Experiments

In this section, we construct data from two generative models and illustrate the performance advantage of the new time warping kernels.

3.3.4.1 Three Classification Problems

Our intuition leads us to believe that the new time warping kernels will perform better than the kernel due to Cuturi et. al. [31] when the time between observations is important. In this section, we illustrate that our intuition is correct, at least in one situation.

Consider a family of distributions. Each distribution will have the same basic structure, that of a Markov model with explicit state duration distributions. The distributions will differ in their emission and duration distributions.

The specific model we used is illustrated in Figure 3.3. The first state, the *emission state*, emits one of two symbols, a or b. Exactly one symbol will be emitted before a transition to the second state. The second state, called the *duration state*, will not emit symbols. We will stay in this state a random amount of time. For simplicity, we have chosen a discrete distribution for the durations. Note that these

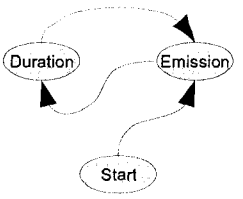


Figure 3.3: A generative model. Sequences alternate between the two states: *Emission* and *Duration*. The Emission state lasts one time unit and emits a random symbol. The Duration state lasts a random number of time units, and emits no symbols.

sequences are *timed sequences* with the duration state providing the random time between observations.

We will explore the ability of our kernel and the kernel by Cuturi et. al. [31] at classifying examples generated by two different instances of this model. The instances will either differ in emission distributions, in duration distributions, or in both distributions. A description of the three specific classification problems follows.

The first classification problem, illustrates the ability of the two kernels to differentiate between instances of the model that vary only in the duration distributions. The distributions for each model are illustrated in Figure 3.4.

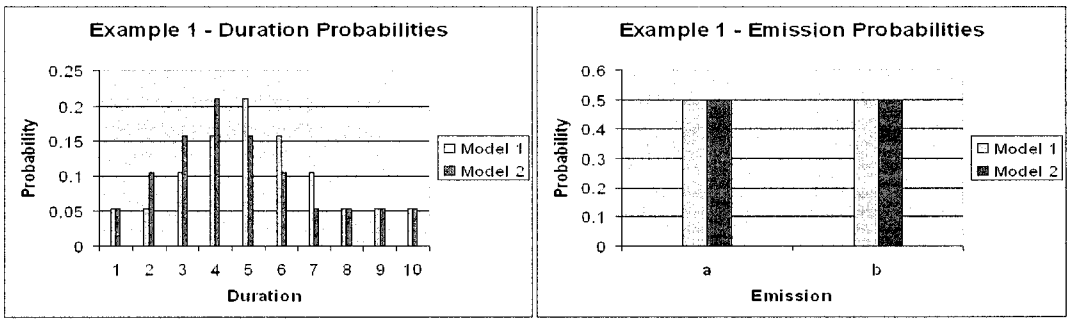


Figure 3.4: The first classification problem. In this classification problem, the duration distributions are slightly different while the emission distributions are the same.

In the second classification problem, we illustrate the ability of the two kernels to differentiate between two instances of the model that differ only in the emission distributions. See Figure 3.5 for an illustration of the distributions.

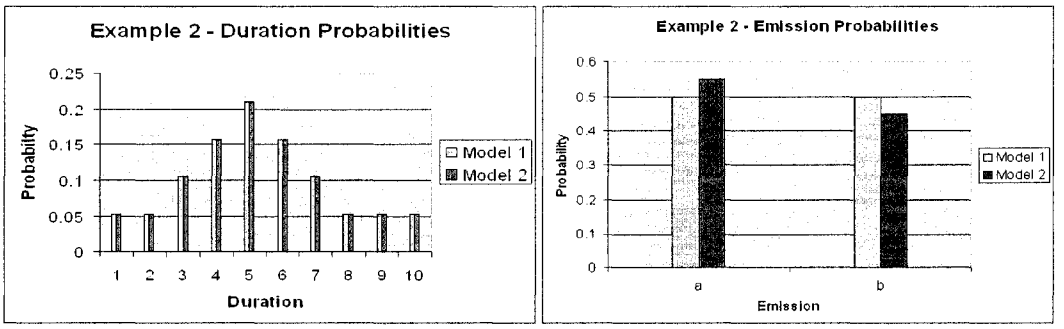


Figure 3.5: The second classification problem. The two models in this classification problem have identical duration distributions, and slightly different emission distributions.

The third classification problem explores the ability of the two kernels to differentiate between two models that differ in both the duration and emission distributions. Figure 3.6 shows the different distributions.

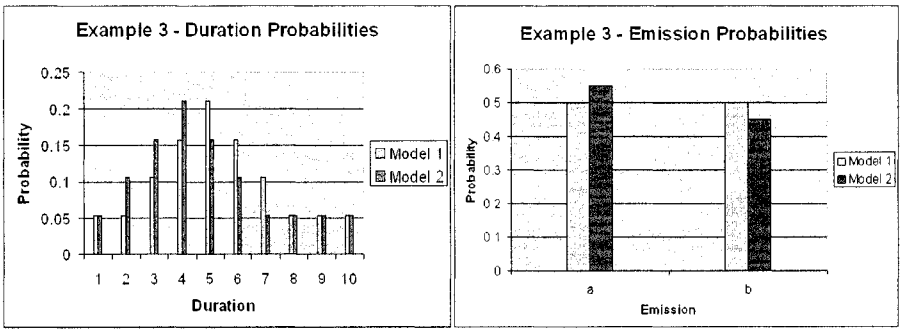


Figure 3.6: The third classification problem. For the final classification problem, both the duration and emission distributions are slightly different.

3.3.4.2 Classification Methods

We will use two different methods to classify observations for each of the three classification problems. A support vector machine (SVM) will be the base of both classifiers, one of which uses one of the proposed time warping kernels and the other uses an adaptation of Cuturi et. al.'s time warping kernel.

In preliminary testing the local version the new time warping kernel outperformed the global variant. For this reason, we will focus on the results of the local time warping kernel.

To make for a fair comparison, Cuturi et. al.'s kernel was also adapted to be a local alignment method, which involved minor changes to the dynamic programming equations. The performance of the local version was compared to the performance of the standard (global) kernel. Again, the local version had the best performance, and will be used for all comparisons.

Next we describe the specific implementation of each kernel. We start by defining a kernel on the emissions. Since we have two possible observations the kernel is a 2x2 matrix between observations. We use the matrix

$$S = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \tag{3.21}$$

which is non-negative definite. This kernel is then used to define the distances d_1 and d_2 given in Equations 3.2 and 3.3, thus defining an instance of the newly proposed local time warping kernel.

The kernel by Cuturi et. al. requires us to define a kernel on emissions of the form $\chi(x, y) = \exp(-\phi(x, y))$, such that χ is positive definite and strictly less than 1 in absolute value. We use $\chi(x, y) = \exp(-d_1(x, y)/\sigma + \beta)$, which was shown to fit these requirements in Theorem 3.3.3.

As is typical for alignment kernels, the values of the kernel can be large and the matrix could be diagonally dominant(see, for example [14]). Diagonal dominance in

particular is a problem for SVMs, as it means that the feature vectors are all nearly orthogonal, making classification difficult. As noted earlier, the tuning parameters β and σ will give us some control over this problem.

Diagonal dominance was a problem encountered by Cuturi et. al. when illustrating the performance of their kernel [31]. They suggest taking the log of the kernel. Taking the log of a kernel matrix can cause some eigenvalues to become negative, and Cuturi et. al. added the identity matrix times the absolute value of the smallest eigenvalue to the kernel to guarantee the kernel remained positive definite. Unfortunately, this action can only increase any diagonal dominance.

The addition of the parameter β allowed us to control diagonal dominance. For each of the kernels we compared the performance of the kernel "as-is" versus the log of the kernel plus a multiple of the identity matrix. In both cases, using the kernel "as-is" led to better performance. For this reason, we refrain from "massaging" either kernel in the following experiments.

3.3.4.3 Parameter Tuning

Next we need to decide on the final form of the kernel. This includes deciding on kernel normalization as well as fixing all tuning parameters.

It is sometimes advantageous to normalize a kernel through the application of various functions. Suppose that K is a kernel. Then, for example, we might apply the cosine kernel

$$K_{cosine}(x, y) = K(x, y)/\sqrt{K(x, x)K(y, y)},$$

which has the effect of normalizing all vectors in the feature space to length one. When the application of such a kernel only changes the lengths of the vectors in the feature space, we refer to the action as *kernel normalization.*

It may also be beneficial to change the feature space by nesting on kernel inside another. For example, we may apply the Gaussian kernel

$$K_G(x, y) = \exp(-\gamma * (K(x, x) - 2K(x, y) + K(y, y))).$$

We may even put these kernels to use sequentially, i.e. apply a cosine kernel then a Gaussian kernel.

We also have many parameters to tune. The parameters β and σ were selected in a way to reduce the size of the kernel entries and reduce diagonal dominance. Other parameters, such as the soft margin SVM cost parameter C or parameters such as the Gaussian kernel's γ, as well as any kernel transformations, were selected using cross-validation. A list of possible kernels was made and for each candidate, a grid search over the space of tuning parameters (not including β and σ from above) was performed to estimate performance. The performance measure was success rate when performing a 5-fold cross-validation on a set of examples set aside for tuning. The final form of the kernel was the transformation/tuning parameter combination with the best performance.

3.3.4.4 Testing the Classifier's Performance

Let p be the average classifier accuracy over all training sets of size 500. We wish to generate a confidence interval for p based on the performance of ten such randomly selected classifiers. This is a two part estimation problem, where first the performance of each classifier is independently estimated[6], then the performance of all classifiers trained on 500 examples is estimated from the estimated performance of ten classifiers. An appropriate method of doing this is to generate a confidence interval for the mean of the estimated performance of the ten classifiers. We used Bonferroni's method to ensure that the comparisons for each classification problem had no worse than a 5% chance of any confidence interval being incorrect.

To gauge the relative success for each classifier, the performance of the Bayes classifier was also estimated. The Bayes classifier is the classifier that has the highest

---

[6]For each classifier, a test set of 1000 examples was independently generated and used to estimate the performance.

possible success rate. This is achieved by using full knowledge of the model and applying the log-odds classification rule. The success rate of the Bayes classifier was estimated for each classification problem by constructing a confidence interval for the true success rate using the success rate of 2 million examples, half from each distribution. This confidence interval was constructed simultaneously with the above intervals.

3.3.4.5 Results

Table 3.1: Summary of results for classification problems 1, 2, and 3. The Bayes classifier uses explicit knowledge of the model to pick the most probable class. This classifier has the best possible success rate. The Local Alignment Kernel is a time warping kernel that includes the time coefficients present in the classical time warping. The kernel due to Cuturi et.al. is also time warping kernel, but without the time coefficients.

	95% Simultaneous CI for Mean Success Rate		
Classification Problem	Bayes	LTW Kernel	Cuturi et. al. Kernel
1	$(0.938, 0.939)$	$(0.924, 0.937)$	$(0.738, 0.767)$
2	$(0.656, 0.658)$	$(0.640, 0.663)$	$(0.642, 0.679)$
3	$(0.945, 0.946)$	$(0.922, 0.937)$	$(0.760, 0.796)$

The results for the experiments are given in Table 3.1. We see that the LTW kernel does significantly better than local version of the kernel by Cuturi et. al. when there is a difference in the duration distributions [7] and has similar performance when the models have the same duration distributions. Furthermore, the LTW kernel is fairly close to the best possible success rate, that of the Bayes classifier! All of the differences are statistically significant at a 5% family error rate.

---

[7] It may seem counter-intuitive that the kernel by Cuturi et. can do much better than 50% classification success on the first example, after all it does not use any explicit information about the waiting times. Keep in mind how the examples were generated. To mirror the data in the diabetes problem, we generated observations from a point process for a fixed window of time, which resulted in a variable number of observations. The lengths of the string contain some information about the duration distribution, and it is our contention that it is this information that allows the kernel by Cuturi et. to perform better than a 50% success rate.

3.4 Conclusions

We presented two new kernels for time warping, one based on global alignment and the other on local alignment, that better capture the essence of time warping by including the timing coefficients that will properly penalize stretching time to align two sequences. We have also addressed an issue with the definition of classical time warping by substituting similarities for distances.

For specific classification problems, we have shown that our local time warping kernel performs better than the kernel by Cuturi et. al. at classifying timed sequences when the timing between observations contains relevant information.

Furthermore, it was shown that the performance of the new kernel for these classification problems was very close to the best obtainable success rates, that of the Bayes classifier. This is a promising result, as our kernel makes no use of information about the distribution from which the data is generated.

3.5 Further Work

The main challenge in using alignment methods, including the LTW and GTW kernels, is that of computation time. Computing even moderately sized kernel matrices can take a long time. We will explore methods of reducing this computational burden, while attempting to keep the flavor of classical time warping intact.

We would also like to explore other applications of this kernel. A problem that has been getting recent attention is that of classifying internet network traffic (see, for example, Moore and Papagiannaki [32]). Like claims data, this is another example of data that is a timed sequence.

When two computers communicate over the internet, they send small "packets" of information. Each packet has a number of associated features: protocol (e.g. TCP, UDP), size, port number, source, and destination. A *flow* is defined as a sequence of packets from a source to a destination all with the same port number and protocol. In

their work on this problem, Nguyen and Armitage [33] successfully used the minimum, maximum, mean and standard deviations of the inter-packet arrival times as part of their feature set. This leads us to believe that this is a good application for either the GTW or LTW kernel.

Chapter 4

CLASSIFYING RISK OF TYPE 2 DIABETES USING CLAIMS DATA

4.1 Introduction

In this chapter, we will present our research on predicting patients at risk of type 2 diabetes. Diabetes is a major problem both in terms of number of diagnosed individuals [2] and the financial burden [2] these individuals accrue. It has been shown that early detection of diabetes combined with an intervention can delay or prevent the onset of the disease [3].

Current methods for identifying patients in need of an intervention involves a costly and inconvenient test [5]. In this chapter, we have developed methods that allow accurate prediction of patients at risk of type 2 diabetes based on claims data alone. These methods can be used as an initial screening, cutting down on the number of individuals that need the costly test and helping identify patients that might not have otherwise been screened. To the best of our knowledge, no other paper attempts this classification problem using only insurance claims data.

In this section, we will motivate the problem by discussing the specifics of type 2 diabetes, look at specific reasons early detection of type 2 diabetes is useful, discuss the type of data that we have available, and ask the specific research questions we wish to answer: "Can you predict which patients will receive their first diabetes diagnosis using a three years window of their insurance claims data?" and "Can this be done well in advance of the date of this first diagnosis"?

4.1.0.6 Facts about Type 2 Diabetes

Next we will summarize the facts about diabetes presented in the Center for Disease Control and Prevention's *National diabetes fact sheet, 2005* [2]. Diabetes is a disease in which the body either doesn't produce insulin (type 1) or is resistant to insulin (type 2). A third form of diabetes (gestational) affects pregnant women.

There are over 20 million Americans with diabetes. Type 1 diabetes only affects 5-10% of people with diabetes and is usually diagnosed in children or young adults. Type 2 diabetes makes up the majority of the Americans with diabetes, between 90% and 95% of all diagnosed cases of diabetes.

Insulin allows glucose to enter a cell. People with type 2 diabetes experience a build up of glucose in the blood due to their resistance to insulin. As glucose builds in the blood, the demand for insulin rises and the body slowly loses its ability to produce it. The complications of the disease are heart disease and stroke, high blood pressure, blindness, kidney disease, nervous system disease, amputations, dental disease, and complications of pregnancy. The treatments used to prevent these complications are glucose control, blood pressure control, control of blood lipids, and preventative care practices for the eyes, kidneys, and feet. It is estimated that the total cost of diabetes care was $132 billion dollars in 2002.

People that have some resistance to insulin, but not enough to be considered diabetic are considered pre-diabetic. To give the medical definition of type 2 diabetes and pre-diabetes, we need to discuss the two tests used to diagnose patients, the fasting plasma glucose test (FPG) and the oral glucose tolerance test (OGTT) and the associated conditions they detect, impaired fasting glucose (IFG) and impaired glucose tolerance (IGT) respectively.

IFG is detected using the fasting plasma glucose test (FPG). The FPG test involves first having a patient fast for at least 8 hours. At this point, a blood sample is taken and the persons blood glucose level is tested. If the blood glucose level is 126

mg/dl or more, the person is retested. If the second test is consistent, the person is diagnosed as having diabetes. If the patient has a blood glucose level greater than or equal to 100 mg/dl, but less than 126 mg/dl; they are considered to have pre-diabetes.

IGT is tested for using the oral glucose tolerance test (OGTT). The OGTT involves having a patient drink a solution containing 75g of glucose. Their blood glucose level is measured 2 hours after drinking the solution. If their blood glucose level is greater than or equal to 200 mg/dl they are diagnosed as having diabetes. Patients with a blood glucose level between 140 mg/dl and 200 mg/dl are pre-diabetic [8].

It is important to note that the existence of a pre-diabetes condition allows for some hope in delaying or preventing entirely the onset of type 2 diabetes. There have been a number of successful attempts at an intervention program designed to detect people at risk of type 2 diabetes and help them make the necessary lifestyle changes needed to lower their risk [3].

In 2002, the American Diabetes Association released their position statement regarding the prevention of diabetes [3]. They concluded that (1) type 2 diabetes could be prevented or delayed, (2) persons at risk could be determined clinically using the above mentioned tests, and (3) intervention programs that focus on lifestyle changes such as weight loss and modest exercise are a cost-effective way of reducing the adverse effects of diabetes. Finally, they called for research on the question "What is the most effective way to identify individuals who are at high risk for unrecognized IFG or IGT?". We feel the following work addresses this need.

4.1.0.7 Previous Work

The presence of diabetes, along with the risk of future diabetes can be determined clinically using the FPG or OGTT tests. There have been other clinical alternative

---

[8]To be clear, type 2 diabetes is clinically defined as a person with blood glucose level is 126 mg/dl or more after the FPG test or a blood glucose level is greater than or equal to 200 mg/dl after the OGTT test.

suggested (see for example [5]) but these tests remain the primary methods for detection of type 2 diabetes and pre-diabetes.

In Smith et. al. [34], the authors looked at predicting the onset of type 2 diabetes in women of the Pima Indian tribe. They used the following variables to make their prediction: number of times pregnant, plasma glucose concentration from the OGTT, diastolic blood pressure, triceps skin fold thickness, 2-hour serum insulin, body mass index (BMI), a diabetes pedigree function, and age. These data were made public through the UCI Machine Learning Repository [35] and many papers were published in attempts to improve classification performance [35]. Note that all of this research is on a small data set that is relevant only to a particular population.

Claims data have been used to evaluate the quality of care of different providers, measure patients' use of various medical services, and identify patients with various diseases [36]. For diabetes in particular, there have been examples of using claims to measure the adherence to medications [37] and identify existing cases of the disease (see, for example, [38] and [39]). To the best of our knowledge, no one has attempted to predict patients at risk of a diabetes diagnosis in the future using claims data as their sole source of information.

4.1.1 Problem Statement

Our main point of interest is the following question: How well can one predict patients at risk of type 2 diabetes based on their insurance claims? First, we wish to determine if we can predict a patient's first diabetes diagnosis with all the information up to the day before the diagnosis. We also wish to determine how well we can predict a diabetes diagnosis without access to the most recent claims history.

4.2 Methodology

In this section, we will give an overview of the methods that we used to approach this problem. This includes the classification method that we selected, SVM and

kernel methods, our general approach for the different types of data, and the resulting kernels.

4.2.1 Our Approach

The data in this problem is interesting and complex. Each patient has a variable number of claims in their claim history. These claims also contain a variety of types of information, including diagnosis, procedure and drug codes, cost information, and demographic information. The different types of claims contain different components. Even claims of the same type may contain a different number of elements. All of these facts make it very hard to apply any method that relies on a fixed number of input variables.

4.2.1.1 The Classifier

SVMs, on the other hand, have had demonstrated success in a wide variety of problems that involve sequence data. SVMs have been used successfully in text classification [40]. They have also been used for a wide variety of problems in computational biology which involve sequences of different lengths, for example classification of genes and proteins and prediction along the DNA or protein strand [41]. SVMs have also been used successfully in content-based audio classification and retrieval [42].

Furthermore, the properties of kernels allow the ability to easily combine kernels for very different types of data. For example, Noble [41] pointed out a number of ways that different types of information have been combined using SVMs. Perhaps most importantly, kernels do not require data to come in the form of a fixed length vector, and in fact, have been constructed for structures such as variable-length strings, graphs, and trees [43].

For these reasons, we have chosen to use kernels in conjunction with an SVM as our classifier.

4.2.1.2 Dealing with Heterogenous Data

Our strategy in approaching this problem was to first split the data into its different types and find the best kernel we could for each specific type of information separately. The three timed sequences that we generate for each patient are discussed in Section 1.3.2 and illustrated in Figure 1.2. We also include the demographic information for each patient in the classification process. A kernel was constructed for each form of data. The kernels were then combined into a single kernel. Specifics on each kernel follow.

4.2.2 Kernel Construction and Adaptation

In this section we will give details of our kernel construction for each type of data: diagnosis/procedure codes, drug codes, cost, and demographic information.

4.2.2.1 Bag of Codes

We have constructed a number of kernels to deal with diagnosis, procedure and drug codes: the frequency-based Bag of Codes kernel (BoC), zero-one Bag of Codes kernel (BoCI), the frequency-based Stratified Bag of Codes kernel (SBoC), zero-one Stratified Bag of Codes kernel (SBoCI)

For each of these Bag of Code kernels, we create one feature for each code in the database and let the value of the feature either be an indicator of the presence of that code in a patient's claim history (BoCI features), or the count of the number of times the code appears in the patients claim history (BoC features). The kernels are the inner products of these feature vectors. These kernels were inspired by a classic method in text classification called the "bag of words" technique [11].

The BoC and BoCI kernels completely ignore the time at which a code is observed. To introduce some time information into the feature set, we implement the following change: We first divide each patient's claim window into n time strata of a fixed number of time units w. We set the number of strata, n such that all patients

have a claim window of at least $n * w$ time units. For patients that have a claim window longer than $n * w$ time units, the most recent $n * w$ units are used. We then generate either the BoC or BoCI features for each stratum and concatenate them into one combined feature vector. The inner product of this vector gives the SBoC or SBoCI kernels, respectively.

4.2.2.2 Hidden Markov Models and the Fisher Kernel

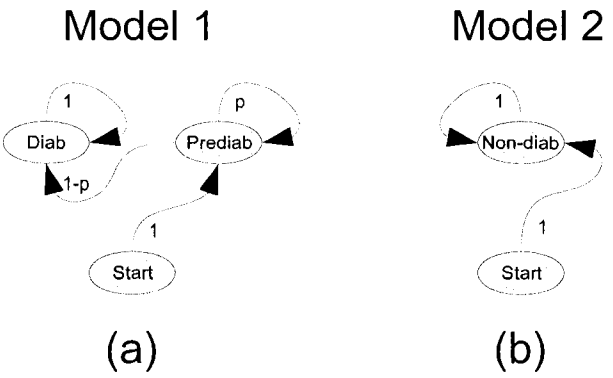


Figure 4.1: Two HMM models. (a) Represents patients likely to become diabetic. Two states are prediabetic and diabetic. (b) Represents patients unlikely to become diabetic, thus one non-diabetic state.

We construct a model for patients that will develop diabetes and a model for patients that will not, allowing us to model the data probabilistically and thus take advantage of the log-odds classifier. The first model has two states and is for patients that will become diabetic, state one being non-diabetic and state two being diabetic. The second model is for patients that do not develop diabetes. It is a one-state model that only contains the non-diabetic state. See Figure 4.1 for an illustration of the two models.

The emission probabilities for each state were estimated on a set of examples separate from those that will be later used to tune and test the various methods.

The transition probabilities were treated as tuning parameters. One way to use these models is to make a classification using the log likelihood ratio based on the two models.

We also constructed a Fisher kernel. As indicated in section 2.1.4.2, we imagine we are constructing a hypothesis test comparing two models, in this case the diabetes and non-diabetes HMMs. We treated the diabetes model, the model that includes both the diabetic and non-diabetic states, as the null model. Thus, the fisher kernel measures departures from the diabetes model.

4.2.2.3 Time Warping

We have developed two new kernels based on time warping presented in Chapter 3. Both a local version, called the Local Time Warping kernel (LTW), and global version, called the Global Time Warping Kernel (GTW), are given.

We will be aligning codes, either drug or diagnosis/procedure, using one of these kernels. We must decide if a local or global alignment method is appropriate. It is commonly assumed that the progression of diabetes is a continuous process. Since each patient in the database will most likely be at a different point in this process, it is important to be able to compare various portions of each claim history to look for similarity. Also, certain portions of the claim history maybe unrelated to diabetes. Because of these two facts, we will be using the LTW kernel.

To implement the LTW kernel, we must specify a kernel on the codes. We used the kernel $K(c_1, c_2) = I_{c_1=c_2}(c_1, c_2)$ to compare the aligned codes, where $I_{c_1=c_2}$ is the indicator function that is 1 when $c_1 = c_2$ and zero otherwise.

4.3 Experiments

This section contains details on the experiments that were used to develop kernels for use in identifying patients at risk of type 2 diabetes and details of how the performance of the kernels was estimated. This includes the performance when using

complete or truncated patient histories, as well as the performance when using a full set of diagnosis, drug, and procedure codes or a subset of the codes.

4.3.1 Data Set

In this section, we will summarize the experiments used to evaluate and test the performance of various kernels. First, we discuss the construction of the data sets. Next, we give an overview of the process by which the various kernels will be tuned and tested. Finally, we give a description of each of the kernels that will be compared and point out any parameters that need tuning.

4.3.1.1 Selection of Examples

Let's define some of the important terms that we use when referring to the diabetes data. Our patients are members of the MarketScan® database. Each patient has a date on which they enrolled for the policy. We call this date *startenrol.* If they ended the policy, we keep track of the date the policy ended and call this value *endenrol.* If the patient was still a policy holder on 12/31/2003, we let endenrol be this date. The dates are stored as the number of days since some particular date in time. Thus the difference between two data variables gives the difference in number of days.

A *positive example* is defined as a patient that has at least one observed diabetes diagnosis and has no insulin prescriptions prior to the first observed diabetes diagnosis. We make the assumption that the first observed diagnosis is the actual first diabetes diagnosis. We refer to the date of this first diabetes diagnosis as *mindate.*

A *negative example* is defined as a patient that with no observed diabetes diagnosis and no observed insulin prescriptions. For negative examples, we set mindate to be 12/31/2003.

We define the *claim window* in the following way:

$$claimwindow = \min(endenrol, mindate) - \max(1/1/1999, startenrol)$$

Table 4.1: Groupings of claim windows and their relative sizes

Claim Window Group	Counts		
	Positive	Negative	Total
Train-Test Set	14,746	14,754	29,500
Tuning Set	14,777	14,723	29,500
Extra Set	178,851	178,851	357,702

It is possible that a patient doesn't have complete drug information. We add the restriction that all positive and negative examples have full drug information for the years spanned by their claim window.

4.3.1.2 Division of Examples - Tuning, Train-Test, and Extra Sets

Here are the specifics on how the examples were divided. All of the positive examples with a claim window of length three to five years were split randomly into two equally sized sets, one for tuning and the other for training-testing the classifier, and an equal number of randomly selected negative examples with a 3 to 5 year claim window were added to complete each set. We refer to these sets as the *tuning* set and the *train-test* set, respectively. The total size of each set is given in Table 4.1.

The reason that we decided to split the patients with a three to four year claim window in this way is two fold. First, there appears to be little difference between the performance when 29,500 patients are used verses the full 59,000 patients (see Appendix B for learning curves). Second, it is convenient to have the tuning set and the training-testing set of the same size.

Finally, a set to be used in kernel construction, HMM parameter estimation, etc was set aside. It consisted of all diabetes patients that fit our definition of a positive example having a claim window of a length between zero and three years. A random sample of the same number of negative examples, also with a claim window between 0 to 3 years, were added to this set. We call this the *extra* set of examples.

For each group of examples, we will construct two data sets, one for each of our two research questions. We construct both a tuning and train-test set that includes

the full patient claim history up to and including the day before their initial diagnosis. We refer to these as a *complete patient history* or simply use the prefix "complete". These data are designed to answer the question: "Can we predict the onset of type 2 diabetes the day before the initial diagnosis"?

We also want to investigate our ability to detect at-risk patients early. Consequently, we construct a data set for each set of examples, train-test and tuning, that have the most recent portion of the patients' history removed. We call these the *truncated patient histories* or simply use the prefix "truncated".

When constructing the truncated data set, we were concerned that removing the same portion of the histories from all examples would leave them aligned to the first diabetes diagnosis. That is they all ended at the same spot relative to the event that was being predicted. To make sure that we weren't gaining a performance advantage through this alignment, we shortened the windows by truncating patient histories by a random amount. We constructed two data sets differing by the average amount of claims data that was removed. The first set had an average of 12 months removed and the second set had an average of 18 months removed.

4.3.2 Tuning, Model Selection, and Testing

We used the tuning set to tune the various tuning parameters for each kernel, as well as to give a preliminary comparison between the various methods. The initial comparison was done using a subset of the tuning set, and the methods were compared using the success rate resulting from five-fold cross-validation. The methods that showed promise were included in the later testing phase.

Tuning was done by performing multiple runs of cross-validation over the entire tuning set and selecting the set of parameters that maximized the success rate. Multiple runs of cross-validation were done on the same randomly selected partitions to reduce the variability in the results and aid in comparison.

The final cross-validation results were computed using the train-test set. Five runs of five-fold cross-validation were used on the tuning parameters selected in the tuning phase. The final performance is measured using the mean success rate and mean area under the ROC curve, averaged over the five runs of cross-validation. We also plot an ROC curve for each kernel. All of these curves were generated through cross-validation on the same set of randomly selected folds.

Please note that the test set was never used for kernel construction, tuning, or model selection; only the final testing phase.

4.3.3 Kernel Implementation

Recall our general strategy for dealing with the various forms of data in the database: The input data is split into four general types; diagnosis/procedure codes, drug codes, cost information, and demographic information. We construct one or more kernels for each type of data, and when necessary, select the best kernel through cross-validation. We then combine the best kernel for each type of data into an overall kernel by adding the kernels. The following sections give an overview of each type of data and the kernels that we have selected in each case.

4.3.3.1 Cost Kernel

For each claim, we have the total cost of the claim as part of our data. We wish to construct a kernel that will uses this information to aid in the prediction of the onset of diabetes.

We constructed a stratified bin kernel. The details of the *bin kernel* follow. Suppose that we have a continuous variable x. We select a number of values that act as the boundaries of our bins. Suppose that $b_0 < b_1 < \ldots < b_n$ are the boundaries of n bins. We construct a feature vector with one component per bin. The value of the ith component is 1 provided that the variable x is larger than the bin's upper limit b_i, otherwise it is set to zero. When we take the inner product of such vectors, we get a count of how many bins the two values have in common.

Here is our specific implementation of the stratified bin kernel. First, the claims were stratified into six month sections. Then the total cost for each stratum was calculated. From the extra data set, we calculated the distribution of six month costs for those patients, and noted the 10th, 20th, ..., 90th percentiles. Theses values acted as the boundaries for our eight bins. The feature vector was made by using the features from each stratum's bin kernel. The only parameters that will need tuning are γ from the Gaussian kernel and the SVM cost parameter C. This is the only cost kernel that we found to be competitive.

Our preliminary search for a cost kernel took many paths. In one example, we calculated the total cost to date for each date in the patient's history. We then fit a polynomial through these points. The polynomial was of degree six and constructed using orthogonal components. We used the coefficients of this polynomial as the feature vector and used it with various classifiers. This method didn't prove to be successful. We also tried the local time warping (LTW) kernel on the cost data, but again, this method failed to produce results.

4.3.3.2 Code and Drug Kernels

The next kernels that we constructed were designed to take the diagnosis and procedure codes or drug codes as input.

The methods that were tried include the Bag of Codes with counts (BoC),Bag of Codes with indicators (BoCI), Stratified Bag of Codes with counts (SBoC), Stratified Bag of Codes with indicators (SBoCI), Fisher, and Local Time Warping (LTW) kernels. The kernels that passed the preliminary round of testing were the BoCI and SBoCI kernels.

The BoCI only has the regular Gaussian kernel parameters that need tuning. We also needs to tune these parameters for the SBoCI kernels, as well as the width of the strata. We set the width of the strata at 6 months after preliminary testing indicated this was an acceptable value.

4.3.3.3 Code Subset Selection

There are thousands of diagnosis, drug, and procedure codes in the database, and many of them likely have little to do with diabetes. Consider, for example, a diagnosis code for the flu. While it is possible that there is some hidden connection between the flu and type 2 diabetes, reasonable to think that many such codes are unrelated and uninformative with respect to diagnosing diabetes. Therefore, a method was developed for limiting the number of codes.

The benefits are two fold. A patient history that is limited to only a select few codes is much shorter in terms of the number of observed codes. This leads to a savings in the computational cost. In the case of alignment kernels, such as the LA or LTW kernels, the computational cost goes from impractical for a full code set to feasible for a reduced code set. Also, if we are able to find a set of codes that is much smaller, say a few hundred, that predict nearly as well as the full set of codes (nearly ten thousand), we gain the ability to analyze the codes and their implications. This will be the goal of an upcoming research paper.

In the next section, we will give details on the method that we used to select a subset of codes that can be used to accurately classify diabetes and non-diabetes patients. First we give a parametric rule used to generate this subset of codes, and then we discuss how we went about selecting the parameters involved in this rule.

4.3.3.4 Subset Selection Rule

The details of our code selection method follow. We consider a classifier that makes a prediction based off on presence/absence of a single code. Such a predictor would work best if there was a large difference in the number of times the code occurred for one class versus the other. We define n_+ as the number of times a code was observed for positive examples and n_- as the number of times the code was observed for negative examples. We define absdiff $= |n_+ - n_-|$. A code is included in the subset of selected codes if absdiff $> a$, for some value of a.

Suppose that we have a code that is only present in a small number of the positive examples. Such a variable would not perform well, when used on all the examples, but it may predict a small number of positive examples with great accuracy and would be useful in conjunction with other codes. Such codes may not be selected by the absdiff criterion so we use an additional code selection criterion that captures such codes.

The *positive predictive value* (ppv) is defined as the proportion of the examples we predict as positive that are actually true positives. We calculate this value for each code, based on the extra data set for our presence/absence classifier which, in this case, predicts the positive case on presence. We also calculate the *negative predictive value* (npv), defined as proportion of the examples we predict as negative that are actually true negatives, by having our presence/absence classifier predict the negative class on presence.

We define *predictive value* as $pv = \max(npv, ppv)$. If we simply allow all codes with a high pv into the code set, the set will contain many codes that are very infrequent. For this reason, we also set a minimum number of times that a code must be seen before it can be part of the code subset. Let $total = n_+ + n_-$. Our second rule for including a code is $pv > p$ and total $> t$ for some values of p and t.

The overall rule is

$$(\text{absdiff} > a) \text{ or } (\text{pv} > p \text{ and total} > t).$$

4.3.3.5 Selection of Rule Parameters

We wish select the values for a, p, and t so that the number of resulting codes is around some user-defined threshold. We found this to be a surprisingly difficult task, and in this section we share our method for tackling this problem.

First, we illustrate the selection of a. For each code observed in the database, that code has a value of absdiff calculated using the extra data set. This collection

of values for absdiff give us all the relevant cut off values for a. For example, suppose that we have a database that only contains the codes c_1, c_2, and c_3 and suppose that absdiff for these codes is 100, 200, and 300 respectively. Let C be the set of codes that are included after the selection process. Then

$$\begin{aligned} a < 100 &\Rightarrow C = \{c_1, c_2, c_3\} \\ 100 \leq a < 200 &\Rightarrow C = \{c_2, c_3\} \\ 200 \leq a < 300 &\Rightarrow C = \{c_3\} \\ a > 300 &\Rightarrow C = \emptyset. \end{aligned}$$

Therefore, for each code we compute the absdiff value as well as the number of codes that would remain if we used this value for a.

Similarly, knowing the values for total and pv for each code gives us all the necessary information for picking values of p and t. Thus, for each code, we calculate the value of the pair (pv, total) and also the number of codes that would remain if p and t were set to these values.

Now we will discuss the method we used to arrive at the final values for a, p, and t. Suppose that we only want around 300 codes. Once the information above is collected, we search for values of a and values of the pair (p, t) that will result in between 250 and 300 codes remaining. Include all of resulting values of a in the set A and the resulting pairs (p, t) in the set P. All possible combinations of $a \in A$ and $(p, t) \in P$ are constructed, and the performance of the BoCI classifier when using this combination is estimated through cross-validation on the tuning set. We select the combination with the best performance as the final values for a, p, and t.

We followed this procedure for drug codes and the combination of the diagnosis and procedure codes. We selected a number of thresholds, and selected the ones that balanced the number of codes with the classification success. The result was a set of a little over 250 diagnosis and procedure codes and around 60 drug codes. In the

Table 4.2: Comparison of the SBoCI and LTW kernels on 1000. Both kernels use the selected codes resulting from the method discussed in Section 4.3.3.3. The balanced success rate is the average of the success rate for each class. The value given in the table is the average of five independent five-fold cross-validations, with the same set of five partitions used for each kernel. The number in the parenthesis is the SE for the associated mean.

Data Type	Balanced Success Rate	Area Under the ROC Curve
SBoCI	0.706 (0.002)	0.772 (0.001)
LTW	0.697 (0.002)	0.764 (0.001)

results section, using a kernel on either of these subsets of codes will result in the prefix "selected". Kernels that used all available codes will get the prefix "full".

4.3.3.6 Comparing the SBoCI and LTW Kernels

Many of the methods listed above did not make it out of the first round of comparison. In particular, the LTW kernel was not used after this point, but we would like to highlight its performance. Table 4.2 is the comparison of of the LTW kernel and the SBoCI kernel using multiple runs of five-fold cross-validation. You can see that the LTW kernel is competitive with the SBoCI kernel. Both of the kernels used the subset of diagnosis and procedure codes discussed in the last sections. Due to the fact that the LTW kernel requires a much greater computation time, we have decided to focus on the various "bag of codes" kernels.

4.3.3.7 Demographic Kernel

This kernel was used to incorporate the patients' demographic information. All of the variables were categorical in nature. For each variable A, a feature vector was constructed with a length equal to the number of distinct values of A. Suppose that the possible values are $\{a_1, \ldots, a_n\}$. The ith component of the vector was associated with a_i, it's value being 1 when $A = a_i$ and 0 otherwise. We then add the kernels for all the demographic variables to form the demographic kernel. The variables that are included in the kernel are age group, geographical region, employment classification of

the primary beneficiary, employment status of the primary beneficiary, relationship to the primary beneficiary, industry classification of the employee responsible for payment of the claim, and gender. The parameters that will need tuning are those associated with the Gaussian kernel, C and γ.

4.3.3.8 Combined Kernels

To get an idea of the best kernel of each type, we ran five-fold cross-validation using the tuning set as the examples. We found that the SBoCI kernel was the best kernel for both the drug codes and the combined diagnosis and procedure codes and the bin kernel was the best for the cost data.

Once we have decided the best kernel of each type, the kernels are combined into one overall kernel. We do this by first normalizing each kernel using the cosine kernel and then adding the kernels together, which is equivalent to concatenating the feature vectors. We will be making two versions of this kernel, one with the full set of codes for diagnosis, drug, and procedure codes; and a kernel using selected code subset.

4.4 Results

4.4.1 Performance of Individual Data Types

First, Table 4.3 gives the results for the individual kernels on the four types of data. ROC curves for the individual and summed kernels with the full set of codes are given in Figure 4.6. The curves for the kernels using the selected codes can be seen in Figure 4.6. We see that the type of data that has the best individual success (based on the ROC curves) was the BoCI code kernel for the diagnosis/procedure codes information, with an area under the ROC curve of 0.821 and a 74.3% success rate. The ROC curves also show that combining the kernels provides a clear improvement. The best kernel based on the full codes had an area under the ROC curve of 0.821 while the best kernel based on selected codes had an area of 0.810. Thus, the subset

Table 4.3: Results for various data types and kernels. The balanced success rate is the average of the success rate for each class. The value given in the table is the average of five independent five-fold cross-validations, with the same set of five partitions used for each kernel. The number in parentheses is the SE for the associated mean.

Data Type	Kernel	Balanced Success Rate	Area Under the ROC Curve
Cost	Stratified Bin	0.647(0.000)	0.686(0.000)
Demographics	Indicator	0.757(0.000)	0.807(0.000)
Diagnosis/ Procedure	Full Codes BoCI	0.743(0.000)	0.821(0.000)
	Full Codes SBoCI	0.743(0.000)	0.818(0.000)
	Selected Codes BoCI	0.731(0.000)	0.809(0.000)
	Selected Codes SBoCI	0.733(0.000)	0.810(0.000)
Drug	Full Codes BoCI	0.711(0.000)	0.759(0.000)
	Full Codes SBoCI	0.711(0.000)	0.764(0.000)
	Selected Codes BoCI	0.706(0.000)	0.759(0.000)
	Selected Codes SBoCI	0.708(0.000)	0.758(0.000)

of codes leads to nearly the same results as using the full codes, particular when comparing the summed kernels.

4.4.2 Combined Kernels - Full and Selected Codes

Next, we give the performance for the combined kernels, both for the full set of codes as well as using the partial code set. Results can be seen in the table in Figure 4.6. Notice that the process of combining the kernels leads to nearly a 3% gain in success rate, going from 75.7% for the best individual kernel to 78.5% for the combined kernel using the full code set. Thus, adding the kernels was successful in combining the different forms of information into an even better kernel.

We also see that the kernel that uses the partial set of codes did very nearly as well as the kernel that used the full set of codes, with the full code set resulting in an area under the ROC curve of 0.865 and the selected codes having an area of 0.864. The benefit of this fact is that we have shown that a manageable number of codes can be used for nearly identical prediction performance. We will look into a specific analysis of these codes in a future paper.

Recall that the positive (negative) predictive value of a classifier is the proportion of the values that our classifier labels as positive (negative) that are truly positive (negative) examples. The sensitivity of the classifier is the proportion of positive example that are labeled correctly. The specificity is the proportion of negative examples that are labeled correctly. For the classifier based on the combined kernels with a full set of codes, we get a positive predictive value of 0.743, a negative predictive value of 0.841, a sensitivity of 0.867, and a specificity of 0.701. These values are nearly identical for the classifier using combined kernels for the selected codes.

4.4.3 Combined Kernels - Complete and Truncated Claim Windows

Finally, we compare the results of combined kernels when using a patient's entire claim window to using a truncated claim window. We used the BoCI kernel, but not the SBoCI kernel. The reason for this is simple: We cannot guarantee that the truncated patient histories will fill a full three years.

The stratified kernel only gives fair comparisons when the patients being compared have a full history over all strata. If we were to compare a patient with 2 years of history with a patient with three years of history, two of the strata would lead to inner products of 0, indicating that these strata were from patients that were "orthogonal" to each other. The problem is that this sort of orthogonality is justified only by the different sized claim windows, not by virtue of any observed difference in the codes.

Recall that we considered two versions of truncated claim windows, one with an average of *12 month* removed and another with an average of *18 months* removed. The results are available in Table 4.6. The best kernel using the entire claims window had an area under the ROC curve of 0.867. The best kernel for the 12 month truncated claims windows had an area under the ROC curve of 0.859. Finally, the best kernel for the 18 month truncated claims windows had an area under the ROC curve of 0.845. Here we see our most profound result: Even after removing an average of 18

months of a patients claim history, we are still able to classify the patients nearly as well. These results are promising in that our ability to successfully classify the patients is nearly as good over a year before the first diabetes diagnosis as it is the day before the diagnosis.

4.5 Conclusion

We have shown that medical claims data can be used to provide good results when classifying positive examples (patients with their first diabetes diagnosis in our claim window) vs. negative examples (patients with no indication of diabetes in the claim window). In this study, claims data was the only form of information used; which, to the best of our knowledge, has not been tried before.

We were able to reduce the number of diagnosis, drug, and procedure codes to a manageable number while retaining performance that was comparable to the best kernel using the full set of codes. Most importantly, we showed that these data allow us to identify patients that will be diagnosed with type 2 diabetes in the next 15 to 18 months with reasonable success.

Diabetes is a costly and prevalent disease. It is somewhat unique in that it has a pre-disease state that can be detected before the onset of the disease. Overall, we see these methods as constituting a first step in an intervention program. A diabetes intervention program would give incentives to participants to lower their risk of type 2 diabetes through moderate lifestyle changes. Such programs have shown success in delaying or completely preventing the onset of type 2 diabetes. The one thing that remained was a cost effective method for identifying candidates for an intervention. The methods presented here could be used as a very cheap method of identifying possible candidates for intervention, provided that the patient's insurance claims data is available. Persons identified as "positive" could then be tested for diabetes or pre-diabetes using more expensive medical tests. The power of such a program is that

we will be able to detect people at risk of type 2 diabetes, most of whom won't get tested until it is too late.

4.6 Future Work

In a future paper, we will specifically analyze the codes that resulted from our codes subsetting method. Our hope is to provide supporting evidence for codes that are known to be related to diabetes, and perhaps provide new insight by identifying codes that would not be intuitively associated with diabetes.

In the construction of our data set, we identified diabetes patients as enrollees with one or more diabetes diagnosis. It must be noted that the presence of a diabetes diagnosis does not necessarily mean that the patient has diabetes. There have been studies that looked at the ability of insurance claims to predict *current* diabetes patients[9] through claims data. At least one [38] found that using a rule of 2+ diabetes diagnoses was a better rule for predicting diabetes than a rule of 1+ diabetes diagnoses. It may be worthwhile to redo our experiments using a different definition of a positive example and compare the results.

[9]This is in contrast to our goal of identifying people that will be at a risk of diabetes in the future.

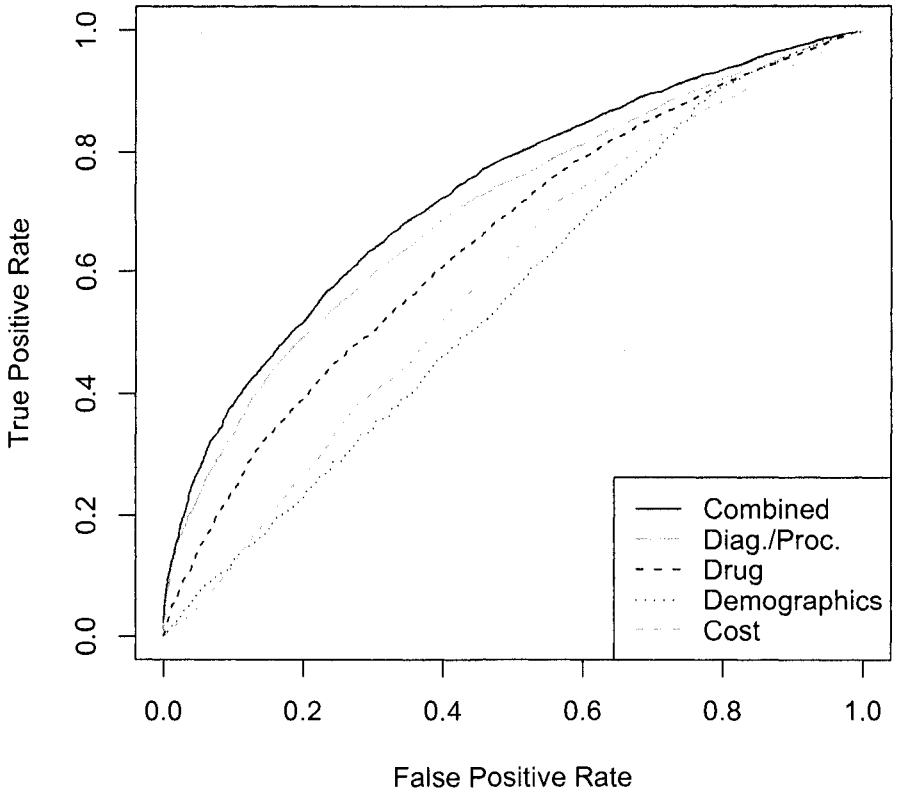


Figure 4.2: ROC curves for various kernels: full codes.

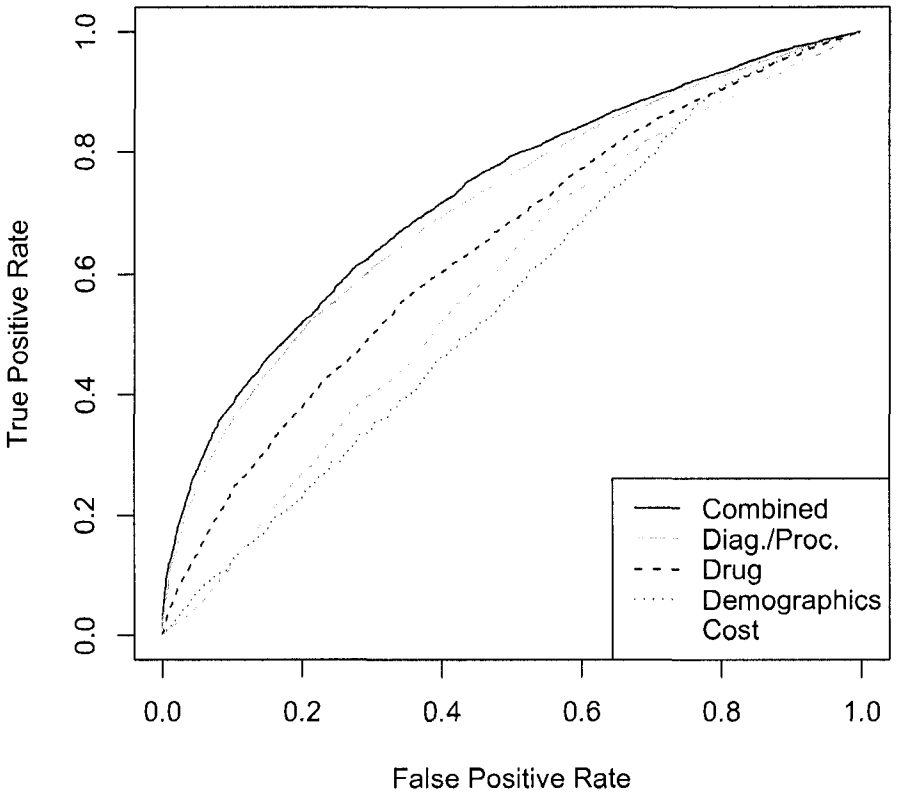
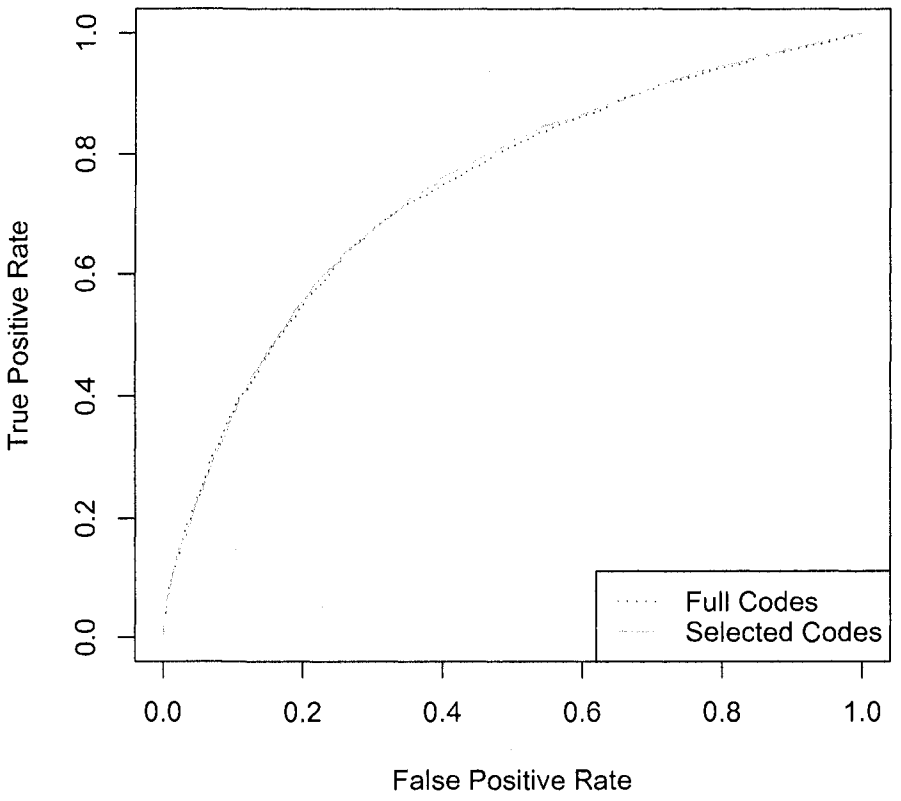
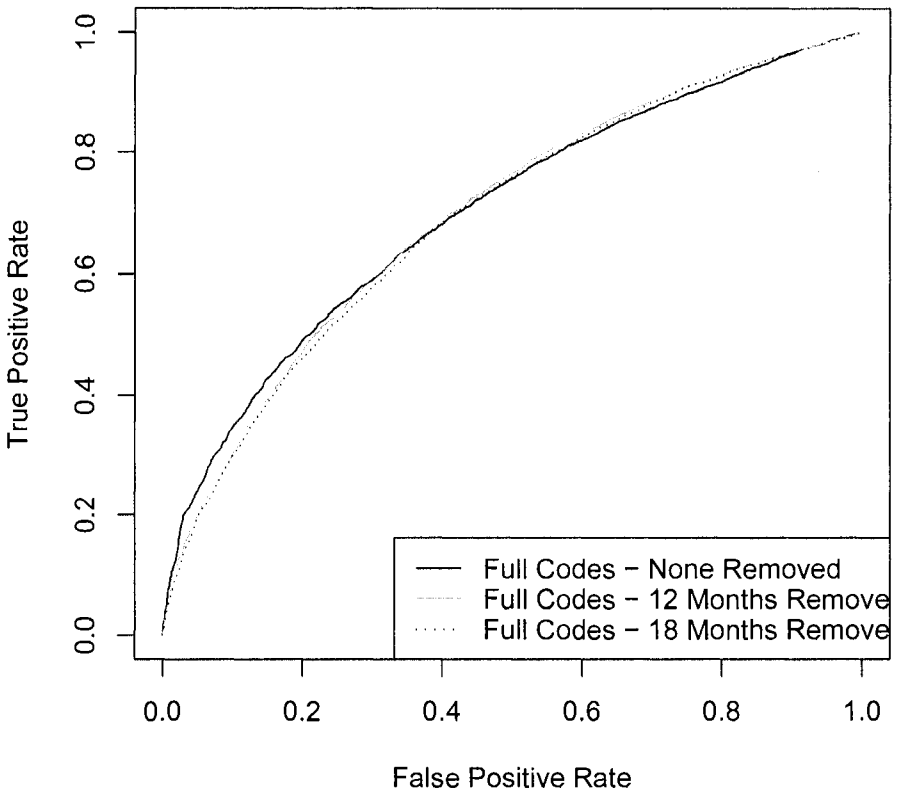


Figure 4.3: ROC curves for various kernels: selected codes.



Data Type	Balanced Success Rate	Area Under the ROC Curve
Full Codes	0.785(0.000)	0.865(0.000)
Selected Codes	0.782(0.000)	0.864(0.000)

Figure 4.4: Comparing kernels for full and selected codes. PLOT: ROC curves for the two kernels. TABLE: Results for sum of the kernels for different data types. The Full Codes results are using all the available diagnosis, drug, and procedure codes. The Partial Codes results only use the codes resulting from the code subsetting discussed in Section 4.3.3.3. The balanced success rate is the average of the success rate for each class. The value given in the table is the average of five independent five-fold cross-validations, with the same set of five partitions used for each kernel. The number in parentheses is the SE for the associated mean.



Code Set	Average Amount Removed	Balanced Success Rate	Area Under the ROC Curve
Full Codes	None	0.784 (0.000)	0.867 (0.000)
Selected Codes	None	0.781 (0.000)	0.865 (0.000)
Full Codes	12 Month	0.771 (0.000)	0.848 (0.000)
Selected Codes	12 Month	0.784 (0.000)	0.859 (0.000)
Full Codes	18 Month	0.770 (0.000)	0.845 (0.000)
Selected Codes	18 Month	0.768 (0.000)	0.844 (0.000)

Figure 4.5: Results for the truncated data sets. PLOT: ROC plot for each of the full code kernels. The results for the selected codes were nearly identical. TABLE: The Full Codes results are using all the available diagnosis, drug, and procedure codes. The Select Codes results only use the codes resulting from the code subsetting discussed in Section 4.3.3.3. The 12 month and 18 month examples had a random amount of their most recent history removed that averaged out at 12 months and 18 months respectively. We have included the results on the full histories using the BoCI code kernel for comparison.

Chapter 5

ASSOCIATION RULES

In contrast to the last chapter, the current chapter is concerned with unsupervised learning using the MarketScan® database. Here we are required to extract *interesting* and *useful* patterns from the database. A standard approach for unsupervised learning on categorical data involves the mining of association rules [44]. We first review a commonly used algorithm, called the *Apriori* algorithm [45], for extraction of rules. In this section, we also review the concept of closed rules and discuss some recent results from the literature. We then state and prove a new theorem which is particularly useful when considering mining for rules according to which a set of diagnoses imply a set of procedures. Additionally, we introduce a class of *interesting rules* that take into account costs associated with various procedures prescribed based on available diagnoses for a patient. We provide examples of such interesting rules by mining the MarketScan® database.

5.1 Association Rules: The Apriori Approach

Association rules were first introduced in relation to the market basket analysis problem. Let $\mathcal{I}$ denote the collection of all items sold at a supermarket. Suppose the supermarket has stored records of items purchased by customers. Each transaction is a subsets of $\mathcal{I}$ and has an identification number y. The set of all transaction identification numbers is denoted by $\mathcal{T}$. Thus, saying $T_y = \{x_{y_1}, \ldots, x_{y_{n_y}}\}$ indicates that T_y (transaction with identification tag y) was associated with the purchase of the n_y items $x_{y_1}, \ldots, x_{y_{n_y}} \in \mathcal{I}$.

Our goal is to find rules of the form $A \stackrel{q}{\Rightarrow} B$, where A, B are disjoint subsets of $\mathcal{I}$ and $A \cup B \subseteq T_y$ for some $y \in \mathcal{T}$. That is, the elements of A and B are common to at least one transaction T_y.

Association rules typically have two attributes, the *support* and *confidence*. The support is defined as the number of transactions that contain the items in $A \cup B$. The confidence q is the probability that a transaction will contain B given that it contains A.

Let's look at the total number of possible rules. For any subset of $\mathcal{I}$ with k elements, there are $2^k - 2$ possible rules. This comes from selecting all possible left-hand-sides of the rule except the null set and the full set. Suppose that there are I unique elements in $\mathcal{I}$. The total possible number of rules will be

$$\sum_{k=2}^{I} \binom{I}{k} \left(2^k - 2\right)$$

For 20 unique items, this is already just shy of 3.5 billion rules. This is far too many rules to mine from even a moderate database. We must apply some constraints to limit the size of the class of rules we wish to explore.

Consider using constraints on the support of the item set to limit the number of possible solutions. Zaki et. al. [46] point out the relationship between finding frequent item sets and finding constrained bipartite cliques, a problem that has been well studied. Thus finding all frequent sets at or above some threshold for support K can be solved in polynomial time, but finding all frequent item sets with support of exactly K is NP-complete [10]. For this reason, the Apriori algorithm and other similar algorithms force the support of a rule to be above a preset *minsup* (which stands for minimum support). Itemsets with a support greater than *minsup* are called *frequent* itemsets.

---

[10]Meaning it is a hard problem for which there no known polynomial time solution.

The confidence will need to be above *minconf* (*minimum confidence*), which will further limit the size of the set of resulting rules. Limiting the confidence of the rules is also an example of defining rules that are "interesting".

The Apriori algorithm was the first major breakthrough in mining association rules. The search is divided into two parts. First, it uses a branch-and-bound approach to search the space of itemsets for frequent itemsets. We first look at all frequent item set of cardinality 1, then all frequent item sets of cardinality 2, etc. Next association rules are created for these frequent itemsets. The first portion dominates the computation time, and as such is the focus of much of the discussion.

To see how the Apriori algorithm works, notice that any subset of a frequent itemset must also be frequent. Taking advantage of this fact, we start with small itemsets and eliminate infrequent sets. No subset containing these eliminated sets need be searched, thus we limit our search space.

Table 5.1: Example of a transactional database

tid	items
1	a, c
2	a, b
3	a, b, c
4	a, b
5	a, b, c
6	a, c
7	c
8	c, d

Consider the database in Table 5.1. There are 8 transaction and four items. Suppose that we set $minsup = 4$. We start with the individual items a, b, c, d and scan the database to count their frequency. The only item that is not frequent (happens less than 4 times) is d, which we eliminate. This leaves a, b, c. We now form all pairs of frequent items, namely ab, ac, bc and scan the database to count their frequency. The itemset bc is not frequent and is removed leaving ab, ac. Again, all combinations

of frequent itemsets from the last set are generated and checked. In this case the procedure leads to abc which we know is infrequent because bc is infrequent. There is nothing left to check and the algorithm terminates. Many variations on the Apriori algorithm exist. These variations focus on clever ways of storing and checking the frequency of the itemsets.

There are a few problems with Apriori. First, we may miss low support items that are of interest. For example, as stated on [6, page 444], customers that purchase vodka may also purchase caviar. As these purchases happen infrequently, they will most likely be missed by an Apriori-like algorithm.

Furthermore, high confidence does not translate to a dependence relationship. For example, suppose that A and B are independent and frequent, and B occurs in 80% of the transactions. The confidence of the rule $A \overset{0.8}{\Rightarrow} B$, $P(B|A) = P(B)$, is high. This may be thought to indicate a predictive association between A and B, which is incorrect. Other metrics have been suggested to solve this problem. Many of these metrics attempt to measure the departure from independence in one way or another.

5.1.1 Mining Rules Using Closed Frequent Itemsets

Next, we review an appealing way, first introduced by Zaki and Hsiao [47], in which one can shrink the search space and eliminate redundant rules. Suppose that the items $a, b, c \in \mathcal{I}$ always occur together and $D \subseteq \mathcal{I}$ not containing a, b, c. The following rules would form a redundant set of rules in the sense that they are guaranteed to have the same support and confidence.

$$\begin{aligned} \{a\} &\overset{q}{\Rightarrow} D \\ \{a, b\} &\overset{q}{\Rightarrow} D \\ \{a, c\} &\overset{q}{\Rightarrow} D \\ \{b, c\} &\overset{q}{\Rightarrow} D \\ \{a, b, c\} &\overset{q}{\Rightarrow} D \end{aligned}$$

In this example, the set $\{a, b, c\}$ is a *closed* set. We will now give some results from Formal Concept Analysis that will provide a definition for a closed set. We follow closely the notation and development given in [46]. The development starts with the definition of a partially ordered set and a lattice.

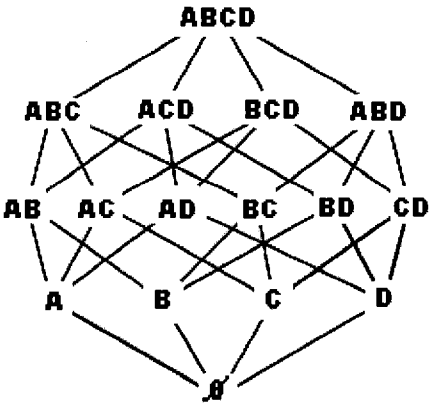


Figure 5.1: The complete lattice of itemsets. This is the lattice consisting of all subsets of our set of items. The meet for two items can be found by following lines down to a common vertex. The join can be found by following lines up to a common vertex. Notice that the meet and join exists for all pairs of items, making it a lattice. The definition of *complete* is given in the main body of text.

A *partial order* on a set S is a binary relation $\leq$ with the following properties:

Reflexive : $x \leq x, \forall x \in S$.
Anti-Symmetric: $x \leq y$ and $y \leq x, \Rightarrow x = y, \forall x, y \in S$.
Transitive : $x \leq y$ and $y \leq z, \Rightarrow x \leq z, \forall x, y, z \in S$.

Let $(S; \leq)$ be a partially ordered set and $A \subseteq S$. Define the following:

Upper Bound of A: $u \in S$ such that $a \leq u$ for all $a \in A$.
Lower Bound of A: $l \in S$ such that $l \leq a$ for all $a \in A$.
Meet of A: The greatest lower bound, denoted $\bigwedge A$.
Join of A: The least upper bound, denoted $\bigvee A$.

A partially ordered set $(L, \leq)$ is a *lattice*, provided that for any two elements x and y in L, the join $x \vee y$ and meet $x \wedge y$ always exist. L is a complete lattice if

$\bigwedge A$ and $\bigvee A$ exist for all $A \in L$. The partially ordered set $(\mathcal{P}(S), \subseteq)$ is a *complete lattice.* Here $\mathcal{P}$ denotes the set of subsets of P (power set). The complete lattice for the itemsets in the database in Table 5.1 is shown in Figure 5.1. Sets that are higher in the lattice are "larger" than the sets below them according to the ordering defined by $\subseteq$. The meet for two items can be found by following lines down to a common vertex. The join can be found by following lines up to a common vertex. The sets AB and BCD meet at B and join at $ABCD$, their intersections and unions respectively.

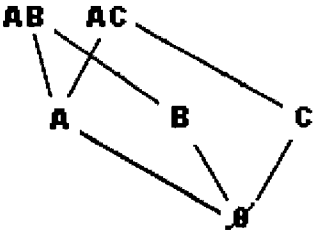


Figure 5.2: The meet semi-lattice of frequent itemsets. This semi-lattice consists of all frequent subsets ($minsup = 4$) from our example database. While each pair of items has a meet, there are some items, for example B and C that don't have a frequent join.

If we are only guaranteed that joins will exist, then L is called a *join semi-lattice.* If only meets are guaranteed to exist, then L is called a *meet semi-lattice.* Figure 5.2 shows all frequent itemsets for our example database, which form a meet semi-lattice. Notice that for any two items, there is a common vertex below the pair, but not necessarily a vertex above above the pair. For example, B and C meet at $\emptyset$ but their join, BC is not frequent. Here is the consequence of the fact that frequent itemsets form a meet semi-lattice: While we may search the lattice by starting with the (single) smallest item and moving up the lattice, we will not be able to search the lattice by starting at the one largest item and moving down.

Now consider the connection between the complete lattice of itemsets $(\mathcal{I}, \subseteq)$ and the complete lattice of tidsets (transaction identification number sets) $(\mathcal{T}, \subseteq)$. Let $X \subseteq \mathcal{I}$ be an itemset and $Y \subseteq \mathcal{T}$ be a tidset.

First, define functions that map from one lattice to the other.

$$t : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{T}); \qquad t(X) = \left\{ y \in \mathcal{T} \middle| \forall x \in X; x\delta y \right\}$$
$$i : \mathcal{P}(\mathcal{T}) \rightarrow \mathcal{P}(\mathcal{I}); \qquad i(Y) = \left\{ x \in \mathcal{I} \middle| \forall y \in Y; x\delta y \right\}$$

The notation $x\delta y$ means that x and y are "related", that is, item x occurs in transaction y ($x \in T_y$). For example, in the database in Table 5.1, we can write $a\delta 1$ as item a occurs in transaction 1.

The function i applied to Y gives all the items that occur together in every transaction in the tidset Y and the function t gives all the transactions that contain every item in the itemset X. It is obvious that $t(X) = \bigcap_{x \in X} t(x)$.

As an illustration, consider the database in Table 5.1. One can see that $t(ab) = 2345$ and $i(12) = a$.

These functions form a *Galois connection.* A Galois connection must satisfy the following conditions:

(i) $X_1 \subseteq X_2 \Rightarrow t(X_1) \supseteq t(X_2)$.
(ii) $Y_1 \subseteq Y_2 \Rightarrow i(Y_1) \supseteq i(Y_2)$.
(iii) $X \subseteq i(t(X))$ and $Y \subseteq t(i(Y))$.

The following proposition is easily proved.

Proposition 5.1.1 *The functions $t : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{T})$ and $i : \mathcal{P}(\mathcal{T}) \rightarrow \mathcal{P}(\mathcal{I})$ form a Galois connection between $(\mathcal{I}, \subseteq)$ and $(\mathcal{T}, \subseteq)$.*

The utility of a Galois connection is that it gives us an easy way to form two closure operators. A function $c : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ is a *closure operator* on S if, for all $X, Y \subseteq S$, c satisfies the following properties:

(i) Extension: $X \subseteq c(X)$.
(ii) Monotonicity: if $X \subseteq Y$, then $c(X) \subseteq c(Y)$.
(iii) Idempotency: $c(c(X)) = c(X)$.

A subset X of S is *closed* if $c(X) = X$.

Proposition 5.1.2 *Let $X \subseteq \mathcal{I}$ and $Y \subseteq T$. Let $c_{it}(X) = i(t(X))$ and $c_{ti}(Y) = t(i(Y))$. Then $c_{it} : \mathcal{P}(I) \rightarrow \mathcal{P}(I)$ and $c_{ti} : \mathcal{P}(T) \rightarrow \mathcal{P}(T)$ are both closure operators on itemsets and tidsets respectively.*

Define a closed itemset as an itemset X such that $X = c_{it}(X)$. For any closed itemset X, there exists a closed tidset given by Y, with the property that $Y = t(X)$ and $X = i(Y)$. Such a pair (X, Y) is called a *concept*. It turns out that the concepts form a lattice with meets and joins given in [47]. This leads us to the two main results from Zaki and Hsiao [47].

Theorem 5.1.1 *For any itemset X, its support is equal to the support of its closure, i.e., $|t(X)| = |t(c_{it}(X))|$.*

Theorem 5.1.2 *The rule $X_1 \overset{p}{\Rightarrow} X_2$ is equivalent to the rule $c_{it}(X_1) \overset{q}{\Rightarrow} c_{it}(X_2)$ in the sense the rules share the same support and $q = p$.*

The second theorem tells us that when searching for rules based on frequent itemsets, it suffices to find rules based on frequent closed itemsets.

The big advantage of searching for closed itemsets is that one is searching over a smaller lattice, and hence can potentially get a much faster rule-mining algorithm. Furthermore, the list of association rules is smaller and does not contain redundant rules. We will see that there is even more benefit to looking for association rules between closed sets in a medical database.

5.1.2 Closed Frequent Itemsets and Medical Databases

Suppose that our itemset $\mathcal{I}$ is partitioned into diagnoses $\mathcal{D}$ and procedures $\mathcal{P}$. We wish to find rules of the form $D \stackrel{p}{\Rightarrow} P$, where $D \subseteq \mathcal{D}$ and $P \subseteq \mathcal{P}$.

We will define two new functions that map from the transaction sets to each partition.

$$i_{\mathcal{D}} : \mathcal{P}(\mathcal{T}) \rightarrow \mathcal{P}(\mathcal{D}); \qquad i_{\mathcal{D}}(Y) = \left\{x \in \mathcal{D} \middle| \forall y \in Y; x\delta y\right\}$$
$$i_{\mathcal{P}} : \mathcal{P}(\mathcal{T}) \rightarrow \mathcal{P}(\mathcal{P}); \qquad i_{\mathcal{P}}(Y) = \left\{x \in \mathcal{P} \middle| \forall y \in Y; x\delta y\right\}$$

If we ignore the items in $\mathcal{P}$, the results of the previous subsection imply that $i_{\mathcal{D}}$ and t form a Galois connection between the lattices $(\mathcal{P}(\mathcal{D}), \subseteq)$ and $(\mathcal{P}(\mathcal{I}), \subseteq)$, and similarly $i_{\mathcal{P}}$ and t form a Galois connection between the lattices $(\mathcal{P}(\mathcal{P}), \subseteq)$ and $(\mathcal{P}(\mathcal{I}), \subseteq)$. Therefore, the functions $c_{i_{\mathcal{D}}t}(X) = i_{\mathcal{D}}(t(X))$ and $c_{ti_{\mathcal{D}}}(Y) = t(i_{\mathcal{D}}(Y)$ are closure operators. Similarly, the functions $c_{i_{\mathcal{P}}t}(X) = i_{\mathcal{P}}(t(X))$ and $c_{ti_{\mathcal{P}}}(Y) = t(i_{\mathcal{P}}(Y)$ are closure operators. Therefore, methods described in Zaki and Hsiao [47] can be used to prove the following extention to Theorem 5.1.2.

Theorem 5.1.3 *The rule $X_1 \stackrel{p}{\Rightarrow} X_2$ is equivalent to the rule $c_{i_{\mathcal{D}}t}(X_1) \stackrel{q}{\Rightarrow} c_{i_{\mathcal{P}}t}(X_2)$ in the sense that the rules share the same support and $q = p$.*

Proof: Note that we can find the support of an itemset X by applying t and counting the number of items in the resulting set, i.e. $|t(X)|$. Another important fact is that the support of a set is the same as the support of its closure. Thus

$$|t(X_1 \cup X_2)| = |t(X_1) \cap t(X_2)| = |t(c_{i_{\mathcal{D}}t}(X_1)) \cap t(c_{i_{\mathcal{P}}t}(X_2))| = |t(c_{i_{\mathcal{D}}t}(X_1) \cup c_{i_{\mathcal{P}}t}(X_2))|$$

and

$$\frac{|t(X_1 \cup X_2)|}{|t(X_1)|} = \frac{|t(X_1) \cap t(X_2)|}{|t(X_1)|} = \frac{|t(c_{i_{\mathcal{D}}t}(X_1)) \cap t(c_{i_{\mathcal{P}}t}(X_2))|}{|t(X_1)|} = \frac{|t(c_{i_{\mathcal{D}}t}(X_1) \cup c_{i_{\mathcal{P}}t}(X_2))|}{|t(c_{i_{\mathcal{D}}t}(X_1))|}.$$

Note that this theorem states that to find rules of the form $D \stackrel{P}{\Rightarrow} P$, it suffices to find rules from closed frequent diagnoses itemsets (closed with respect to the database containing only diagnoses) to frequent procedure itemsets (closed with respect to the database containing only procedures). This result has the possibility of reducing the search by a significant margin. If the length of the longest frequent diagnoses or the longest frequent procedure is much smaller than the length of the longest combined frequent itemset, we should see a major boost to the speed of the algorithm. For example, in the MarketScan® database the length of the longest diagnoses and procedure has size 15 for combined in-patient claims. On the other hand, the longest itemset has size 30. See Section 5.3 for a new proposed algorithm based on this theorem.

5.1.3 Apriori-like Algorithms and Medical Databases

Now suppose that the database has a row for each insurance claim. To apply the methods developed for market basket analysis (MBA), we first need to define what we mean by items and transactions. Let $\mathcal{D}$ denote the set of possible diagnoses and $\mathcal{P}$ the set of possible procedures that could be prescribed. These will take the place of items in MBA.

There are a couple of ways to define the "transaction". Suppose that we consider a patients claims history as the transaction. This is the approach taken by Doddi et al [48] as well as Ordonez et. al. [49]. The work by Doddi et. al. resembles our work in that it only considers rules generated from insurance claims data. Ordonez et. al. consider finding rules for medical data that is not limited to insurance claims, but includes clinical data, such as weight, age and measurements estimating the health of certain regions of the heart.

Using a patients history as a transaction allows for rules that span time and could suggest the progression of a medical condition. They do not necessarily tell us

about the decision making process of medical professionals, as two related codes need not even involve the same doctor.

It was pointed out in Ordonez et. al. [49] that many rules that are generated by the standard association rules software are not interesting. These uninteresting rules come in many forms: effect predicts cause, well-known medical facts, rules with very high confidence, etc. [49]. The authors dealt with this problem by reworking the Apriori algorithm to eliminate uninteresting rules. The downside to their approach is that groups of codes that lead to uninteresting rules must be defined in advance by an expert in the field. We have also suggested a possible solution, specifically to the problem of rules of the form *effect implies cause*, by only searching for rules of the form *diagnosis implies procedure* using Theorem 5.1.2.

The approach that we suggest in the next section differs from this previous work in two ways: We will consider an alternate definition of a transaction, namely a transaction is a patient claim. When a transaction is an insurance claim, the procedures and diagnoses are directly related, that is the prescribed procedures are a direct result of the diagnoses and a doctor's decision. Thus the *doctor* provides the link between diagnoses and procedures, since it was the doctors decision to use these procedures based on the diagnoses.

Finally, we would like to point out that we may learn more about the doctors decision making process by considering groups of rules. For each claim, say claim number y in the database, we have a record of the patient's diagnoses $d_{y_1}, \ldots, d_{y_{n_y}} \in \mathcal{D}$ and the procedures $p_1, \ldots, p_{y_{m_y}} \in \mathcal{P}$ that were performed.

Consider rules of the form $D \stackrel{p}{\Rightarrow} P$ where $D \subseteq \mathcal{D}$ and $P \subseteq \mathcal{P}$. These are natural rules to consider, as they represent a frequent and high probability decision that patients with diagnoses D would receive procedure P. For example, suppose we have the rule $\{d_1\} \stackrel{.7}{\Rightarrow} p_1$, and this rule is frequent. What have we learned? There is a high chance that a patient with diagnosis d_1 will be prescribed procedure p_1.

We believe there is more to the story than one rule can tell. If we instead knew that $\{d_1, d_2\} \overset{.3}{\Rightarrow} p_2$ and $\{d_1 \cap d_2^C\} \overset{.7}{\Rightarrow} p_1$, we are in a much better position. We now know that the common procedure for d_1 is p_1 except in the presence of d_2, which leads to procedure p_2. In this light, our reformulation of the problem in the next section will involve finding interesting collection of rules.

5.2 A New Objective Related to Cost

Let's reconsider this problem. If we are interested in finding connections between diagnoses and procedures then we should be interested in rules of the form $D \overset{p}{\Rightarrow} P$, where $D \subseteq \mathcal{D}$ and $P \subseteq \mathcal{P}$. A few points need to be made. Use of rules where $D \subseteq \mathcal{D}$ should, for the most part, give us a general idea of which diagnoses are related to a procedure or set of procedures. They will not give us specific information on the decisions that doctors make on a patient by patient basis.

On the other hand, suppose that a group of patients all have a claim with the same set of diagnoses D, but we observed different procedures. Some natural questions arise: Why did the same diagnoses result in different procedures? Did any of these procedures produce better results than others? Are there two procedures that produced similar results, but one costs less? This could be the starting place for some very interesting and fruitful investigation.

The cost of health-care is of major concern. Association rules in their usual form do not allow for constraints involving individual, or in particular, total cost of either side of the rule. In a medical database, we have the cost information available. It would be useful to produce a way of restricting rules based on their individual or total cost as well as their frequency.

5.2.1 Problem Formulation - Interesting Rules

We suggest the following reformulation of the definition of interesting rules:

Problem We wish to find a family of rules $D \overset{q_1}{\Rightarrow} P_1, D \overset{q_2}{\Rightarrow} P_2, \ldots, D \overset{q_{n_D}}{\Rightarrow} P_{n_D}$, called a *diagnoses family of rules* where:

1. For each i, D and P_i are the entire set of diagnoses and procedures for a group of patients.

2. We redefine the *support* of the rule $D \stackrel{q_i}{\Rightarrow} P_i$ as the number of patients that have D and P_i as their entire set of diagnoses and procedures.

3. We redefine the *confidence*, q_i, of the rule $D \stackrel{q_i}{\Rightarrow} P_i$ as the fraction of patients with diagnoses D that received procedure P_i.

4. We will compute the average cost for patients with diagnoses D and procedures P_i.

5. The total cost of diagnoses D is the sum of the costs for all patients with diagnoses D.

There are a number of ways that we can explore interesting diagnosis families. First, if there is a particular procedure of interest; we can find the diagnosis families that involve it by taking a "round-trip". First, we find all the diagnoses that lead to the procedure of interest. Then, we construct the diagnosis family for each unique diagnosis. This is done by collecting all the procedures that resulted from the diagnosis.

An alternate way to find interesting diagnosis families is to define an "interesting" criteria. For example, a diagnoses family of rules could be deemed interesting if

1. One of the rules in the family has a high total cost. In our exploratory work, the top 1000 total cost diagnoses are considered to have high total cost.

2. There is a high amount of variability in the average cost of rules in the family. To find families with high variability, the top 1000 total cost families are sorted by their average cost variability.

As a result, we will find collections of diagnoses and procedures that greatly contribute to high health care costs, but for one reason or another lead to a different procedures with different cost structures. It should be very interesting to study the reasons for the variability in cost of procedures.

5.2.2 Implementation

Algorithmically, this problem is not as difficult as the original formulation of association rule mining. We are not searching the space of subsets, but instead we are only looking at sets that occur in totality. This a much smaller set of items. Even on a database the size of the Market Scan database, the search for individual rules can be done with SAS SQL in a relatively short amount of time.

Next, we will illustrate the application of this new definition of association rules by giving two examples found in the MarketScan® database.

5.2.3 First Example Family

In this example we considered the 2003 MarketScan® database information on inpatient claims. The database consolidates all the claims for one visit to the hospital onto one line in a table. This line includes up to 15 diagnoses and 15 procedures, as well as some demographic information.

We selected the top 1000 diagnoses in terms of maximum total cost and sorted these by the variability in the average cost of the family of procedures resulting from each diagnosis. The example that we show here is ranked 38th in variability of average cost of procedures. The two diagnoses are *Intermediate Coronary Syndrome* and *Chest Pain.* There were 351 different procedures that resulted from this diagnosis.

A good question is: Can the variability be explained by the demographic information that we have. Let's attempt to answer this question. First we will look at the average cost split by each demographic category. The standard error for each average is also included. This can be used when determining if two costs are "different". The total number of cases and the standard deviation have been provided. We can use the standard deviation to look for specific classes were there is a highly variable cost. All the numbers have been rounded to three significant digits.

The family included two cases that had extreme relative costs. These two cases have a cost that is an order of magnitude larger than all other cases. It is unclear why

these two cases cost so much more, but it seems unlikely that it is a data entry error. The averages calculated after excluding these two patients is given in parenthesis. We will use these trimmed mean instead of the regular mean in our analysis.

Some obvious things to note: Table 5.2.3 indicates that this diagnosis is more expensive for men than women. Excluding the two extremes, Table 5.2.3 shows that the cost tends to rise as the patient age increases. Not surprisingly, we see in Table 5.2.3 that the cost also increases with the length of stay. Once the two extremes are excluded, the other demographic variables at our disposal showed little difference in cost between categories.

It seems strange that out of 359 patients with this diagnoses, there were 351 different sets of procedures performed. It might be the case that a number of procedure sets are only superficially different, but this would require more specific domain knowledge than is currently available to me. This is one place where medical experts can enhance the data-mining exercise by providing suitable metrics for similarity between diagnoses and/or procedures.

Table 5.2: Mean cost by gender for first example family. Note that there were two cases with unusually high cost. We provide the trimmed mean cost that excludes these two cases in parentheses.

Gender	Mean Cost	SE	Number	Standard Deviation
Male	15700(13500)	1750	237	27000
Female	9810	687	122	7590

5.2.4 Second Example Family

The next example family was picked with the desire to find a particular type of family. The desire was to find a parsimonious example where the diagnosis had a large number of occurrences, but a relatively small number of procedures. [11]

[11] As a side note, it was a little shocking how difficult it was to find this type of example. Typically, there were nearly as many different procedures as there were patients for a given set of diagnoses.

Table 5.3: Mean cost by age group for first example family. Note that there were two cases with unusually high cost. We provide the trimmed mean cost that excludes these two cases in parentheses.

Age Group	Mean Cost	SE	Number	Standard Deviation
18-34	7240	N/A	1	N/A
35-44	19600(9380)	10300	24	50400
45-54	13800(11800)	2190	142	26000
55-64	13000	859	191	11900
65 and older	5270	N/A	1	N/A

Table 5.4: Mean cost by length of stay for first example family. Note that there were two cases with unusually high cost. We provide the trimmed mean cost that excludes these two cases in parentheses.

Days	Mean Cost	SE	Number	Standard Deviation
1	9440	552	160	6980
2	15800(13400)	2630	118	28600
3	17800(12900)	5070	49	35500
4	18200	3510	27	18200
5+	37400	9160	5	22000

This family ranked 212th in total variability among the top 1000 maximum total cost families. There are 7 cases of the diagnosis family, which involves *Digestive System Complications (ICD-9-CM code 278.01)*, *Respiratory Distress (ICD-9-CM code 786.09)*, and *Localized Adiposity (ICD-9-CM 997.4)*. It is interesting that these cases all involved the same doctor and patients with very few differences. One of the patients was charged a total of just over $110,000, where as all the other patients were charged close to $14,000. All of the procedures occurred between 02/03/2003 and 04/24/2003.

First thing to note, is that there appear to be a large number of mistaken entries in the total cost throughout the database. These are usually indicated by a total cost that is an order of magnitude above the typical cost, but having a net cost that is of the same magnitude as the typical cost. This difference is attributed to a data entry error in which an extra decimal is added to the total cost. This is not the case in

this example. Both the net and total cost are one order of magnitude larger for the high-cost patient than for the typical patient.

Let's look at similarities and differences between the patient with the high cost and the typical patients.

- All of the low-cost patients were treated in hospitals with the same zip-code (perhaps the same hospital). The high-cost patient received the procedure in a hospital with a different, but very similar, zip-code.
- All of the patients had similar zip-codes and thus from nearly the same place.
- All of the patients were discharged with the same status, *Discharged to home self-care.*
- The length of stay for the high cost patient was 2 days, which was 1 day shorter than the other patients.
- Five of the patients had a comprehensive insurance plan and the other two, including the high-cost patient, had a PPO plan.
- The age of the high-cost patient was 46. The ages of the rest of the patients were 31, 47, 53, 54, 54, 58.
- The high-cost patient and all but one of the low-cost patients were female and the spouse of the primary beneficiary. The other patient was male and the primary beneficiary.
- The primary beneficiaries for all of the low-cost patients had employee classifications of *Hourly Union*, where as the primary beneficiary for the high-cost patient was classified as *Non-Union.*
- The primary beneficiary for the high-cost patient had an industry classification of *Transportation, Communications, Utilities.* The primary beneficiary for the rest of the patients was classified as *Manufacturing, Durable Goods.*

It would be very interesting to explore this family more. It seems unlikely that the difference in cost could be explained by the differences noted above. Perhaps the cost difference is a clerical error. It would definitely be worth the time of the employer or health insurance provider to find and investigate such a discrepancy further.

5.3 Further Research

There appears to be a potential to mine further knowledge from the diagnoses and procedure codes. One possible approach would be to have an expert assign a subjective "distance" between two procedures. If such a set of distances was available, a large number of learning methods would be applicable.

In regards to the theorem presented in Section 5.1.3, we will be constructing a new algorithm to take advantage of this result. Our proposal involves adapting the CHARM algorithm [47] to find all frequent item sets using only the diagnosis codes and all frequent item sets using only the procedure codes. It should be noted that this approach will not have computed the support of the union $X_1 \cup X_2$, which we plan to deal with by using the methods developed by Zaki, namely using the *diffsets* [47].

Bibliography

[1] Thomson Medstat. MarketScan® database.

[2] Center for Disease Control and Prevention. National diabetes fact sheet: general information and national estimates on diabetes in the United States, 2005.

[3] American Diabetes Association, Digestive National Institute of Diabetes, and Kidney Diseases. The prevention or delay of type 2 diabetes. *Diabetes Care*, 25(4):742–749, April 2002.

[4] L.N. Pani, D.N. Nathan, and R.W. Grant. Clinical predictors of disease progression and medication initiation in untreated patients with type 2 diabetes and A1C less than 7%. *Diabetes Care*, 31(3):386–390, March 2008.

[5] M.P. Stern, K. Williams, and S.M. Haffner. Identification of individuals at high risk of type 2 diabetes: do we need the oral glucose tolerance test? *Ann Intern Med.*, 136(8):575–581, April 2002.

[6] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning.* Springer, August 2001.

[7] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[8] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[9] L. Brieman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[10] D. Sankoff and J. B. Kruskal, editors. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparisons.* Addison Wesley, Boston, MA, 1983.

[11] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[12] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

[13] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[14] J.-P. Vert, H. Saigo, and T. Akutsu. *Kernel Methods in Computational Biology*, chapter Local alignment kernels for biological sequences, pages 131–154. MIT Press, 2004.

[15] J. Kruskal and M. Liberman. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparisons*, chapter The symmetric time-warping problem: from continuous to discrete. Addison-Wesley, Reading, Massachusetts, 1983.

[16] B.E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[17] V. Vapnik. *Estimation of Dependences Based on Empirical Data.* Information Science and Statistics. Springer, 1st ed. 1982. Reprint, 2006.

[18] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[19] B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning).* The MIT Press, December 2001.

[20] I. Steinwart. Support vector machines are universally consistent. *J. Complex.*, 18(3):768–791, 2002.

[21] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, June 2004.

[22] D. Haussler. Convolution kernels on discrete structures. Technical report, UC Santa Cruz, 1999.

[23] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[24] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. Technical report, Dept. of Computer Science, Univ. of California, 1998.

[25] C.R. Rao. Large sample tests of statistical hypotheses concerning several parameters with applications to problems of estimation. *Proceedings of the Cambridge Philosophical Society*, 44:50–57, 1948.

[26] C.R. Rao and S.J. Poti. On locally most powerful tests when alternative are one sided. *SankhyVa*, 7:439, 1946.

[27] H. Sakoe and S. Chiba. A dynamic programming approach to continuous speech recognition. In *Proceedings of the Seventh International Congress on Acoustics, Budapest*, volume 3, pages 65–69, Budapest, 1971. Akadémiai Kiadó.

[28] H. Shimodaira, K.-I. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Advances in NIPS 14*, 2002.

[29] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines—a kernel approach. In *Proc. 8th Int. Workshop Front. Handwriting Recognition (IWFHR)*, pages 49–54, 2002.

[30] V. Wan and J. Carmichael. Polynomial dynamic time warping kernel support vector machines for speech recognition with sparse training data. In *Proc. Interspeech*, pages 3321–3324, 2005. AMI-67.

[31] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. *CoRR*, abs/cs/0610033, 2006. informal publication.

[32] A.W. Moore and K Papagiannaki. Toward the accurate identification of network applications. In *PAM*, pages 41–54, 2005.

[33] T.T.T. Nguyen and G. Armitage. Training on multiple sub-flows to optimise the use of Machine Learning classifiers in real-world IP networks. *lcn*, 0:369–376, 2006.

[34] J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *12th Annual Symposium on Computer Applications in Medical Care*, pages 261 – 265, Washington DC(USA), Nov 1988.

[35] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[36] A.P. Legorreta, J.-F. Ricci, M. Markowitz, and P. Jhingran. Patients diagnosed with irritable bowel syndrome: Medical record validation of a claims-based identification algorithm. *Disease Management & Health Outcomes*, 10(11):715–722, 2002.

[37] M. Pladevall, L.K. Williams, L.A. Potts, G. Divine, H. Xi, and J.E. Lafata. Clinical outcomes and adherence to medications measured by claims data in patients with diabetes. *Diabetes Care*, 27(12):2800–2805, Dec 2004.

[38] D.R. Miller, M.M. Safford, and L.M. Pogach. Who has diabetes? best estimates of diabetes prevalence in the Department of Veterans Affairs based on computerized patient data. *Diabetes Care*, 27:B10–B21, 2004.

[39] P.L. Hebert, L.S. Geiss, E.F. Tierney, M.M. Engelgau, B.P. Yawn, and A.M. McBean. Identifying persons with diabetes using Medicare claims data. *Am J Med Qual*, 14(6):270–277, Nov-Dec 1999.

[40] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In Nédellec C and Rouveirol C, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[41] W.S. Noble. *Kernel Methods in Computational Biology*, chapter Support vector machine applications in computational biology, pages 71–92. MIT Press, Cambridge, MA, 2004.

[42] G. Li and A. Khokhar. Content-based indexing and retrieval of audio data using wavelets. In *IEEE International Conference on Multimedia and Expo (II)*, pages 885–888, 2000.

[43] W. S. Noble and A. Ben-Hur. Integrating information for protein function prediction. In *Bioinformatics - From Genomes to Therapies*, volume 3, pages 1297–1314. Thomas Lengauer, 2007.

[44] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI Press, 1991.

[45] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[46] M.J. Zaki and M. Ogihara. Theoretical foundations of association rules. In *Proceedings of 3rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'98)*, Seattle, Washington, 1998.

[47] M.J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining.

[48] S. Doddi, Marathe A., Ravi S.S., and Torney D.C. Discovery of association rules in medical data. *Medical Informatics and the Internet in Medicine*, 26(1):25–33, January 2001.

[49] C. Ordonez, N. Ezquerra, and C.A. Santana. Constraining and summarizing association rules in medical data. *Knowl. Inf. Syst.*, 9(3):259–283, 2006.

[50] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

Appendix A

FISHER FEATURES FOR AN HMM

We were interested in the specific features that the Fisher kernel generated when applied to an HMM. In this section, we will work out the resulting features for a small example. This follows the work that can be found in Shawe-Taylor and Cristianini[50].

Let's look at the meaning of the fisher features for a Hidden Markov Model (HMM). We are looking specifically at a model for strings of length n. Note that we let s be the sequence of (observed) emissions, that is $s = (s_1, s_2, \ldots, s_n)$, where all the s_i are part of some alphabet $\mathcal{A}$. Let $\mathcal{M}$ be a collection of states and let $m = (m_1, m_2, \ldots, m_n)$, $m_i \in \mathcal{M}$ for all i, be a possible sequence of (unobserved) states and $\mathcal{M}^n$ be the class of all state sequences of length n. We also add a special beginning state, b, to allow us to begin in various states and for simplicity, we assume $m_0 = b$.

We are interested in two model parameters: the state transition probabilities $P_t(n|l)$ for all $n \in \mathcal{M}$ and $l \in \mathcal{M} \cup \{b\}$;and the set of emission probabilities $P_e(a|n)$ for all $n \in \mathcal{M}$ and $a \in \mathcal{A}$.

First, we need to remove the constraints that are a natural part of probability. We do this as follows:

Pick one state and call it m'. For each state m in our state space, we replace $P(m'|m)$ with $1 - \sum_{m_i \neq m'} P(m_i|m)$. If there are n_m states, this leads to an unconstrained model with $n_m - 1$ parameters.

Let's find the Fisher features for $P(n|l)$ where $n \in \mathcal{M}/\{m'\}$ and $l \in \mathcal{M} \cup \{b\}$. Then

$$\frac{\partial logP(s)}{\partial P_t(n|l)} = \frac{1}{P(s)} \sum_{m \in \mathcal{M}} \frac{\partial}{\partial P_t(n|l)} \prod_{k=1}^{n} P_t(m_k|m_{k-1}) P_e(s_k|m_k),$$

but

$$\frac{\partial P_t(m_k|m_{k-1})}{\partial P_t(n|l)} = \begin{cases} 1 & m_k = n \text{ and } m_{k-1} = l \\ -1 & m_k = m' \text{ and } m_{k-1} = l \\ 0 & \text{otherwise} \end{cases}.$$

Thus

$$\begin{aligned} \frac{\partial logP(s)}{\partial P_t(n|l)} &= \frac{1}{P(s)} \sum_m \sum_k \left[I_{m_k=n, m_{k-1}=l} - I_{m_k=m', m_{k-1}=l} \right] \\ &\quad \prod_{j \neq k} \left[P_t(m_j|m_{j-1}) * P_e(s_j|m_j) \right] * P_e(s_k|m_k) \\ &= \sum_m \sum_k \left[\frac{I_{m_k=n, m_{k-1}=l}}{P_t(n|l)} - \frac{I_{m_k=m', m_{k-1}=l}}{P_t(m'|l)} \right] \frac{P(m,s)}{P(s)} \\ &= \sum_k \sum_m \left[\frac{I_{m_k=n, m_{k-1}=l}}{P_t(n|l)} - \frac{I_{m_k=m', m_{k-1}=l}}{P_t(m'|l)} \right] P(m|s) \\ &= \sum_k \frac{E(I_{M_k=n, M_{k-1}=l}|s)}{P_t(n|l)} - \frac{E(I_{M_k=m', M_{k-1}=l}|s)}{P_t(m'|l)} \\ &= \sum_k \frac{P(M_k = n, M_{k-1} = l|s)}{P_t(n|l)} - \frac{P(M_k = m', M_{k-1} = l|s)}{P_t(m'|l)}. \end{aligned}$$

Now let's consider the emission probabilities $P_e(a|n)$, where $a \in \mathcal{A}$ and $n \in \mathcal{M}$. Similar to before, we need to switch to an unconstrained model. We pick a symbol s' and replace $P_e(s'|m)$ with $1 - \sum_{s \neq s'} P_e(s|m)$ for all states $m \in \mathcal{M}$.

Then for all $a \in \mathcal{A}/\{s'\}$ and all $n \in \mathcal{M}$

$$\frac{\partial logP(s)}{\partial P_e(a|n)} = \frac{1}{P(s)} \sum_m \frac{\partial}{\partial P_e(a|n)} \prod_{k=1}^{n} P(m_k|m_{k-1}) * P(s_k|m_k),$$

but

$$\frac{\partial P(s_k|m_k)}{\partial P(a|n)} = \begin{cases} 1 & s_k = a \text{ and } m_k = n \\ -1 & s_k = s' \text{ and } m_k = n \\ 0 & \text{otherwise} \end{cases},$$

and therefore

$$\begin{aligned}
\frac{\partial log P(s)}{\partial P_e(a|n)} &= \frac{1}{P(s)} \sum_m \sum_k [I_{s_k=a,m_k=n} - I_{s_k=s',m_k=n}] \\
&\quad \prod_{j \neq k} [P_t(m_j|m_{j-1}) * P_e(s_j|m_j)] * P_t(m_k|m_{k-1}) \\
&= \sum_m \sum_k \frac{I_{s_k=a,m_k=n}}{P_e(a|n)} - \frac{I_{s_k=s',m_k=n}}{P_e(s'|n)} \frac{P(m,s)}{P(s)} \\
&= \sum_k \sum_m \frac{I_{s_k=a,m_k=n}}{P_e(a|n)} - \frac{I_{s_k=s',m_k=n}}{P_e(s'|n)} P(m|s) \\
&= \sum_k \frac{E(I_{S_k=a,M_k=n}|s)}{P_e(a|n)} - \frac{E(I_{S_k=s',M_k=n}|s)}{P_e(s'|n)} \\
&= \sum_k \frac{P(S_k=a, M_k=n|s)}{P_e(a|n)} - \frac{P(S_k=s', M_k=n|s)}{P_e(s'|n)}
\end{aligned}$$

This completes the derivation of the Fisher kernel for an HMM.

Appendix B

LEARNING CURVE

First, it is necessary to make sure that we include enough examples to allow the classifier to achieve good performance. To get an idea of the effect of sample size on the accuracy of the SVM, we used our bag of codes(BoC) kernel applied to the diagnosis and procedure codes. The positive and negative examples selected above that had a three to five year claim window were used in the process. We tested the performance of a classifier trained with the following training set sizes: 100; 250; 500; 1,000; 10,000; 20,000; 40,000; and 59,000. Performance was measured using mean classification success rate measured through multiple runs of five-fold cross-validation.

Figure B.1 shows the results of the experiment. We can see that the success rates quickly climb to the near maximum value and plateau.

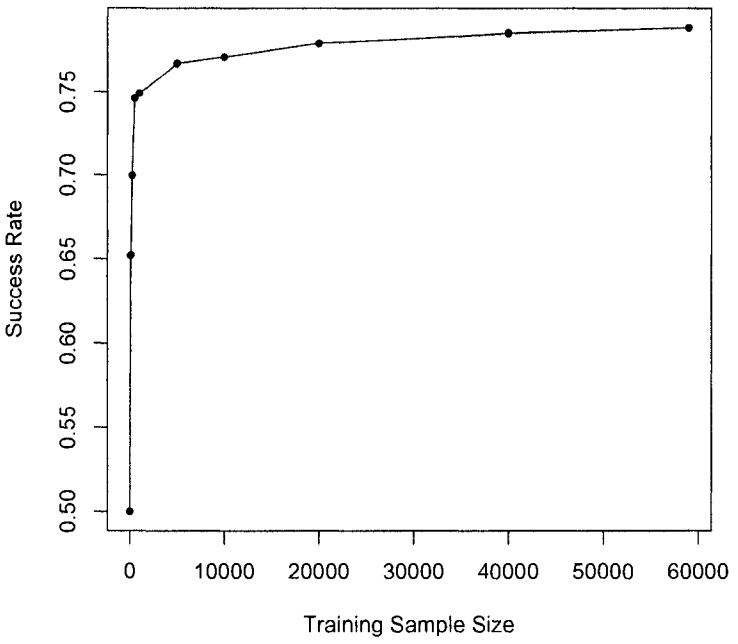


Figure B.1: Learning curve for the BoC kernel. The curve represents an estimate of the average success rate, when trained on different sizes of sized training sets.