A HIERARCHICAL STRUCTURED MACHINE-LEARNING METHOD FOR LARGE-SCALE MULTI-CLASS PROBLEMS

by

Michael Bernard Butler

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science

at

Dalhousie University
Halifax, Nova Scotia
August 2014

Table of Contents

List of Tables

List of Figures

Abstract

When a clinician diagnoses a patient, they do so by choosing one from many possible diagnoses. This is a laborious process, one that requires input from many different sources of information. It would be useful to have an objective tool to give a prediction of a patients diagnosis using readily available clinical information.

Although this would be useful, one needs to still choose from many different possible choices, a large scale multi-class problem that conventional classification methods may not be suited to solve. We describe a method that assigns a class label to an observation from a large number of class possible labels, and gives the probability of said observation having such. The method uses a combination of support vector machines, and an agglomerative hierarchical clustering algorithm to perform the task. We display the performance of the method on a benchmark problem, and a hospital-based dataset from Halifax, NS.

List of Abbreviations Used

CTAS:	Canadian Triage Assessment Scale
ED:	Emergency Department
EDIS:	Emergency Department Information System
HME:	Hierarchical Mixture of Experts
ICD-9:	International Statistical Classification of Disease
MSM:	Maximum Separating Margin
SVM:	Support Vector Machine

Acknowledgements

I would like to recognize the support of my loving wife, Mari Ito, who tolerated the long nights of my furrowed brow while I completed my thesis. It is only with her comforting care and ability to feed me takeout food that I ever had the chance to finish this work. Love is nature's methamphetamine.

I also need to recognize the wisdom and kindess of my thesis advisor, Dr. Hong Gu, who held me accountable when I needed to be and guided me gently along my way. In this work, like others, there are no stupid questions, but certainly not for my lack of trying. You once told me that you are not a tiger, but I kindly disagree. Your passion for your students is obvious, and I certainly am the beneficiary of it. I appreciate your patience and understanding.

I appreciate the advice of my clinical supervisor, Dr. Alix Carter, who guided me through the research ethics process at the Capital District Health Authority as well as my grant application with the same institution. Her keen eye for detail, and her patient-oriented approach consequently has made me more focused as well.

I would be remiss to not mention the support of the Capital District Health Authority Research Fund Trainee Grant, which allowed me to apply this work to a real, clinical dataset, and provide the foundation for future research.

Finally, I would like to thank my fellow graduate students in the Department of Mathematics and Statistics: Lihui Liu, Maria Reyes, Dong Lin, Shen Ling, Yun Cai, David Fey, Mary Roop and Li Li. Your thirst for knowledge and your indomitable work ethics provided an example for me to follow. Know that my research was as much an effort to mimic your own qualities as it was for its own sake.

Chapter 1

Introduction

"The art of medicine consists in amusing the patient while nature cures the disease."
- Voltaire

The unfortunate reality of patient diagnosis is that it is a mottled process, full of noise, and miscues. A physician needs to, often in a hurried manner, collate information about a patient from multiple sources in order to arrive at a specific, clear diagnosis.

Much as one would like to think of this process as perfectly objective, the truth is that there is an art to medicine. There is very rarely one clear choice of pathology for a particular patient presentation. Usually, a physician needs to decide between several diagnoses, and each of these may have a very different treatment and risks for the patient in question.

In attempt to lend some objectivity to this decision-making process, we consider the process of differential diagnosis. In the method of differential diagnosis, possible pathologies that a patient may be stricken with are given probabilistic weight as evidence is weighed. As evidence surfaces from clinical exam, radiology results and clinical lab values, the probability of a given disease increases or decreases in response. A schematic example from a 2007 study involving infections of the eyelid is below (Papier et al., 2007).

Eventually though, one needs to decide on a diagnosis. In this method, one chooses the disease that is most probable given the evidence to that point. One should note that this does not preclude alternative diagnoses: our decision is simply the one that is likeliest with the information available.

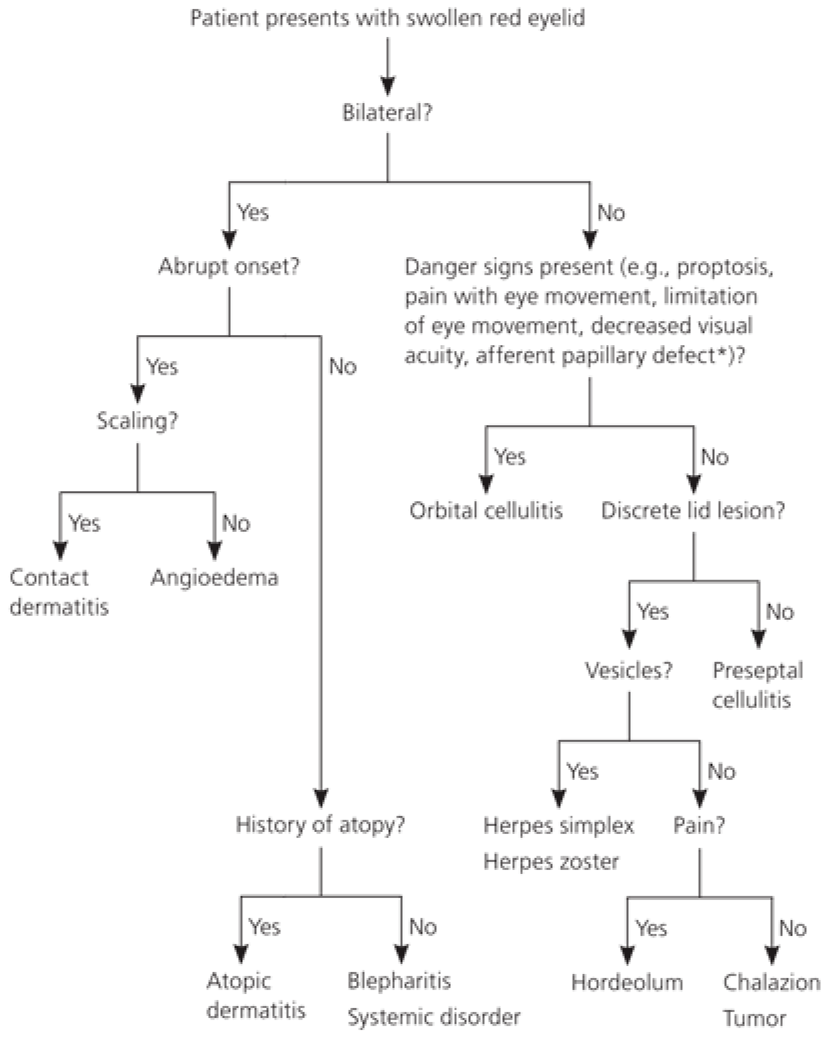

Figure 1.1: An example of the differential diagnosis process

Although the process of differential diagnosis is certainly attractive, it is also not fool-proof. Studies of emergency department patients indicate that the rate of misdiagnosis may be as high as fifteen percent(Burroughs et al., 2007). This suggests that an accurate, efficient and objective system would be invaluable to patient diagnosis in this setting. Unfortunately, there are specific challenges to the design of such a system. An obvious stumbling block is the existence of so many diagnoses.

Although the problem of prediction given a high-dimensional predictor space is well-described, there has been much less consideration given to when the "predicted" space is high-dimensional. Additionally, as noted above, the act of diagnosis is extremely complex. This likely makes the use of conventional linear methods a poor choice of methodology. This provides the impetus to attempt the use of machine-learning methodology, which may provide a better approximation to the messy, nonlinear nature of patient pathology.

Even so, the high-dimensional nature of the problem still poses challenges to accurately predicting the proper class label. As we shall see, there have been attempts to resolve the difficulty using structured classification, which presumes a structure in the class labels, namely that they are grouped in logical clusters. Utilizing this structure may facilitate the accurate prediction of class labels, even in the high-dimensional case.

It can be easily intuited from the concept of differential diagnosis that there is a strong analog between the clinical method and structural class learning. With this in mind, we will attempt to develop our own structural learning method and apply it in a real, clinical application.

1.1 Examples of Machine-Learning Methodology in Medicine

The field of machine-learning has piqued intense interest in the field of medicine, specifically in the disciplines of cardiology, radiology and oncology. Here, we present some examples of applications of data-mining methodology.

A study from Beijing studied the performance of an ensemble data-mining method to predict the treatment of hypertension in 167 cardiology patients (Chen et al., 2012). The treatment outcome was stratified into three classes (good treatment, normal treatment and poor treatment). The investigators used four methodologies: a back-propagation neural network, a generalized regression neural network, a naive bayesian classifier and one-versus-one support-vector machines (SVM) with a radial basis kernel.

Ultimately, the predictions of the component classifiers were combined using a weighted vote, and achieved a correct classification rate of 95.34% with a kappa multi-rater reliability coefficent of 92.54%. However, it should be noted that the researchers validated these results over ten test sets generated by boot-strapping.

Another study from Germany compared the ability of SVMs the authors had previously trained in a prior study to differentiate patients with Alzeimer's disease from those with dementia associated with normal aging or fronto-temporal lobe degeneration with experienced radiologists, using a test set of 105 structural magnetic resonance imaging (MRI) scans (Klöppel et al., 2008). The SVM method outperformed or performed at least as well as the radiologists in every case, with the lowest sensitivity being 83.3%, and the lowest specificity being 85.7%. This study is novel in that it represents a follow-up to previous work by the authors in an attempt to give the method a clinical context.

Also from Germany is a paper from a pediatric hospital located in Hannover. In this paper, the authors evaluated an ensemble machine-learning technique to diagnose patients that presented to the emergency department with one of eighteen different abnormalities (Grigull and Lechner, 2012). Their ensemble method included an artificial neural network, a SVM and fuzzy logics. The predictions were again combined into a majority weighted vote. The training set included 566 patients' records, and the algorithm had a correct classification rate of 97.7%. The test set included 126 patients, and the method showed a correct classification rate of 81.1%.

This should suffice to convince the reader about the general interest in data-mining in the medical community. However, there are short-comings that are illustrated by the studies above in the current state of the art. One such limitation is the limited number of diagnoses the methods above are differentiating between. This naturally limits the utility of a given classifier, since it assumes some significant pre-knowledge about a patient's final diagnosis.

In our MRI study above the researchers are certain of a given patient's diagnosis.

In prospective, clinical application there always exists alternative possibilities. As an example, neurosyphilis often mimics the symptoms associated with Alzeimer's such as progressive cognitive decline and drastic behavioural changes (Mehrabian et al., 2012). Neurosyphilis is also associated with mesiotemporal atrophy on MRI, and it is uncertain what the SVM method would predict in this case. To be of maximum utility, a diagnostic machine-learning method needs to account for these deviations that could occur in the natural population.

An additional concern also is the nature of the candidate predictors in the studies above. In both studies by Chen et al. (2012) and Grigull and Lechner (2012), the data were gathered in a retrospective fashion. A predictive model that is designed for use in clinical application will not benefit from such a retrospective review. In particular, Grigull performed a chart review on his patient group, and entered his findings into a standardized database. Although sufficient as proof-of-concept, the relevant features for diagnosis in practice must be readily available, ideally in an electronic format.

We hopefully have developed a sense of the utility of machine-learning applied to the clinical context, specifically using features obtained in real-time by electronic patient information systems, and differentiating between many possible class labels (in this case diagnoses). As mentioned above, we have developed our own methodology to perform this task, and have applied it to a real dataset obtained from the local emergency department in Halifax, Nova Scotia. However, there are many diagnostic labels to differentiate, and to make our discussion clear, we have applied the method to a benchmark problem in machine-learning using a dataset from the United States Postal Service. We review both below.

1.2 Description of the Emergency Department Data

In 2013, the Capital District Health Authority (CDHA) Emergency Departments (ED) saw a total of 137,006 adult patient presentations. These hospitals include the Charles V. Keating Emergency Department in Halifax, the Dartmouth General Hospital, the Cobequid Health Sciences Centre and the Hants Community Hospital.

They serve as the provincial and maritime referral centeres for trauma, burns, critical care and a number of medical surgical subspecialities, as well as being the community hospitals that serve the population of the Halifax Regional Municipality, which is roughly 390,000 people.

The Emergency Department Information System (EDIS) is the central patient information system for the EDs in the CDHA. It currently contains over a million patient records, which may be uniquely linked to other clincial laboratory systems, such as those in the CDHA Lab and Pathology Systems. EDIS also offers the advantage of having these data available in real-time, as all the values in the system are entered as the patient presents to and proceeds through the ED.

Using EDIS, one can extract the following variables relevant to building a diagnostic system for the emergency medicine population:

- Account number of the patient visit, which is a unique value and can be used to collate information from other CDHA systems.
- Patient age
- Gender
- Vital signs at triage, such as pulse rate, blood pressure, glucose level, Glascow coma scale(Teasdale and Jennett, 1974)
- Clinical comments, a free-text field the physician uses to add notes about the patient
- Length-of-stay within the department
- Canadian Triage Assessment Score (CTAS)(Warren et al., 2008; Bullard et al., 2008)
- Patient's final ICD-9 diagnosis

The remainder of the data elements are obtained from the Lab and Pathology System, which contains all clinical lab results, such as electrolyte panels, hematology results, and radiology reports.

For the purpose of this study, we utilized our methodology on a dataset provided by the Department of Emergency Medicine at CDHA. This dataset included all patients that presented from May 1^{st}, 2013 to April 30^{th}, 2014 to the departments at the Halifax Infirmary, the Dartmouth General Hospital, the Cobequid Community Health Centre and the Hants County Hospital. These totaled 137,006 presentations in total, with 1581 unique ICD-9 diagnoses within.

All diagnoses that occurred less than 100 times were removed from the dataset. The reason for this is two-fold: first, there may not be enough data to differentiate such a diagnosis from the many others in EDIS. The other reason is to protect patient confidentiality, as particularly rare diagnoses may expose such people to the danger of unnecessary identification. We also excluded all patients that were trauma team activations, as their injuries are usually multi-systemic and variable. Also, the utility of predicting pathology in such a patient population is minimal, as the clinician is already aware of their pathology.

After these exclusions, 108,870 observations and 287 unique class labels remained. The observations were stratified by their ICD-9 diagnosis, and then 80% (87,209 presentations in total) were randomized into a training set and 20% (21,161 in total) were randomized into a test set to ensure a balanced representativeness of the class labels in both datasets. A full listing of the diagnostic labels can be seen in the appendix.

Some time should be spent exploring the final point in the list above. The ICD-9 diagnosis is our target outcome in this particular data problem. The ICD-9 diagnosis is a codified field in EDIS. When the physician is ready to diagnose a particular patient, he or she picks the relevant choice from a list in the system. Herein lies the complexity of the problem: there exists over a thousand unique ICD-9 codes, or diagnoses, within this system. This is an extremely high-dimensional problem, and "off-the-shelf" methods may not be successful. Additionally, one can anticipate that the data may be quite noisy, as patients tend to have a unique presentation even for the most common of pathologies, and there may often be missing or erroneous data. In short, it is likely this is a complicated, non-linear problem.

The data reside at the Chase Building at Dalhouise University, with access to the cluster computing resources, and will remain there for a period of no less than seven years, in accordance with CDHA ethical procedures. Please note that this study was supported with Capital District Health Authority Research Fund Trainee Grant of $5000.00. Full ethical approval for the study came from the CDHA Research Ethics Board.

1.3 Description of the Zip Code data

The discussion for the remainder of the thesis will be guided using a data set available from the *Elements of Statistical Learning* (Hastie et al., 2009). Although the final model will attempt to differentiate between an extremely large number of classes, for the purpose of demonstration we will consider a smaller problem with ten classes. The goal of the resultant method is to be generalizable to a varying number of classes.

The data are a series of handwritten digits that were scanned by the U.S. Postal Service. The images are deslanted and size-normalized, and each is a 16 pixel x 16 pixel greyscale image. Each observation consists of the digit in question, zero to nine, and the greyscale values, 256 predictors in total. The original images may be seen at the website http://cs.nyu.edu/~rowels/data.html. This dataset had been worked on since the neural network research group at AT&T research labs in 1990. The training set includes 7291 observations, and the test set includes 2007 observations.

The zip code data are conveniently split into a training set and test set that offers a relatively constant ratio of the digits within both datasets:

	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine
Training	1194	1005	731	658	652	556	664	645	542	644
Test	359	264	198	166	200	160	170	147	166	177

Finally, all data processing was performed using mySQL 6.0 from the Oracle Corporation, and all programming implementing the method was performed in the R-statistical computing environment, version 3.10 within the R-studio GUI Version 0.98.501. Additionally, it should be noted that our method used the *e1071* package (Meyer, 2012), the R implementation of the excellent work from MIT on support vector machines (Chang and Lin, 2011). We also utilized the *cluster* package from Maechler et al. (2014), in particular the *hclust()* function. All R-code can be found within the appendix.

Chapter 2

Background Methods

"Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away."
- Antoine de Saint-Exupery

To differentiate between the many classes in the large multi-class problem we have developed an amalgamation of the methodologies associated with support vector machines, hierarchical mixture of experts and agglomerative clustering. We provide a brief overview below of each of these methodologies, and integrate them into a comprehensible whole at the conclusion. In addition we will introduce "class-structure learning", a concept that utilizes the idea that there is an underlying informative structure underneath the different class labels that enable us to improve the predictive performance of our model.

2.1 Support Vector Machines

Support vector machines play an important role in the formation of the class structure that is the foundation of our proposed method, so we now briefly review the concepts associated with them. Support vector machines in their current form are considered to have been introduced and developed by Cortes and Vapnik (1995), which itself is a contemporary form of the first algorithm for pattern recognition by the familiar R.A. Fisher (1936). For the purpose of our discussion we will focus exclusively on the case with only two class labels, as is the case with the application of our method. We will additionally first focus on the case where the data are linearly separable, and then extend this to the case where the data are not linearly separable. This review is primarily based on the description of the method in Hastie et al. (2009).

2.1.1 Optimal Separating Hyperplanes

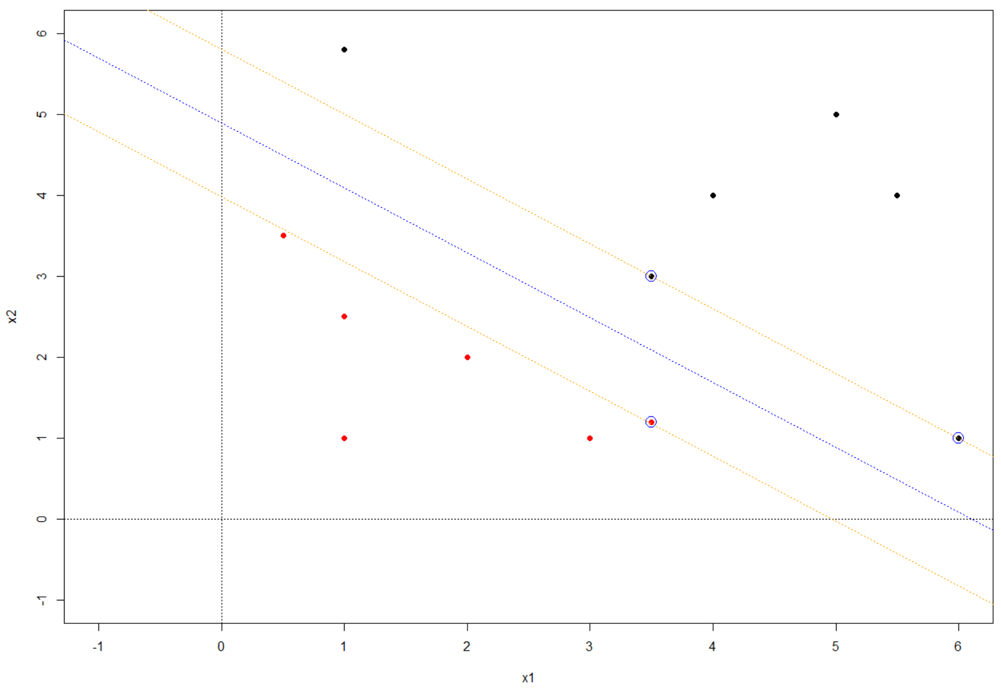

Figure 2.1: A linearly separable SVM, with margins plotted

The central concept in support vector machines is finding the hyperplane that maximizes the distance between the points that constitute either of the class labels. A useful representation of this is in Figure 2.1. Consider first that we have some data that consists of N pairs $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$, where $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. Here, x_i is the feature space for a particular observation, and y_i is the binary class label, represented by blue points for class 1, and red points for class 2 in Figure 2.1. Now, consider a hyperplane defined as:

$$\{x : f(x) = x^T\beta + \beta_0 = 0\}$$

Further note that for any two points x_1 and x_2 lying on the hyperplane $\beta^T(x_1 - x_2) = 0$, and the vector normal to this hyperplane is given by $\beta^* = \frac{\beta}{||\beta||}$. Furthermore, for any point, x_0 on the hyperplane $\beta^T x_0 = -\beta_0$. Then, the signed distance of any point x to $\{x : f(x) = 0\}$ is given by:

$$\beta^{*T}(x - x_0) = \frac{1}{||\beta||}(\beta^T x - \beta^T x_0) = \frac{1}{||\beta||}(\beta^T x + \beta_0)$$

Since $f(x) = \beta^T x + \beta_0$, we can see that $f(x)$ is proportional to the signed distance from x to the hyperplane defined by $f(x) = 0$. The significance of the signed portion is that we can intuit a useful classification rule from the sign of said distance:

$$G(x) = sign(f(x)) = sign(x^T \beta + \beta_0)$$

Put explicitly, if $G(x)$ is less than zero, we classify the observation with feature vector x^T to the class -1. Otherwise we classify it to the class 1. Now, consider the quantity $y_i(\beta^T x_i + \beta_0)$. One can see that for the signed distance $f(x)$ discussed above, this quantity will be negative should the class label and the prediction given by the dividing hyperplane not agree. It is obviously in our benefit to maximize this quantity. This leads us to the idea of *optimal separating hyperplanes* (Cortes and Vapnik, 1995). The optimal separating hyperplane separates the two classes and also maximizes the distance between the closest points of each class and the hyperplane. Now, if M is the size of the margin between two classes, define the optimization problem:

$$\underset{\beta,\beta_0,||\beta||=1}{max} M$$

$$\text{subject to y}_i(x_i^T \beta + \beta_0) \geq M, i = 1, ..., N$$

(2.1.1)

The constraints ensure that all of the points are, at minimum, a signed distance M away from the hyperplane defined by β and β_0. Now, we can drop the norm constraint on $||\beta||$ above by re-expressing (2.1.1) as:

$$\max_{\beta,\beta_0} M$$

$$\text{subject to } \mathrm{y}_i(x_i^T\beta + \beta_0) \geq M||\beta||, i = 1, ..., N$$

(2.1.2)

The above inequality can be satisfied by multiplying by any positive constant on both sides, thus we can set $||\beta||$ to $\frac{1}{M}$, and (2.1.2) becomes:

$$\min_{\beta,\beta_0} \tfrac{1}{2}||\beta||^2$$

$$\text{subject to } \mathrm{y}_i(x_i^T\beta + \beta_0) \geq 1, i = 1, ..., N$$

(2.1.3)

This defines a margin around the hyperplane $f(x) = 0$ of thickness $\frac{1}{||\beta||}$, represented by the parallel lines in Figure 2.1. Overall, the total width of the margin is $\frac{2}{||\beta||}$, from bound to bound. One can see that 2.1.3 is a convex optimization problem, as we are minimizing a quadratic function subject to linear constraints. Define the Lagrange primal function

$$L_p = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i[y_i(x_i^T\beta + \beta_0) - 1]$$

(2.1.4)

that we are going to minimize with respect to β and β_0. After taking the derivatives and setting them to zero, we see that:

$$\beta = \textstyle\sum_{i=1}^{N} \alpha_i y_i x_i$$

(2.1.5)

$$0 = \sum_{i=1}^{N} \alpha_i y_i$$

(2.1.6)

We then subsequently substitute these in (2.1.4), the Wolfe dual function is obtained as below:

$$\mathrm{L}_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0$$

(2.1.7)

In addition to the constraints described directly above, the solution by maximizing L_D must satisfy the Karush-Kuhn-Tucker conditions, and also satisfy that for all i:

$$\alpha_i[y_i(x_i^T \beta + \beta_0) - 1] = 0$$

For the above condition to be true, we can see that either $\alpha_i = 0$, or $y_i(x_i^T \beta + \beta_0) = 1$. This means that β in (2.1.5) is defined by a linear combination of the x_i where the latter condition is true. These x_i's are on the boundary of the slabs. These points are known as the *support points* for their role in defining the margin around the dividing hyperplane. The circled points in Figure 2.1 represent these points. Heuristically, we can see the importance of the support points, for if we were to move these points, the margin would concomitantly enlarge or shrink with their transposition.

Now, the solution described above exists when the two classes in question are linearly separable, and allows no points to overlap across the dividing hyperplane. This is known as a *hard margin.* In the case where the classes overlap in their feature space and are not linearly separable, we can solve this problem by allowing for some of the points to exist on the wrong side of the margin and is, in contrast, known as a *soft margin.* We can do so using the *support vector classifier*, which we now describe in more detail.

2.1.2 Support Vector Classifier

Let us return to the convex optimization we first defined in (2.1.1):

$$\max_{\beta,\beta_0,||\beta||=1} M$$

$$\text{subject to } \mathrm{y}_i(x_i^T\beta+\beta_0) \geq M, i = 1, ..., N$$

Now, define the variables $\xi = (\xi_1, \xi_2, \cdots, \xi_N)$, and modify the constraint in (2.1.1) as:

$$\max_{\beta,\beta_0,||\beta||=1} M$$

$$\text{subject to } \mathrm{y}_i(x_i^T\beta+\beta_0) \geq M(1-\xi_i), i = 1, ..., N$$

(2.1.8)

where, for all i and K being some constant, $\xi_i \geq 0, \sum_{i=1}^{N} \xi_i \leq$ K. The value of ξ_i in the above dictates the proportional amount by which the prediction $f(x_i)$ is on the erroneous side of the margin. By modifying the constraint in this way, we may construct the separating hyperplane, while allowing some of the training examples to be misclassified. The bounding condition $\sum \xi_i$ also bounds the total number of misclassifications during the training procedure. It is convenient, as in 2.1.3 to express the above while dropping the norm constraint by defining $M = \frac{1}{||\beta||}$:

$$\underset{\beta,\beta_0}{min}\, \tfrac{1}{2}||\beta||^2$$
$$\text{subject to } \mathrm{y}_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \xi_i \geq 0, \textstyle\sum \xi_i \leq K,$$
$$\mathrm{i} = 1, \mathrm{.~.~.,~N}$$

(2.1.9)

This is also a convex optimization problem. We turn again to the quadratic programming solution using the Lagrangian method. Re-express (2.1.9) as:

$$\underset{\beta,\beta_0}{min}\, \tfrac{1}{2}||\beta||^2 + C\textstyle\sum_{i=1}^{N} \xi_i$$
$$\text{subject to } \xi_i \geq 0, y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i\, \forall i$$

(2.1.10)

Here, C is a *cost parameter* that replaces K above. We then form the Lagrange primal function:

$$\mathrm{L}_p = \tfrac{1}{2}||\beta||^2 + C\textstyle\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{N} \mu_i\xi_i$$

(2.1.11)

We now minimize with respect to β and β_0 as before, and also ξ_i:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$$
$$0 = \sum_{i=1}^{N} \alpha_i y_i$$
$$\alpha_i = C - \mu_i, \forall i$$

(2.1.12)

Note also that $\alpha_i, \mu_i, \xi_i \geq 0 \, \forall i$. Substituting this, and the quantities from (2.1.12) into (2.1.11), we obtain the Wolfe dual,

$$\mathrm{L}_D = \sum_{i=1}^{N} \sum \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

(2.1.13)

We now maximize L_D subject to $0 \geq \alpha_i \geq C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$ and the following Karush-Kuhn-Tucker conditions:

$$\alpha_i[y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0,$$
$$\mu_i \xi_i = 0,$$
$$\mathrm{y}_i(x_i^T \beta + \beta_0) - (1 - \xi_i) \geq 0$$

(2.1.14)

The collection of (2.1.12) and (2.1.13) will give the solution to the primal and dual problem. We are, of course, interested in the solution vector β:

$$\widehat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$$

(2.1.15)

Now, because from the first condition in (2.1.14), $\widehat{\beta}$ will only be non-zero for non-zero $\hat{\alpha}$ and for which the last constraint in (2.1.14) are met. This means that the solution $\widehat{\beta}$ are defined solely, as in the linearly separable case, by those observations that are on, or within the margin. These points are termed the *support vectors* for this reason. Of these observations, for those on the edge of the margin, $\xi_i = 0$, and so:

$$\mu_i \xi_i = 0 \Rightarrow 0 \leq \mu_i \leq C$$

$$\widehat{\alpha_i} = C - \mu_i \Rightarrow 0 \leq \alpha_i \leq C$$

since $\alpha_i \geq 0$. Now, for those points that are in the margin, $\xi_i > 0$, we can see easily that:

$$\mu_i \xi_i = 0 \Rightarrow \mu_i = 0,$$

$$\widehat{\alpha_i} = C - \mu_i = C - 0 = C$$

After obtaining $\hat{\beta}$, we can obtain $\hat{\beta}_0$ from the first criteria in (2.1.14) using any the described points, x_i, on the margin. Given these quantities, we can now utilize the familiar decision function:

$$\widehat{G}(x) = sign[x^T \hat{\beta} + \hat{\beta}_0]$$

(2.1.16)

where again, classifying observations into classes {-1, 1} as dictated by $\widehat{G}(x)$.

It should be noted that C is a “tuning” parameter that is associated with the construction of the SVM, and the solution above depends on the value that we choose. One usually does so by cross-validation on one’s training examples, and the intuition is that good performance will generalize well to prediction on the test observations. We finally note that we have focused on the case where the data can be well-separated linearly. However, this may not necessarily be the case and a non-linear boundary in the original feature space may be a better choice for class separation. We now briefly describe how this may be performed.

2.1.3 Support Vector Machines

It is often easier to find linear separation by enlarging the feature space of the input variables using basis expansions. This linear separation in the enlarged space usually result in better predictive performance, and translate to a non-linear boundary in the original feature space. The elegance of the procedure is that once one selects the basis functions, such as a set of polynomial functions, the optimization procedure and derivation of the solution is identical to the process outlined in the last two sections.

Explicitly, we choose basis functions $h(x), m = 1, \cdots, M$ and transform the original features x_i to $h(x_i) = (h_1(x_i), h_2(x_i), \cdots, h_M(x_i))$. We then utilize these transformed feature vectors in the optimization procedure. Define the Lagrange dual:

$$\mathrm{L}_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \langle h(x_i), h(x_k) \rangle \tag{2.1.17}$$

As we can see, the only difference between (2.1.17) and (2.1.14) is that the product $x_i^T x_k$ is replaced by the inner product $\langle h(x_i), h(x_k) \rangle$. The solution function is:

$$f(x) = h(x)^T\beta + \beta_0 = \textstyle\sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i)\rangle + \beta_0$$

(2.1.18)

From above, we can see that the solutions we are interested in involve the inner products of the enlarged features. So, we do not need to give $h(x)$, but only the form of the kernel function

$$K(x, x') = \langle h(x), h(x')\rangle$$

(2.1.19)

It is for this reason that this is colloquially known as the "kernel trick". We can now re-write (2.1.18) as

$$\widehat{f}(x) = \textstyle\sum_{i=1}^{N} \hat{\alpha} y_i K(x, x_i) + \hat{\beta}_0$$

(2.1.20)

There are many choices one can make for the kernel function, but popular candidates are polynomials of varying degrees or the Gaussian kernel ($e^{(-\gamma||x-x'||^2)}$, also known as the radial basis kernel).

To summarize, a support vector machine attempts to find the maximum margin between points and a dividing, linear hyperplane that separates observations into two classes. One can allow some observations to be classified on the wrong side of the division by utilizing slack variables ξ_i, the so-called *soft margin.* To improve prediction, one can attempt to enlarge the feature space utilizing basis transformations and the kernel trick. As we shall see in the next chapter, the length of this margin, $\frac{2}{||\beta||}$ features prominently in our methodology. We now briefly discuss utilizing SVMs to obtain the probability of being a particular class.

2.1.4 Using SVMs to Obtain Probability Estimates

Up to this point, we have focused on the construction of the decision rule, $\hat{G}(x)$, which assigns a class label based on $sign(\hat{f}(x))$. Work has been made to extend SVMs to give estimates of the probability of a particular class label (Wu et al., 2004; Lin and Weng, 2007; Platt, 2000).

libSVM, the excellent implementation provided by Chang and Lin (2011) implements these probability estimates and a full treatment of the implementation is given there. For the purpose of the thesis, we consider the pairwise comparison between class labels $\{-1, 1\}$, we want to estimate $P(y = 1|x)$. For the two-class problem this is

$$P(y = 1|x) \approx \frac{1}{(1 + e^{A\hat{f}(x)+B})}$$

The solution to A and B above are estimated by minimizing the negative log-likelihood of the training data, using their known class labels y_i, and the decision values $\hat{f}(x)$. As discussed above, $\hat{f}(x)$ is the value at which we assign a class label. As we can see from the equation above, for values of large magnitude of $\hat{f}(x)$ we become more certain of the class label, as the probability becomes closer to one or zero for these values. Luckily, $P(y = -1|x)$ is simply the complement of the above for the two-class problem. We will develop the importance of this probabilistic prediction in the next chapter.

2.2 Agglomerative Hierarchical Clustering

Clustering techniques fall within the realm of unsupervised statistical learning. Hierarchical clustering is an attractive subset of these techniques, as it does not require the practitioner to a priori select the number of clusters to which the data may logically belong. In hierarchical clustering, one specifies a measurement of dissimilarity between groups of the observations, and the groups are merged into clusters based on these differences. As the merges are performed, they produce a ranked structure of $N - 1$ levels, where N is the number of observations, based on the order in which the groups are combined, hence the name *hierarchical* clustering.

There are two possible over-arching strategies for deciding how to group the observations: *top-down* (divisive) or *agglomerative* (bottom-up). In the divisive strategy, all of the observations are grouped initially in a large "master" cluster. What then occurs is that based on the differences between the observations, this cluster is split into two clusters that possesses the maximum between-group dissimilarity.

In the agglomerative strategy, which is our focus for the method at hand, all of the observations begin in their own independent cluster (termed a *singleton*). At each step the most similar (or least dissimilar) observations are combined into a cluster. This process repeats itself until all of the observations and groups are combined into a single cluster. One must provide a measurement of dissimilarity in order to perform this task. For the purpose of this thesis, we consider three types of these measurements here.

First consider two groups, G and H. The dissimilarity between G and H is denoted by $d(G, H)$, and is computed by the pairwise dissimilarities between d_{ik}, where i is in G, and k is in H.

Single Linkage

Single linkage agglomerative clustering takes the intergroup dissimilarity to be that of the closest (most similar) pair.

$$d_{SL}(G, H) = \underset{i \in G, k \in H}{min} d_{ik}$$

Complete Linkage

Complete linkage agglomerative clustering takes the intergroup dissimilarity to be that of the furthest (most dissimilar) pair.

$$d_{CL}(G,H) = \max_{i \in G, k \in H} d_{ik}$$

Group average

Group average clustering uses the average dissimilarity between the groups.

$$d_{GA}(G,H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{k \in H} d_{ik}$$

where N_G and N_H are the number of observations in G and H respectively.

As we shall see in the ensuing chapters, the use of agglomerative hierarchical clustering is a useful tool to provide structure to a given dataset. However, it has been noted that the strucutre conferred by these clusterings above are as much a function of the algorithms as they are of any intrinsic difference in the data (Hastie et al., 2009). The choice of the algorithm or a small change to the data may result in a much different dendrogram. As such, it is important to keep this in mind when interpreting the results of any method that uses it and provides motive to utilize many different measurements of dissimilarity for the purposes of comparison. Although it pays to be a skeptic about the structure of the hierarchy, it does provide a useful topology for our proposed method, which may be considered an adaptation of the technique known as *hierarchical mixture of experts.*

2.3 Hierarchical Mixture of Experts

Hierarchical mixture of experts (HMEs) is a method that utilizes a topological assortment of splits, and fits a softmax model at each node to provide a probabilistic prediction in multi-classification problems (Jordan, 1994). HMEs can be considered a variant of the familiar tree-based methods. A schematic of a simple HME model is presented in Figure 1 below.

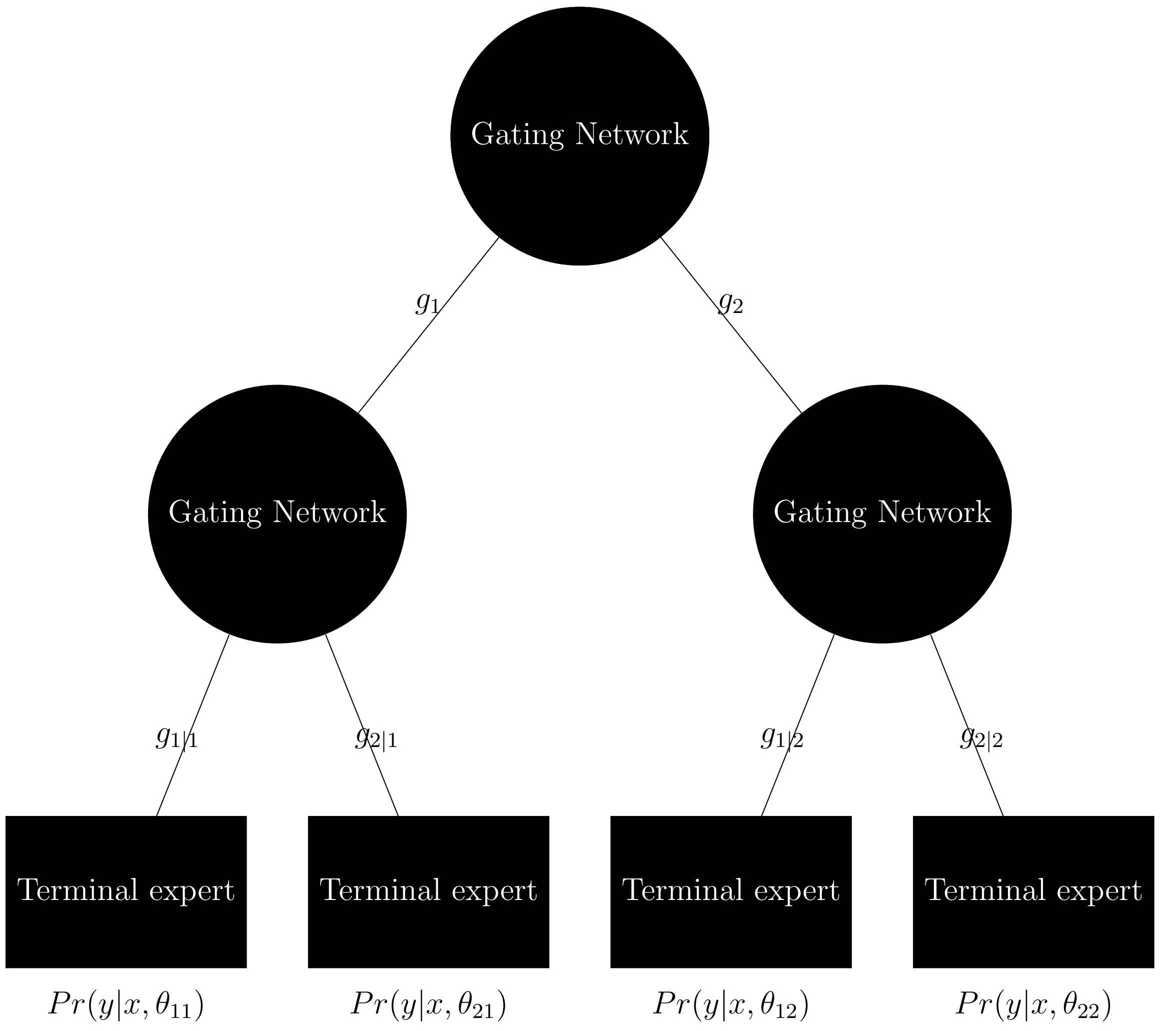

Figure 2.2: Schematic of a HME

In the general configuration of the technique, an observation begins in the most superficial node of the tree, which also includes every other observation. This node is termed a *gating network*. A classifier at this point predicts which branch the observation takes to the next gating network. This process is repeated until the observation reaches the terminal node, which is termed the *expert network*. It is at this node that the observation is given a classification label.

The difference between traditonal tree-based methods and HMEs is that the prediction at each of the nodes is actually a probability. This results in a *soft-split* as opposed to the usual hard-split we are enured to in the classical tree-based method. The top gating network has output:

$$\mathrm{g}_j(x,\gamma_j) = \frac{e^{\gamma_j^T x}}{\sum_{k=1}^{K} e^{\gamma_k^T x}}, \qquad j = 1, 2, \cdots, K.$$

(2.3.1)

The subsequent gating networks will have a similar expression, that is conditional on the result from the preceding nodes:

$$\mathrm{g}_{l|j}(x,\gamma_{jl}) = \frac{e^{\gamma_{jl}^T x}}{\sum_{k=1}^{K} e^{\gamma_{jk}^T x}}, \qquad l = 1, 2, \cdots, K.$$

(2.3.2)

This culminates at terminal, expert node, wherein there is a classifier (usually linear logistic regression) that determines the class label of the observation. Where Ψ denotes the collection of parameters corresponding to the entire model, this leads to a final expression of:

$$Pr(y|x,\Psi) = \sum_{j=1}^{K} g_j(x,\gamma_j) \sum_{l=1}^{K} g_{l|j}(x,\gamma_{jl}) Pr(y|x,\theta_{jl})$$

(2.3.3)

Logically, we predict the label that is the most probable based on the expression above. Traditionally, a softmax function is utilized at each network as a classifier, and the parameter estimates for models in the nodes are estimated using the expectation-maximization algorithm. It has been noted that HMEs are fit with a focus on prediction as opposed to interpretability (Hastie et al., 2009). In this context, it is reasonable to posit that fitting alternative models in each of the gating networks that also focus on predictive performance may provide an optimal classification rate. A similar argument would apply to the expert networks, a subject we explore in the next chapter.

The example given above is the simplest case, and it is possible that the tree structure can be asymmetric, involve greater than binary splits and have leaves higher in the structure. It has additionally been noted that there is no good method for determining an objective tree topology in the HME framework (Hastie et al., 2009). As shall be seen, our proposed method will offer a possible solution to this problem. Additionally, we shall see that soft-split nature of the predictions offers certain benefits that the hard-splits associated with decision trees do not have.

Finally, it should be noted that there are similarities between the general structure of Figure 2.1 with that of Figure 1.1. One can see a strong analog between the two examples. There is also much of an analog between the HME structure, and the structure described in the hierarchical clustering section. This suggests that it may be possible to construe a representative structure to solve multi-classification problems, for example the diagnosis of a patient condition. In the next chapter we develop a technique to do just that, amalgamating elements from the methods discussed in this chapter.

Chapter 3

Proposed Methodology

"He made an instrument to know / If the moon shine at full or no."
- Samuel Butler

In this chapter we will introduce the concept of *class structure learning.* This method assumes that there is an underlying structure within the classes of the data that aids one in determining how to classify the observations. We will discuss some strategies to determine the optimal configuration of said structure using a measurement of the margin between two classes or groupings as one builds the structure up. We will then propose a modification on these methods by introducing a variation on the value of this margin, and the mechanism by which one predicts the final classification. At conclusion, we will collate these points to give an overarching description of the proposed method.

3.1 Class Structure Learning

The problem of multi-class classification is not a particularly new one. A typical approach is to break the problem into an one-versus-one approach. In this context, one decomposes the problem down into $\binom{k}{2}$ two-class comparisons. This involves training the same number of classifiers, each of which predicts a class label for a given observation. After the predictions are made, the label that receives the most "votes" from the trained classifiers will be assigned to said observation.

This is a viable approach, however it has been previously noted that the potential exists for the approach to be too computationally intensive in application (Yang and Tsang, 2011). The patient diagnosis problem we will discuss in a later chapter is a multi-class problem composed of 287 unique patient diagnoses. The one-versus-one infrastructure would require prediction utilizing $\binom{287}{2} = 40,041$ classifiers for a single observation. This is obviously not a tenable scenario, and a possible solution exists utilizing class structure prediction.

Within the context of the following two papers that we will discuss, as well as this thesis, the class structure is assumed to be a binary decision tree. An example is given in Figure 3.1.

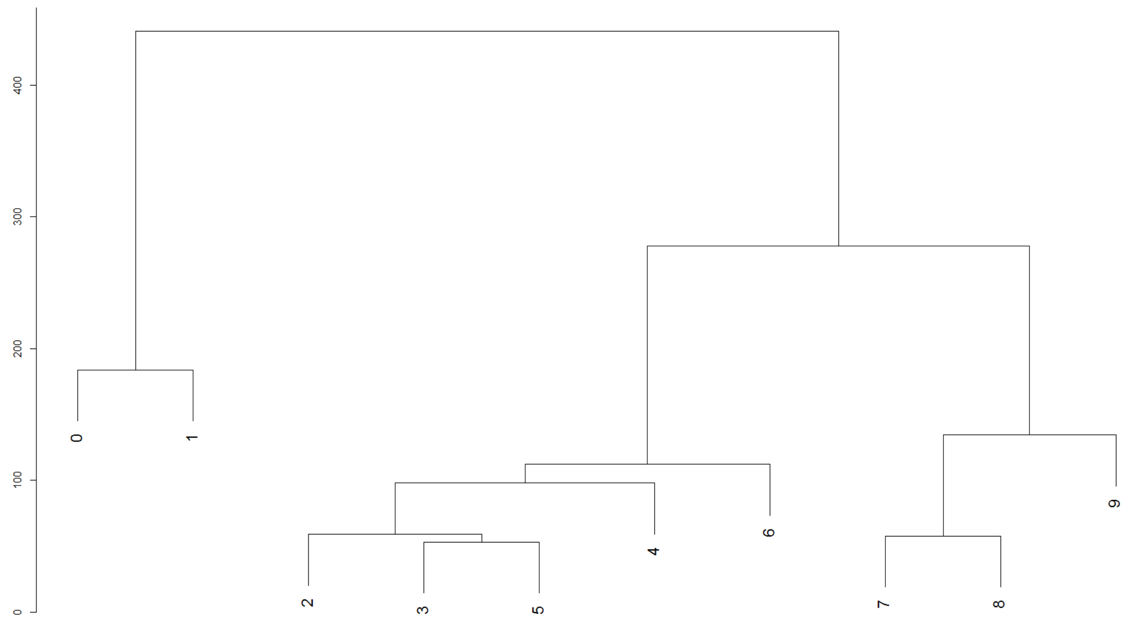

Figure 3.1: An example of a class structure tree

Here, there are ten classes from zero to nine. Using the one-versus-one technique, we would have to fit $\binom{10}{2} = 45$ classifiers. If we fit a classifier at each of the splits of the tree in Figure 3.1 we can reduce this number to $(k-1) = (10-1) = 9$ classifiers, a fifth of the amount needed in the non-structured case.

So, there is sufficient motivation to utilize class structure learning. A reasonable question now becomes how to actually establish such a structure. There have been several attempts to establish how to optimally grow the class structure tree, and we highlight two such instances here.

3.1.1 Hierarchical Maximum Margin Learning for Multi-class Classification

This paper authored by Jian-Bo Yang and Ivor Tsang from Singapore in 2011 proposed a divisive clustering method to construct the structured classification tree. As we reviewed in Section 2.2, divisive clustering builds a hierarchical tree based on differences between observations. In the context of their problem (and ours), we do not consider the differences between individual observations, rather we want to differentiate between the classes.

A metric that has been proposed by both Yang and Tsang (2011) and Tibshirani and Hastie (2007) is to utilize the maximum margin found by the solution to the support vector machine problem for the difference between individual classes, or nodes within the class structure tree splits.

For the purpose of the discussion, let us explicitly establish the class structure tree defined by Yang and Tsang (2011, pg.753):

> **Definition** A tree of k-class structure has d-layers. Its root node is $\{1, 2, \cdots, k\}$. At layer t, $\forall t = 1, 2, \cdots, d$, there are n_t node(s) where $n_t \geq 1 (n_1 = 1)$ and each node $G_i^t \subseteq \{1, 2, \cdots, c\}, \forall i = 1, 2, \cdots, n_t$ is a group of classes. This tree is constrained by: 1) each non-leaf node has two children and each child node is a subset of its parent node, and 2) the nodes in the same layer $t, \forall t > 1$, have non-overlapping class indices, i.e. $G_i^t \cap G_j^t = \emptyset$ for $i \neq j$, where $1 \leq i, j \leq n_t$. 3) G_i^t is non-empty. 4) $G = G_1 \cap G_2$. 5) All leaves are singletons.

Now, consider any non-leaf node G, which contains k^* classes, and k^* is a subset of the total number of classes k. Now, G's child nodes G_1 and G_2 are determined by:

$$(\mathrm{G}_1, G_2) = \underset{\tilde{G_1}, \tilde{G_2}}{max} \Big\{ J(\tilde{G_1}, \tilde{G_2}) | \text{all possible} (\tilde{G_1}, \tilde{G_2}) \Big\},$$

(3.1.1)

where $J(\tilde{G_1}, \tilde{G_2})$ is the margin that is found by the support vector machine discussed in section 2.1. What (3.1.1) amounts to is calculating $2^{(k^*-1)} - 1$ margins, and choosing the (G_1, G_2) that is the largest. The intuition is that the classes that belong to G_1 share more in common than those in G_2. This process is continued until the child nodes correspond to the leaves of the class structure tree.

There is a severe computational cost to this procedure, especially initially in the process where all k classes are still contained in the largest node, as one needs to consider all possible combinations of the k classes in the two child nodes, G_1 and G_2. To combat this, the authors offer a similar criteria to (2.1.10) by defining an additional label variable $z \in \{-1, +1\}$ for each observation $x_i, \forall j \in \Gamma \equiv \{i | y_i \in G\}$. This modifies (2.1.10) to

$$\underset{z \in Z}{min} \underset{\beta,\xi}{min} \tfrac{1}{2} ||\beta||^2 + \tfrac{C}{2} \sum_{j \in \Gamma} \xi_j$$
$$\text{subject to } \mathrm{z}_j \beta^T \phi(x_j) \geq 1 - \xi_j, \forall j \in \Gamma$$

and any class $\omega_l \in G, l = 1, \cdots, k^*$, is put in G_1 or G_2 by:

$$\begin{cases} \omega_l \to G_1, if z_j = 1 \\ \omega_l \to G_2, if z_j = -1 \end{cases}$$

In this case, z_i represents what child node an observation should belong to. The authors go on to describe a process to determine the optimal $\mathbf{z}$ utilizing the *cutting plane algorithm* (Kelly, 1960), and multiple kernel learning using the *SimpleMKL* method (Lanckriet et al., 2004; Rakotomamonjy et al., 2000). Using this modified process, the authors then grow the class structure tree, fitting the maximum margin classifier at each node.

The authors compared this method against several others, including the one-versus-one method, on several benchmark machine-learning problems and found that their method either performed the best, or was competitive, particularly when utilizing a Gaussian kernel in the support vector architecture. This suggests that there is both a computational, and predictive benefit to utilizing class structure trees.

Although this result does indicate a certain optimism to using class structure learning, it does not definitvely answer how to build the structure. Yang et al.'s process does build the tree in the top-down manner, using the divisive paradigm. However, it is not certain that this method is better than an agglomerative approach. The next paper we discuss, looks at this question in great detail.

3.1.2 Margin Trees for High-Dimensional Classification

Similar to Yang and Tsang (2011), Tibshirani and Hastie (2007) focus on using the margin of the support vector machine to grow the class structure tree. The authors build the structure using three different clustering methods: complete linkage, single linkage, and using a proposed "greedy" algorithm, which we will describe below.

In both the single and complete linkage methods, the authors first train $\binom{k}{2}$ support vector machines, which are the pair-wise comparisons of the k classes. The size of the margins are utilized as the measurement of how "different" two classes are. This provides an overall measurement of how different each class is from each other. At this point, one applies the single- or complete-linkage clustering algorithm, successively merging classes and groups based on the minimum dissimilarity (least different) or maximum dissimilarity (most different) until all of the k initial classes are merged together in the largest cluster at the top of hierarchical tree.

The authors also propose a hybrid top-down and bottom-up method using a "greedy" approach. They begin with the same idea considered in Yang and Tsang (2011) in that they consider all partitions between the classes possible, and then choose the split that has the maximum margin among them. As before, for a node that contains k^* classes, this requires the consideration of $2^{k^*-1} - 1$ margins. For the multi-class problems, this can quickly become computationally strenuous. To compensate for this they combine the idea of computing the maximum margin amongst all the possible groupings with the construction of the complete-linkage hierarchical clustering tree. We outline the procedure below from Tibshirani and Hastie (2007).

1. Construct the complete linkage clustering tree based on the $\binom{k}{2}$ margins.
2. Starting with all classes at the top of the tree, find the partition of each individual class versus the rest, and also the partitition that produces two classes in the complete linkage tree (that is, make a horizontal cut in the tree to produce two classes). Let M_0 be the largest margin achieved amongst all of these competitors.
3. Cut the complete linkage tree at height M_0, and collapse all nodes at the height.
4. Consider all partitions of all classes that keep the collapsed nodes intact, and choose the one that gives the maximal margin $\hat{M}$.

This process cuts down considerably on the number of margins one needs to calculate in order to construct the class structure tree in the greedy fashion. The authors then applied their technique to a number of cancer-classification studies, that tried to classify cancer types using gene expressions. The number of features ranged from 2308 to 16063 and the number of observations ranged from 22 to 261. These multi-class problems also ranged from $k = 9$ to $k = 14$, and they also compared their method with the one-versus-one support vector machine framework, and a previous method developed by the authors called "nearest centroids". The findings were that all three of the class structure methods as well as the one-versus-one method performed roughly the same, and better than nearest centroids.

One notable point that the authors make in the discussion is that they have restricted their attention to the cases where there were many more features than observations ($p >> N$). This means that the classes are separable by a hyperplane, specifically a linear support vector machine, further nothing that "When $p \leq N$ and the classes may not be separable, our approach can be modified to work in principle but may not perform well in practice." (Tibshirani and Hastie, 2007, pg. 650) In the next section, we attempt to do just that.

3.2 Description of the Proposed Method

We now briefly describe an overview of our proposed method, and attempt to tie it back to work previously cited in chapters one and two. First, we describe the construction of the class structure tree, and define the difference metric we utilize in the study. Finally, we describe the method utilized in the top-down classification of the data, as an observation is assigned a class label.

3.2.1 Construction of the Class Structure Tree

In this work we utilize the agglomerative paradigm that was reviewed in section 2.2 in the same manner as Tibshirani and Hastie (2007). We first construct $\binom{k}{2}$ SVM classifiers for each pairwise comparison i and j between the k classes, $i \neq j$. However, we do not always take the maximum margin found by the SVM between the two classes as the measurement of the difference between the two classes. Explicitly, we fit the maximum margin classifier, utilizing a linear basis kernel (e.g. the original feature space) and a cost parameter C of 1. Then, for a given pairwise comparison define the difference between these two classes as D_{ij}. Further, let D_{ij}^{LS} be the case where the data are completely linearly separable, and D_{ij}^{NLS} be the case where they are not. Then, the difference is given as:

If there are no misclassified observations in the training examples

$$D_{ij} = D_{ij}^{LS} = \frac{2}{||\beta||} + \frac{\text{number of correct classifications}}{\text{total number of predictions}}$$

If there are any misclassifications in the training examples

$$D_{ij} = D_{ij}^{NLS} = \frac{\text{number of correct classifications}}{\text{total number of predictions}}$$

The first definition is the sum of the maximum margin found by the SVM and the correct classification rate, which is always unity. The second definition has to be less than one, and it is easy to see that, since $\frac{2}{||\beta||} > 0$:

$$D_{ij}^{LS} > D_{ij}^{NLS}$$

By this definition, any two classes i and j in the training examples that can be completely linearly separated by a hyperplane will always be defined as being more different than any two that cannot be. This is in contrast to the work of Yang and Tsang (2011) and Tibshirani and Hastie (2007) where this measurement of the difference was entirely dependent on the maximum margins. It is possible to have a large margin with many support vectors, and have training data that are not well-classified by said margin. Our metric incorporates the idea of two classes being more "different" given a larger margin along with the effectiveness of said margin separating the observations.

Now, we can construct the D_{ij} into a $k \times k$ matrix, and apply a chosen agglomerative clustering algorithm. For the purpose of the zip code data, we will compare the performance of the three paradigms described in section 2.2, and for the purpose of the emergency department data, we will utilize the single linkage clustering algorithm for a reason that will become clear. Finally, once we have constructed the topology of the class structure tree, we now construct a classifier at each node of the splits in a top-down manner in order to interpret the prediction of the final structure of the tree, as in Tibshirani and Hastie (2007). We now deviate. Where both Tibshirani and Yang constructed a SVM at each of the nodes of the tree, we do not limit ourselves to such a model. We allow any classifier at any node, in particular any classifier that gives a prediction of the probability of belonging to a given class, or child node.

Now, within this framework, each class label has a unique path in the class structure tree, and any observation with this class label must belong to a specific set of nodes. Define this set of nodes as G_l, and each node within as G_{lj}. Thus, for a given class label l, it will have a $G_l = \{G_{l1}, G_{l2}, \cdots, G_{lj}, G_{l(j+1)}\}$, where $G_{l(j+1)}$ indicates belonging to the singleton, l. Then, for a given class label, l, we calculate the probability of an observation, x_i, belonging to class l as

$$\mathrm{P}(\mathrm{x}_i \in l) = \prod_{k=1}^{j} P(x_i \in G_{l(k+1)} | x_i \in G_{lk})$$

(3.2.1)

Equation (3.2.1) is inspired by the hierarchical mixture of experts from section 2.3, and the soft-split prediction at each of the gating networks. Here we can consider each of models fit at each of the nodes as an "expert". What this means is that we can also calculate the probability of any class label $l = 1, 2, \cdots, k$, and subsequently rank them. This constitutes the final step of the method: we simply predict the class label, l, that is the most probable amongst these labels. However, we also retain the remainder of the predictions, and know the second-most probable label, the third-most, and so on. This information is beneficial, as we shall demonstrate in a later section with the emergency department data. One can see an overview of the method proposed in algorithm 3.1 below.

We now present an application of the method to the zip code data discussed in chapter one. This dataset was chosen for its accessibility and completeness. It also offers the advantage that Yang and Tsang (2011) applied their method to it, so we can use their result as a basis of comparison.

3.3 Application of the Method to a Benchmark Dataset

To refresh the reader, the zip code data was composed of 7291 training observations, and 2007 test observations. Each of the observations was a 16×16 grey-scale image of the digits 0 to 9, and were processed specifically for analysis. Some notes: for each of the $\binom{10}{2} = 45$ SVM comparisons used to construct the class structure tree, we used a cost parameter equal to one, and a linear basis kernel. We also weighted the observations to counteract unbalanced node membership sizes at each split. We

Algorithm 3.1: Overview of fitting the structure

```
Part A: Fitting the structure
Step One: Construct $\binom{k}{2}$ SVM models for one-versus-one class comparisons.

Step Two: Calculate the distance $D_{ij}$ for each of the $\binom{k}{2}$ class comparisons.

Step Three: Amalgamate the distances into a $k \times k$ distance matrix.

Step Four: Use the distance matrix to construct the hierarchical cluster tree structure.

Step Five: For each of the splits dictated by the topology of the tree, place a classifier
           to decide which node an observation should belong to.

Part B: Observation classificaton
Step One: Obtain the probability of belonging to each node that leads to the terminal
          classification.

Step Two: Calculate the probability of belonging to a particular classification
          by calculating the conditional probabilities of belonging to each node on the path.

Step Three: Rank the class labels from most probable to least probable.

Step Four: Predict the most probable class label.
```

constructed three different class structure trees: one using single-linkage, one using complete-linkage, and the last using the average algorithm, as outlined in section 2.2.

One can refer to these different trees on the following page. A common theme by all three methods was to merge zero and one late in the process, showing they were quite different than the other digits. Additionally, all three methods found a cluster that included (2, 3, 4, 5), and one that was composed of (6, 7, 8, 9). The largest difference in the configuration seemed to be when individual digits were split into singletons. The complete method seemed to produce the "balanced" tree, as compared to the "stringy" appearance we see in the single-linkage tree, a general sentiment echoed by Tibshirani and Hastie (2007). This impression also seems to have been created by the tree created by the average method.

If the reader will refer to Table 3.1 we compare the performance of the various clustering methods against the result found using the maximum separating margin (MSM) technique in Yang and the one-versus-one method (again using $C = 1$). What we see is that our method is competitive with the MSM method in all three of the linkages examined, and all of the class structure methods were outperformed by the one-versus-one setting.

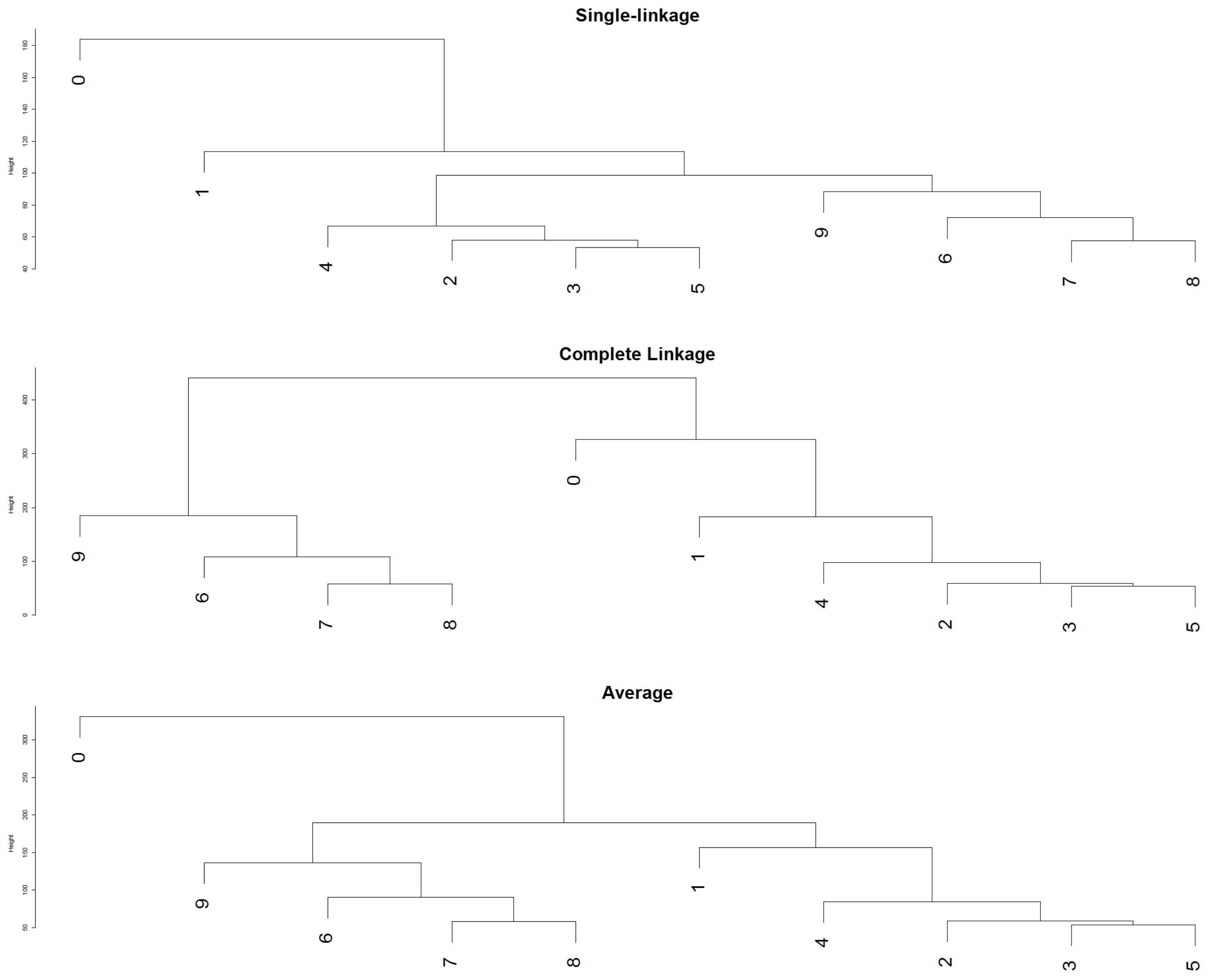

Figure 3.2: Class structure trees produced by the three clustering algorithms

Method	Training Classification	Test Classification
Single Linkage	95.57%	85.35%
Complete Linkage	93.29%	84.35%
Average	95.31%	84.90%
MSM	Unknown	84.30%
1vs1	99.99%	92.97%

Table 3.1: A comparison of correct prediction rates between the different methods

Of the three clustering algorithms, the single-linkage seemed to do the best, so we will focus on using this class structure tree for the remainder of the discussion. Below, in Table 3.3 show the correct classification rate of each node model when assigning to their child node or class label. The nodes are numbered in a top-down manner, such that the first split from the cluster that contains all the classes is "one", and the second split is "two", and so on. Addtionally, we see from the first split of the single-linkage tree that node one splits the digit (0) from (1, 2, 3, 4, 5, 6, 7, 8, 9). In the same way, node two splits the digit (1) off from the remainder of the digits (2 through 9).

Node	Training	Test
One	0.999	0.992
Two	0.999	0.993
Three	0.911	0.897
Four	0.807	0.784
Five	0.764	0.766
Six	0.699	0.697
Seven	0.873	0.850
Eight	0.811	0.815
Nine	0.683	0.677

Table 3.2: Correct classification at each node of the single linkage tree

As one can see, the classifiers higher (one and two) in the tree correctly predict the vast majority of the observations. As we descend down the tree we reach the clusters that contain observations that were judged by the algorithm as being the least different. Concomitantly, the correct classification rate drops. Table 3.3 shows the correct classification of each of the digits.

Digit	Training	Test
Zero	0.999	0.953
One	1.000	0.943
Two	0.940	0.773
Three	0.961	0.825
Four	0.914	0.790
Five	0.923	0.756
Six	0.920	0.829
Seven	0.971	0.850
Eight	0.899	0.765
Nine	0.960	0.904

Table 3.3: Correct classification of each label in the single-linkage tree

We note that the digits we identified as being merged last (0 and 1) have excellent rates of correct classification. However, digits that we found were alike (6, 7, 8, 9), and (2, 3, 4, 5) showed lower rates of correct classification. This is to be expected, as a given classifier would have more trouble differentiating between those classes that were more alike, and thus further down in the class structure tree.

Digit	Training Rate	Test Rate
Zero	0.999	0.978
One	0.996	0.950
Two	0.990	0.924
Three	0.991	0.916
Four	0.983	0.900
Five	0.984	0.900
Six	0.991	0.935
Seven	0.984	0.918
Eight	0.980	0.886
Nine	0.981	0.972

Table 3.4: Classification rates of digits using a radial basis SVM at each node

After much trial-and-error, the optimal model at each node seemed to be a SVM with a radial basis kernel using a cost parameter of 1, and a gamma parameter of 0.00390625. The overall correct test classification rate was 93.37%. Table 3.4 outlines the classification rate for each of the digits. Although the prediction rate has improved, the effect where higher nodes tend to predict more of the observations correctly than those lower in the class structure tree persists. This seems to not be a phenomenon unique to our method, as the concern was echoed by Tibshirani and Hastie (2007).

3.3.1 The Ranking of the Class Labels

It should be noted that the method as outlined above officially predicts the class label as the most probable one of all those considered. However, in step eight of algorithm 3.1, we rank the class labels from most probable to least probable. If there is truly an underlying structure within the data, it is reasonable to expect that expanding the prediction to include the increasingly probable class labels will increase the classification rate. In the case of our initial single-linkage class structure tree, using only the linear basis kernel at each node, if we expand to the top three most probable digits, the method includes the right digit in 96.26% of cases. In the radial basis case, the rate was 98.11%.

This may not seem to be applicable in the classification of hand-written digits, and one would be right. However, in the case of the emergency department data, the application of this expansion of choice in the class labels serves a viable purpose. Specifically, each patient may only have a single diagnosis, and ultimately it is the physician that must choose it. This perceived over-inclusion actually serves the purpose of providing the physician additional information to aid in their decision-making process. Specifically, it gives them the probability of a particular diagnosis, which they may correlate with the patient's pre-test probability of having the same pathology, a feature that we shall see in the emergency department data.

3.3.2 The Effect of the Derived Class Structure

A final point of consideration is whether or not the structure of the tree enabled the best predictive value when compared to alternative structures. In an effort to respond to this question, we re-ran the model-fitting process, randomizing the class labels that were placed in the terminal experts. We then used the new model to predict the test set data. We repeated this procedure fifty times. The original structured model performed worse in thirty of the fifty cases. The maximum predictive performance was 88.14%, and the minimum was 82.86%, with a mean correct classification rate of 85.79%. Our original fitting process does not seem to have a strong effect on the predictive performance of the model. However, it is unknown whether this is due to a lack of hierarchical structure in the zip code data or due to the algorithm itself.

Chapter 4

Application to the Emergency Department Data

"There's more beauty in truth, even if it's dreadful beauty."
- John Steinbeck

We now present the results from the daunting 287 diagnostic label problem. One can refer to Figure 4.1 on the next page for the class structure tree. We can see that the mergings tend to exhibit the stringy nature that we have come to expect by now. The last merge occurred between "diabetic ketoacidosis" and the remainder of the diagnostic labels. This is probably due to a combination of an abnormal blood glucose level, and an acidic blood pH level that is not typically present in most diagnostic morphologies.

The results are presented in the Table 4.1 below. We utilized the method outlined in chapter three, again using the SVM with a linear basis kernel, and a cost parameter of 1, both to construct the class structure tree, and for the models at each of the nodes. As one can see, the predictive performance of the algorithm is markedly worse than in the zip code example. It should be noted that the performance is still measuredly better than random guessing, which would correspond to a classification rate of 0.35%. If one would predict the most numerous class label would achieve 7.18%, so the method is still better than that. However the sensitivity of such a result would not be useful in practice to a physician. It is informative to view the correct classification rates of each of the diagnostic labels, which are placed in the appendix.

The most commonly cited diagnosis by the algorithm was abdominal pain not-yet-diagnosed. This is not an entirely surprisingly result. Anecdotally, roughly a fourth of all emergency department presentations are for abdominal pain. A large portion of these are undiagnosed as the patient improves spontaneously in the course

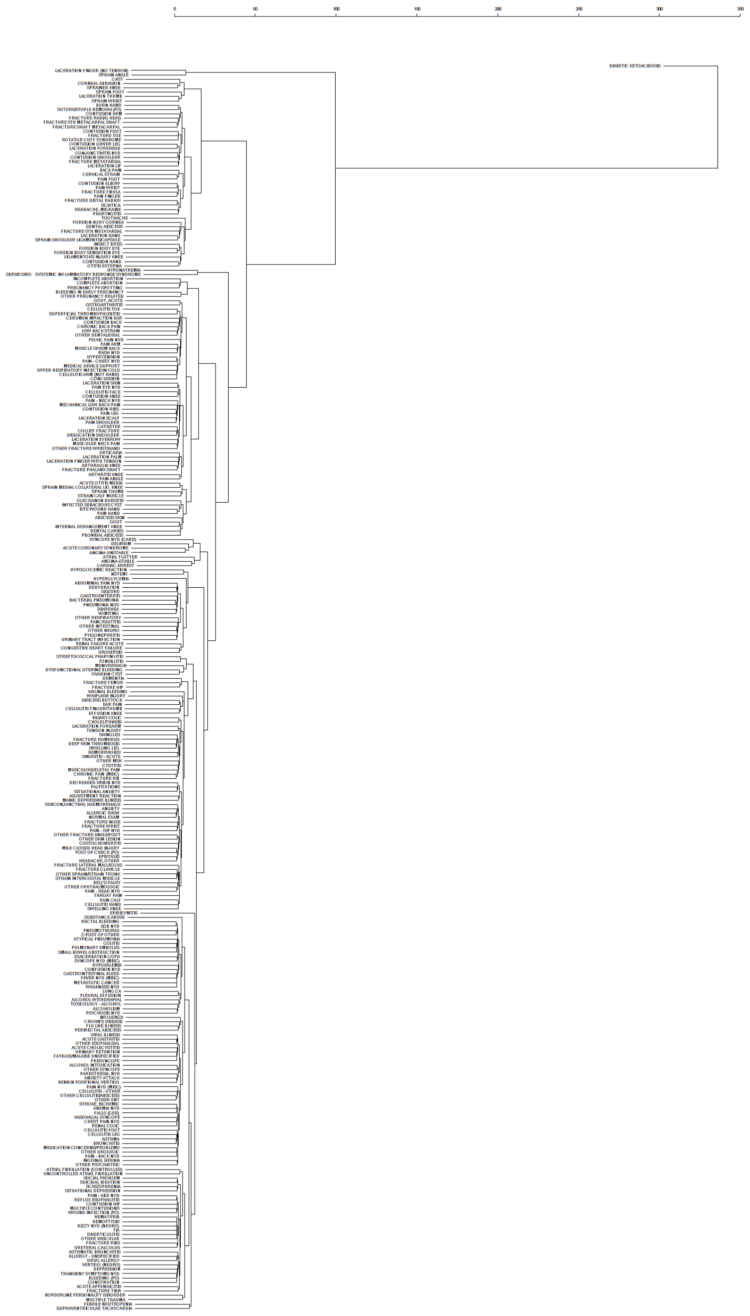

Figure 4.1: Class structure tree of the emergency department data

		Correct on first rank	Correct in first ten ranks
Training set	Absolute number correct	8849	31117
(n = 87209)	Classification rate	10.15%	35.68%
Test set	Absolute number correct	2175	7470
(n = 21661)	Classification rate	10.04%	34.49%

Table 4.1: Results from the EDIS data

of their care without explanation, or are discharged from the department with the expectation they will return if their conditions worsens. This phenomenon is reflected in this class label being the most populous in our dataset. Given so many examples, it is encouraging that the method identified over ninety percent of these diagnoses in both the training and test sets.

A similar phenomenon is observed in the diagnosis of chest pain not-yet-diagnosed for the same reasons cited above, although the method was not quite as successful as above. It would seem that a large number of examples for each class label is required to improve predictive performance. As noted above, the remainder of the diagnostic labels were not so easily identified, which is probably a result of a combination of a sparsity of training examples, as well as inappropriate features within the data. We shall explore this in the discussion.

Although the class structure tree only predicted the right diagnosis as most probable 10.04% of the time, it is heartening to see that the tree correctly identified the right diagnostic label in the top ten more than a third of the time. This may be an indication that there is some hierarchical structure in the data, and that we may be able to use it to accurately predict patient diagnosis. If the reader will refer to Figure 4.2 it shows the classification rate as we consider an expanding ranked list of probable diagnoses. One can see that there is a steep initial gain in the correct classification rate, and it slows as the number of ranks increases.

This suggests that there is an effect conferred on the larger problem by our probabilistic approach. Even if the method does not correctly identify the right diagnosis in the first instance, the probabilistic predictions of the method show a good chance

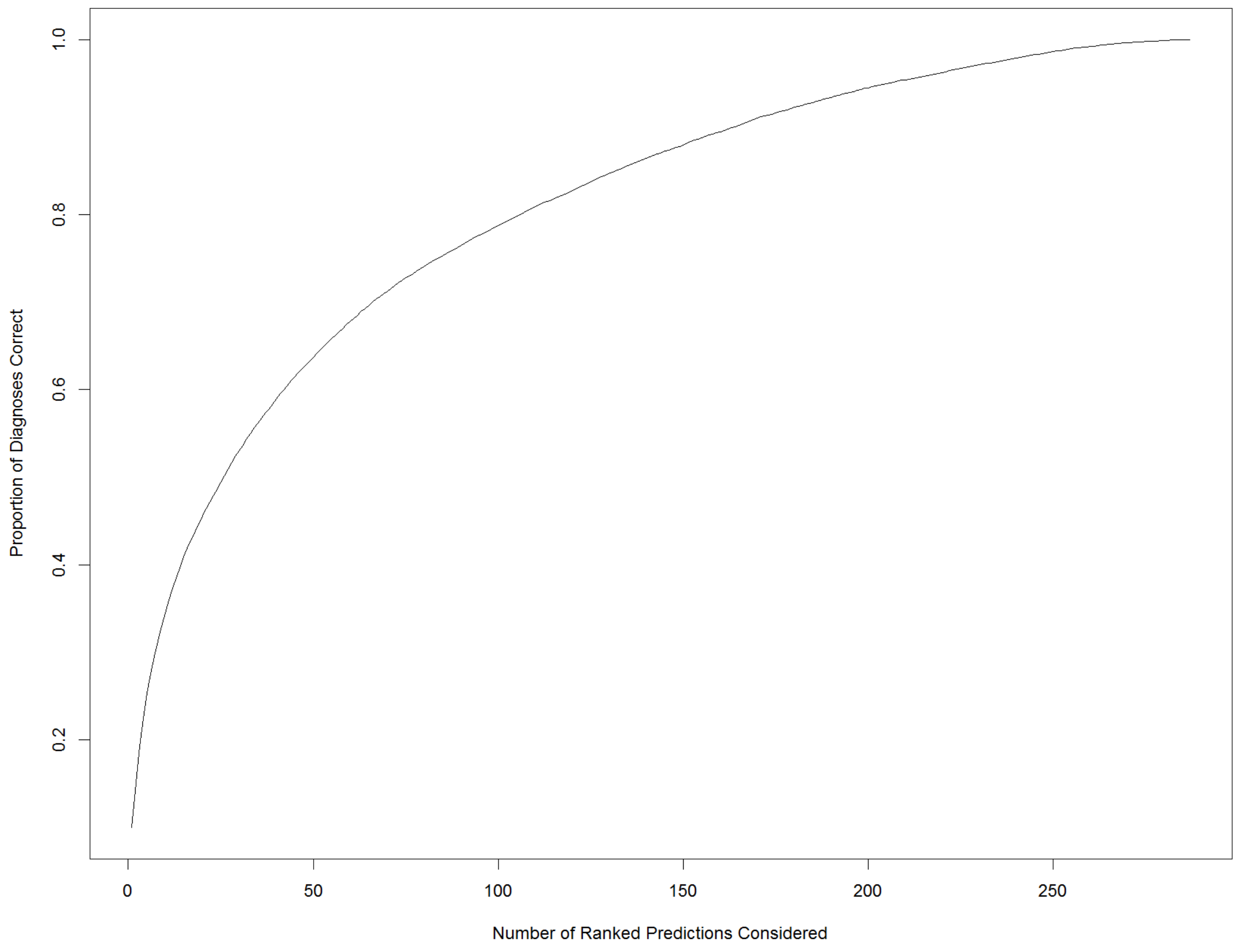

Figure 4.2: Classification rate versus number of ranked diagnoses considered

in ranking the right diagnosis closer to the top of the list than the bottom. In application, this allows the clinician to pick from a number of ICD-9 classes that the method has deemed consistent with the patient record up to this point. However, one cannot ignore that it is still ideal for the class structure tree to predict the right diagnosis as the most probable. This focus may also rank the right diagnosis higher on the list, even if it does not obtain the top ranking. In the next chapter we discuss reasons the method may not have performed this task, and what one should do in order to improve the accuracy of it.

Chapter 5

Discussion

"If you don't know, the thing to do is not to get scared, but to learn."
- Ayn Rand

The effect of our approach is competitive with the approach suggested by Yang and Tsang (2011), and consistent with the impressions given by Tibshirani and Hastie (2007). However, some cautious ruminating may be warranted about our results. One point emphasized by the proposed technique is the underlying complexity in determining the class structure tree, the models fit at each of the splits, and how to choose the parameters for the models that we have chosen.

Ideally, there would be an objective way to do so. The best case scenario would be to simply rely on the subject-matter expert to settle the matter, such as the example presented in the introduction. Indeed, Vapnik himself suggested that the parameters of the SVM model should be set by the expert directly (Vapnik, 1999). However, an intellectual exercise demonstrates the limitation of this approach. It is unlikely that a physician would be able to provide a logical class structure for approximately one thousand different diagnoses, and simultaneously be able to suggest a logical value for the cost parameter C in each of the SVMs, as an example.

One would be able to heuristically dictate that abdominal pathologies likely are clustered together. In a higher node, psychiatric disorders would likely congregate with acute intracranial pathology. However, when differentiating within abdominal pathology, it becomes much more difficult to differentiate between acute cholecystitis and transient cholelithiasis, for example. Similarly, it may be more difficult to differentiate the structure between a subarachnoid hemorrhage and a subdural hemorrhage for the confused patient with head trauma.

In the author's opinion, the grand advantage of machine-learning is to perceive patterns where the human mind may miss them. This notion is at the heart of our methodology. The groupings seem to theoretically offer accurate predictive performance, which is the optimal outcome any physician seeks in the evolution of a patient's care plan.

Much of the discussion in the preceding sections focus on the efficency of the structured learning. An important point to make is to illustrate the difference between training the model and using said model to predict class labels. Training the class structure tree as described is labourious in nature. First, one must train $\binom{k}{2}$ SVM comparisons to determine the tree topology. Following this, one must place a classifier *of one's own choosing* at each node.

Since one fits a classifier at each node, there is a flexibility granted that is not offered by previous work in this area. By optimizing the performance of a given classifier, one improves the performance of the overall model. One cannot stress the importance of each step, as it is likely that performance in nodes at greater height may have drastic effects on the child nodes below. In this particular case, it indeed takes a village to predict accurately.

This amounts to tailoring each node classifier for each classification task. Good prediction requires significant handiwork on the part of the statistician. Once this work has been put in, prediction is faster than the non-structured methods, and competitive in predictive performance, which is a significant consideration in application.

However, one cannot discount the challenge of constructing the model. Significant time was spent utilizing the grid-search method to find the optimal parameters of the SVMs fit at each of the nodes of the zip code example, however it has been noted that " . . . such searches may be computationally expensive and the precision of the results is subject to the chosen granularity of the grid."(Boardman and Trappenberg, 2006). This suggests that it may be beneficial to consider attempts to objectively select these parameters, such as in Boardman and Trappenberg (2006) or Chapelle et al. (2002).

An interesting phenomenon that arises from the soft-split nature of our method is that modifying the performance of a given child node also has an effect on the entire prediction. As an example, if the class structure tree gave the most probable digit as being zero in our zip code example above, and we changed the classifier at a deeper split, it is wholly possible that the most probable prediction could change to another digit. This is a phenomenon not present in the hard-split nature of the other techniques we have discussed. In those frameworks, if the initial prediction was a zero, no effort on the deeper splits would ever change this label assignment.

I feel this demonstrates a salient and relevant feature of the method. The ability to indicate each diagnosis by the probability that it can occur is informative, and may reflect the overall structure of pathology in the emergency department population in general. Although it is ideal that the method predict the right diagnosis as the most probable, it is important in clinical application to handicap the chance of other diagnoses for a particular patient presentation.

5.1 Limitations

Although the author is an eternal optimist, one cannot ignore a determinable fact: one cannot methodology away irrelevant features. Previous work by the author has accentuated that one is only as good as one's data (Butler et al., 2014). In our patient data, complications arise in the data itself. As a working example, consider the patient with chest pain, who may have acute coronary syndrome.

Standard of care dicates that said patient should have serial troponin levels drawn (Alpert et al., 2000; Collinson et al., 2013; Diercks et al., 2013). Troponin is a cardiac enzyme that is only detectable in the blood should damage occur to the myocardium. However, it is possible that a patient has a spurious positive result. So, samples are taken repeatedly to ensure the authenticity of the result.

Our dataset only utilizes the first troponin level (if one was drawn at all). Our final data extract from the Department of Emergency Medicine is extremely inclusive, with over 800,000 patient visits. This shifts the onus on the author to process the data such that the relevant features are available to the methodology. Addtionally, it can be seen in the test set of the EDIS data that many of the musculoskeletal pathologies (e.g. fractures and sprains) are ill-detected. An important feature of musculoskeletal injury management involves diagnostic imaging. With this in mind, although the result of each diagnostic imaging test is available, it is in a free-text format. Some efforts on developing text-mining methods to derive relevant patient features from this information. Moreover, to be of utmost utility, it may be more informative to derive said features from the images themselves. As noted in the first chapter, Klöppel et al. (2008) do exactly this with functional MRI images.

Another example of the importance of relevant features that is easily cited from our data is that of the diagnosis of alcohol intoxication. In our data, patients that were given this diagnosis had an alcohol level drawn 137 times (33.7% of the time). This is in contrast to those in all other diagnostic labels, who had an alcohol level drawn 336 times (a mere 0.4% of the time). It is obvious that the presence of this positive value contributes to the correct classification of a patient that is intoxicated in the ED. This is probably the case with many of the diagnostic labels, and the patient features. This only further enforces the necessity of good feature selection.

A point of consideration is to also acknowledge that our method may ". . . impose hierarchical structure whether or not such a structure exists in the data."(Hastie et al., 2009) We have made the tacit assumption that the method of differential diagnosis is predicated on the variables we have included in our dataset. A different structure may result with the inclusion of further data elements, as well as more training examples. The structure we have found may not be the optimal one, and healthy skepticisim is warranted until we can improve on our results.

Finally, it is unknown whether or not this method would be generalizable to other patient populations. The method was trained specifically on adult patients that presented to emergency departments in the most densely populated portion of Nova Scotia, one of which is the leading trauma and tertiary care centre for the Maritimes in Canada. The result may not be applicable to patients in the pediatric set, as well as centres that predominantly serve a rural area and may not have access to the full-range of diagnostic testing available at CDHA facilities.

5.2 Future Research

This study represents the foundation of a comprehensive research program here at Dalhousie University. Dalhousie is intimately linked with the Department of Emergency Medicine at the Capital District Health Authority, since this is the teaching hospital for the medical school. This association provides the unique opportunity to translate research into practice at the emergency department.

This coincides with an increased prevalence of the electronic patient record (Baker et al., 2014). As electronic systems proliferate, the opportunities to utilize these readily available data will increase the importance of this research. A great effort must be expended to ensure that a system built in Halifax is generalizable to other parts of the nation, and the world.

Another aspect to explore is that the method described has limited itself to binary splits at each of the nodes. It is feasible that the class structure applicable to this problem may not be relegated to such a confined structure. Generalization of the technique above to multiple splits at each node may represent increased flexibility. Tibshirani and Hastie (2007) also note that "The construction of the margin tree could also be done using other kernels . . . however in the p > N case considered in this paper, a linear SVM can separate the data, so the utility of a non-linear kernel is not clear." Since we have constructed the class structure tree in a similar manner, this is a question that is well worth answering. In particular, our distance metric emphasizes the difference between linearly separable (e.g. very different) classes, and those that are not. If, in an enlarged feature space, the classes would be more separable it would lead to a much different configuration of the class structure tree. Whether or not this mitigates the misclassification rate is unanswered.

Additionally, it is likely that for each node, not all of the data are applicable to the decision a physician would make at that step. We may gain predictive and computational benefits from limiting the amount of features we consider at any given step. Tibshirani and Hastie (2007) explore this very issue noting that " . . . it would be clearly beneficial to reduce the set of [features] to a smaller set, if one can improve, or least not significantly worsen, its accuracy." They apply a method termed "hard thresholding" to several datasets, and show experimentally that there is some benefit to their procedure, with very little loss in accuracy.

As cited above, it is probable that the method would achieve greater performance with more training examples. We are in the process of acquiring approximately 500,000 more records to add to our dataset, one that will be more comprehensive than presented in this study. It is only important to note that this dataset does not include results from diagnostic imaging. This is an important part of the diagnostic process, and determining a way to include these results will only aid in our efforts. The combination of more data and a more robust collection of patient features will allow us make firmer conclusions about the validity of the methodology.

The close association with the Department of Emergency Medicine above allows for the opportunity of online learning, wherein the method is constantly learning and adapting to data as it is collected by EDIS. This interactivity would fine-tune the method as time elapsed, constantly improving it's prediction. In the long-term, integration of the method into the EDIS system would make the tool available to the physicians. It also allows the system to be reactive to changes in the underlying patient population. Monitoring of the system output may allow us to determine when the system is detecting novel presentations in the emergency department. This may allow us to delineate when diagnoses that have not yet presented to the emergency department are arriving in force, so there may be epidemiological applications to the method as well.

Additionally, once the learning method has been tuned to a reliable amount of sensitivity and specificity, there may be ready applications to bench clinical research. Consider that there are many patients with the diagnosis “abdominal pain, not yet diagnosed”. A study of great value would be to see how many of these patients return and subsequently are diagnosed with definitive abdominal pathology, then compare with how our method predicted the original visit. Such a metric may allow clinicians to determine what factor may lead to the definitive diagnosis on the original visit.

5.3 Conclusion

The candidate structured learning method is competitive with non-structured learning methodology in predictive performance, with a performance of 93.37% in the test set of the zip code data. The flexibility of fitting individualized classifiers at each of the nodes increase predictive performance, allowing us to raise the predictive performance by fitting different models to each of the nodes in the class structure tree.

In the emergency department data the predictive performance was 10.15% in the training set, and 10.04% in the testing set. The use of structural class learning provides a foundation for high-dimensional multi-class learning. Future directions for the research may allow us to improve the predictive performance and the efficiency of the learning method.

Bibliography

JS Alpert, Thygesen K, Antman E, and JP Bassand. Myocardial infarction redefined–a consensus document of the joint european society of cardiology/american college of cardiology committee for the redefinition of myocardial infarction. *J Am Coll Cardiol*, 36(3):959–69, Sept 2000.

DB Baker, JB Perlin, and J3 Halamka. Evaluating and classifying the readiness of technology specifications for national standardization. *J Am Med Inform Assoc.*, May 2014.

M Boardman and T Trappenberg. A heuristic for free parameter optimization with support vector machines. 2006 International Joint Conference on Neural Networks, 2006.

M Bullard, B Unger, J Spence, and E Grafstein. Revisions to the canadian triage and acuity scale (ctas) adult guidelines. *CJEM*, 10(2):136–142, 2008.

TE Burroughs, AD Waterman, TH Gallagher, B Waterman, DB Jeffe, WC Dunagan, J Garbutt, MM Cohen, J Cira, and VJ Fraser. Patients' concerns about medical errors during hospitalization. *Jt Comm J Qual Patient Safety*, 33(1):5–14, 2007.

M Butler, S MacPhee, and S Newton. Chirpp: A characterization of captured injuries versus uncaptured injuries for patients presenting at a pediatric tertiary care centre. *CJEM*, 16, 2014.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.

O Chapelle, V Vapnik, O Bousquet, and S Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 2002.

Longfeng Chen, Guixia Kang, Xidong Zhang, Lichen Lee, and Xiangyi Li. Hybrid decision making in the monitoring of hypertensive patients. *e-Health Networking, Applications and Services (HealthCom), IEEE 14th International Conference*, pages 32–37, 2012.

P Collinson, D Gaze, P Thokala, and S Goodacre. Randomised assessment of treatment using panel assay of cardiac markers - contemporary biomarker evaluation (ratpac cbe). *Health Technol Assess*, 17(15):1–122, Apr 2013.

C Cortes and V Vapnik. Support-vector networks. *Machine Learning*, 20:273–295, 1995.

DB Diercks, BE Mumma, W Frank-Peacock, JE Hollander, B Safdar, and SA Mahler. Incremental value of objective cardiac testing in addition to physician impression and serial contemporary troponin measurements in women. *Acad Emerg Med*, 20(3), Mar 2013.

R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:111–132, 1936.

L Grigull and WM Lechner. Supporting diagnostic decisions using hyrid and complementary data mining applicatins: a pilot study in the pediatric emergency department. *Pediatric Research*, Jun; 71(6):725–31, 2012.

T Hastie, R Tibshirani, and J Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.

M Jordan. Hierarchical mixture of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

J.E. Kelly. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.

S Klöppel, CM Stonnington, J Barnes, F Chen, C Chu, CD Good, I Mader, LA Mitchell, AC Patel, CC Roberts, NC Fox, CR Jr Jack, J Ashburner, and RS Frackowiak. Accuracy of dementia diagnosis: a direct comparison between radiologists and a computerized method. *Brain*, 131(Pt 11):2969–74, 2008.

G Lanckriet, N Cristianni, P Bartlett, and L El-Ghaoui. Learning the kernel matrix semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

C.-J. Lin and R.-C. Weng. A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, 68:267–276, 2007.

Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, and Kurt Hornik. *cluster: Cluster Analysis Basics and Extensions*, 2014. R package version 1.15.2 — For new features, see the 'Changelog' file (in the package source).

Shima Mehrabian, Margarita Raycheva, Martina Traykova, Tonya Stankova, Latchezar Penev, Olga Grigorova, and Latchezar Traykov. Neurosyphilis with dementia and bilateral hippocampal atrophy on brain magnetic resonance imaging. *BMC Neurology*, pages 12–96, 2012.

D Meyer. Support vector machines: The interface to libsvm in package e1071. Technical report, CRAN, 2012.

A Papier, D Tuttle, and T Mahar. Differential diagnosis of the swollen red eyelid. *Am Fam Physician*, 76(12):1815–1824, 2007.

J.C. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers*, 2000.

A Rakotomamonjy, F Bach, S Cann, and Y Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2000.

G Teasdale and B Jennett. Assessment of coma and impaired consciousness. a practical scale. *Lancet*, Jul 13;2(7872):81–4, 1974.

R Tibshirani and T Hastie. Margin trees for high-dimensional classification. *Journal of Machine Learning Research*, 6:637–652, 2007.

V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2nd edition, 1999.

D Warren, A Jarvis, L Leblanc, and J Gravel. Revisions to the canadian triage and acuity scale paediatric guidelines. *CJEM*, 10(3):224–232, 2008.

T.-F. Wu, C.-J. Lin, and R.C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.

J Yang and I Tsang. Hierarchical maximum margin learning for multi-class classification. *Uncertainity in Artifical Intelligence*, 6:753–760, 2011.

Appendix A: ICD-9 Diagnoses and Correct Classification Rates

Diagnosis	Code	Count	Training	Test
ABDOMINAL PAIN NYD	1	7813	0.905	0.923
ABSCESS BUTTOCK	2	100	0	0
ABSCESS SKIN	3	112	0	0
ACUTE APPENDICITIS	4	357	0	0
ACUTE CHOLECYSTITIS	5	446	0	0
ACUTE CORONARY SYNDROME	6	318	0.012	0
ACUTE GASTRITIS	7	351	0	0
ACUTE OTITIS MEDIA	8	173	0	0
ADJUSTMENT REACTION	9	475	0	0
ALCOHOL INTOXICATION	10	507	0.079	0.059
ALCOHOL WITHDRAWAL	11	142	0	0
ALCOHOLISM	12	267	0	0
ALLERGIC RASH	13	294	0	0
ALLERGY - UNSPECIFIED	14	285	0	0
ANEMIA NYD	15	393	0.076	0.026
ANGINA-STABLE	16	104	0	0
ANGINA UNSTABLE	17	288	0	0
ANXIETY	18	713	0	0
ANXIETY ATTACK	19	193	0	0
ARTHRALGIA KNEE	20	145	0	0
ARTHRITIS KNEE	21	209	0	0
ASTHMA	22	772	0.005	0
ASTHMATIC BRONCHITIS	23	138	0	0
ATRIAL FIBRILLATION (CONTROLLED)	24	442	0.014	0.034
ATRIAL FLUTTER	25	107	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
ATYPICAL PNEUMONIA	26	104	0	0
BACK PAIN	27	2329	0	0
BACTERIAL PNEUMONIA	28	1051	0	0
BELL'S PALSY	29	103	0	0
BENIGN POSITIONAL VERTIGO	30	489	0	0
BILIARY COLIC	31	373	0	0
BITE WOUND HAND	32	188	0	0
BLEEDING (PO)	33	169	0	0
BLEEDING IN EARLY PREGNANCY	34	399	0	0
BORDERLINE PERSONALITY DISORDER	35	174	0	0
BRONCHITIS	36	922	0	0
BURN HAND	37	130	0	0
CARDIAC ARREST	38	118	0	0
CAST	39	314	0	0
CATHETER	40	136	0	0
CELLULITIS - OTHER	41	410	0	0
CELLULITIS ARM (NOT HAND)	42	674	0	0
CELLULITIS FACE	43	324	0	0
CELLULITIS FINGER/THUMB	44	182	0	0
CELLULITIS FOOT	45	668	0	0
CELLULITIS HAND	46	384	0	0
CELLULITIS LEG	47	1679	0.095	0.099
CELLULITIS TOE	48	136	0	0
CERUMEN IMPACTION EAR	49	100	0	0
CERVICAL STRAIN	50	390	0	0
CHEST PAIN NYD	51	4649	0.37	0.395
CHOLELITHIASIS	52	253	0	0
CHRONIC BACK PAIN	53	107	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
CHRONIC PAIN (MISC)	54	172	0	0
COLITIS	55	207	0	0
COLLES' FRACTURE	56	203	0	0
COMPLETE ABORTION	57	116	0	0
CONCUSSION	58	685	0	0
CONFUSION NYD	59	167	0	0
CONGESTIVE HEART FAILURE	60	702	0.025	0.029
CONJUNCTIVITIS NYD	61	148	0	0
CONSTIPATION	62	606	0	0
CONTUSION ARM	63	108	0	0
CONTUSION BACK	64	203	0	0
CONTUSION ELBOW	65	119	0	0
CONTUSION FOOT	66	237	0	0
CONTUSION HAND	67	268	0	0
CONTUSION HIP	68	135	0	0
CONTUSION KNEE	69	240	0	0
CONTUSION LOWER LEG	70	137	0	0
CONTUSION RIBS	71	790	0	0
CONTUSION SHOULDER	72	193	0	0
CORNEAL ABRASION	73	477	0	0
COSTOCHONDRITIS	74	219	0	0
CROHN'S DISEASE	75	185	0	0
CYSTITIS	76	220	0	0
DECREASED VISION NYD	77	134	0	0
DEEP VEIN THROMBOSIS	78	370	0.014	0
DEHYDRATION	79	238	0	0
DELIRIUM	80	193	0	0
DEMENTIA	81	103	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
DENTAL ABSCESS	82	620	0	0
DENTAL CARIES	83	189	0	0
DEPRESSION	84	491	0	0
DIABETIC KETOACIDOSIS	85	122	0	0
DIARRHEA	86	663	0	0
DISLOCATION SHOULDER	87	220	0	0
DIVERTICULITIS	88	573	0	0
DIZZY NYD (NEURO)	89	459	0	0
DRUG ALLERGY	90	129	0	0
DYSFUNCTIONAL UTERINE BLEEDING	91	126	0	0
EAR PAIN	92	139	0	0
EFFUSION KNEE	93	129	0	0
EPIDIDYMITIS	94	135	0	0
EPISTAXIS	95	444	0	0
EXACERBATION COPD	96	1586	0	0
FALLS (GER)	97	275	0	0
FATIGUE/MALAISE UNSPECIFIED	98	210	0	0
FEBRILE NEUTROPENIA	99	123	0	0
FEVER NYD (MISC)	100	525	0.002	0
FLU LIKE ILLNESS	101	178	0	0
FOREIGN BODY CORNEA	102	161	0	0
FOREIGN BODY EYE	103	229	0	0
FOREIGN BODY SENSATION EYE	104	136	0	0
FRACTURE 5TH METACARPAL SHAFT	105	112	0	0
FRACTURE 5TH METATARSAL	106	186	0	0
FRACTURE CLAVICLE	107	107	0	0
FRACTURE DISTAL RADIUS	108	322	0	0
FRACTURE FEMUR	109	102	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
FRACTURE FIBULA	110	178	0	0
FRACTURE HIP	111	539	0.009	0
FRACTURE HUMERUS	112	226	0	0
FRACTURE LATERAL MALLEOLUS	113	165	0	0
FRACTURE METATARSAL	114	152	0	0
FRACTURE NOSE	115	106	0	0
FRACTURE PHALANX SHAFT	116	175	0	0
FRACTURE RADIAL HEAD	117	157	0	0
FRACTURE RIB	118	191	0	0
FRACTURE RIBS	119	116	0	0
FRACTURE SHAFT METACARPAL	120	113	0	0
FRACTURE TIBIA	121	106	0	0
FRACTURE TOE	122	213	0	0
FRACTURE WRIST	123	175	0	0
GASTROENTERITIS	124	1168	0	0
GASTROINTESTINAL BLEED	125	503	0	0
GOUT	126	162	0	0
GOUT, ACUTE	127	144	0	0
HEADACHE, MIGRAINE	128	1091	0	0
HEADACHE, OTHER	129	1267	0	0.004
HEMATURIA	130	529	0.005	0
HEMOPTYSIS	131	110	0	0
HEMORRHOIDS	132	204	0	0
HYPERGLYCEMIA	133	249	0	0
HYPERTENSION	134	550	0	0
HYPOGLYCEMIC REACTION	135	110	0	0
HYPOKALEMIA	136	192	0	0
HYPONATREMIA	137	131	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
INCOMPLETE ABORTION	138	178	0	0
INFECTED SEBACEOUS CYST	139	118	0	0
INFLUENZA	140	104	0	0
INGUINAL HERNIA	141	131	0	0
INSECT BITES	142	133	0	0
INTERNAL DERANGEMENT KNEE	143	116	0	0
LACERATION EYEBROW	144	185	0	0
LACERATION FINGER (NO TENDON)	145	1344	0	0
LACERATION FINGER WITH TENDON	146	106	0	0
LACERATION FOREARM	147	171	0	0
LACERATION FOREHEAD	148	222	0	0
LACERATION HAND	149	536	0	0
LACERATION LIP	150	159	0	0
LACERATION PALM	151	105	0	0
LACERATION SCALP	152	400	0	0
LACERATION SHIN	153	150	0	0
LACERATION THUMB	154	364	0	0
LIGAMENTOUS INJURY KNEE	155	120	0	0
LOW BACK STRAIN	156	430	0	0
LUNG CA	157	139	0	0
MANIC DEPRESSIVE ILLNESS	158	112	0	0
MECHANICAL LOW BACK PAIN	159	523	0	0
MEDICAL DEVICE SUPPORT	160	247	0	0
MEDICATION CONCERNS/PROBLEMS	161	710	0	0
MENORRHAGIA	162	110	0	0
METASTATIC CANCER	163	376	0	0
MILD CLOSED HEAD INJURY	164	553	0	0
MULTIPLE CONTUSIONS	165	308	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
MULTIPLE TRAUMA	166	120	0	0
MUSCLE SPASM BACK	167	270	0	0
MUSCULAR NECK PAIN	168	318	0	0
MUSCULOSKELETAL PAIN	169	265	0	0
NORMAL EXAM	170	397	0	0
NSTEMI	171	448	0	0
OLECRANON BURSITIS	172	136	0	0
OSTEOARTHRITIS	173	121	0	0
OTHER CELLULITIS/ABSCESS	174	228	0	0
OTHER DENTAL/ORAL	175	273	0	0
OTHER ENT	176	202	0	0
OTHER ESOPHAGEAL	177	150	0	0
OTHER FRACTURE ANKLE/FOOT	178	347	0	0
OTHER FRACTURE WRIST/HAND	179	111	0	0
OTHER INTESTINAL	180	178	0	0
OTHER MSK	181	100	0	0
OTHER NEURO	182	555	0.018	0.009
OTHER OPHTHALMOLOGIC	183	166	0	0
OTHER PREGNANCY RELATED	184	384	0	0
OTHER PSYCHIATRIC	185	224	0	0
OTHER RESPIRATORY	186	475	0.011	0.011
OTHER SKIN LESION	187	108	0	0
OTHER SPRAIN/STRAIN TRUNK	188	112	0	0
OTHER SYNCOPE	189	155	0	0
OTHER UROLOGIC	190	411	0	0
OTHER VASCULAR	191	143	0	0
OTITIS EXTERNA	192	195	0	0
OVARIAN CYST	193	227	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
PAIN - ABD NYD	194	633	0	0
PAIN - BACK NYD	195	547	0	0
PAIN - CHEST NYD	196	2273	0.007	0
PAIN - HEAD NYD	197	159	0	0
PAIN - HIP NYD	198	217	0	0
PAIN - NECK NYD	199	229	0	0
PAIN ANKLE	200	192	0	0
PAIN ARM	201	250	0	0
PAIN CALF	202	135	0	0
PAIN EYE NYD	203	131	0	0
PAIN FINGER	204	113	0	0
PAIN FOOT	205	451	0	0
PAIN HAND	206	161	0	0
PAIN LEG	207	809	0.002	0
PAIN NYD (MISC)	208	118	0	0
PAIN SHOULDER	209	460	0	0
PAIN WRIST	210	210	0	0
PALPITATIONS	211	828	0.006	0
PANCREATITIS	212	323	0	0
PARESTHESIA, NYD	213	276	0	0
PELVIC PAIN NYD	214	379	0	0
PERIRECTAL ABSCESS	215	138	0	0
PHARYNGITIS	216	650	0	0
PILONIDAL ABSCESS	217	157	0	0
PLEURAL EFFUSION	218	215	0.012	0
PNEUMONIA NOS	219	890	0.185	0.112
PNEUMOTHORAX	220	194	0	0
POST OP CHECK (PO)	221	671	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
PREGNANCY PV SPOTTING	222	140	0	0
PRESYNCOPE	223	306	0	0
PSYCHOSIS NYD	224	244	0	0
PULMONARY EMBOLUS	225	230	0	0
PYELONEPHRITIS	226	400	0	0
RASH NYD	227	412	0	0
RECTAL BLEEDING	228	353	0	0
REFLUX ESOPHAGITIS	229	280	0	0
RENAL COLIC	230	1772	0.106	0.054
RENAL FAILURE ACUTE	231	195	0	0
ROTATOR CUFF SYNDROME	232	140	0	0
SCHIZOPHRENIA	233	135	0	0
SCIATICA	234	437	0	0
SEIZURE	235	587	0	0
SEPSIS	236	137	0.009	0
SHINGLES	237	210	0	0
SINUSITIS - ACUTE	238	337	0	0
SITUATIONAL ANXIETY	239	244	0	0
SITUATIONAL DEPRESSION	240	107	0	0
SMALL BOWEL OBSTRUCTION	241	553	0	0
SOB NYD	242	679	0	0
SOCIAL PROBLEM	243	111	0	0
SPRAIN ANKLE	244	1432	0	0
SPRAIN FOOT	245	361	0	0
SPRAIN MEDIAL COLLATERAL LIG. KNEE	246	120	0	0
SPRAIN SHOULDER LIGAMENTS/CAPSULE	247	217	0	0
SPRAIN THUMB	248	118	0	0
SPRAIN WRIST	249	392	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
SPRAINED KNEE	250	549	0	0
STRAIN CALF MUSCLE	251	104	0	0
STRAIN INTERCOSTAL MUSCLE	252	175	0	0
STREPTOCOCCAL PHARYNGITIS	253	122	0	0
STROKE, ISCHEMIC	254	407	0	0
SUBCONJUNCTIVAL HAEMORRHAGE	255	115	0	0
SUBSTANCE ABUSE	256	204	0	0
SUICIDAL IDEATION	257	272	0	0
SUPERFICIAL THROMBOPHLEBITIS	258	130	0	0
SUPRAVENTRICULAR TACHYCARDIA	259	121	0	0
SUTURE/STAPLE REMOVAL(PO)	260	112	0	0
SWELLING KNEE	261	186	0	0
SWELLING LEG	262	354	0	0
SYNCOPE NYD (CARD)	263	159	0	0
SYNCOPE NYD (MISC)	264	409	0	0
TENDON INJURY	265	103	0	0
THROAT PAIN	266	130	0	0
TIA	267	415	0	0
TONSILLITIS	268	131	0	0
TOOTHACHE	269	194	0	0
TOXICOLOGY - ALCOHOL	270	123	0	0
TRANSIENT SYMPTOMS NYD	271	390	0	0
UNCONTROLLED ATRIAL FIBRILLATION	272	444	0	0
UPPER RESPIRATORY INFECTION/ COLD	273	842	0	0
URETERAL CALCULUS	274	229	0	0
URINARY RETENTION	275	588	0	0
URINARY TRACT INFECTION	276	2501	0	0
UROSEPSIS	277	220	0	0

continued on next page . . .

. . . continued from last page

Diagnosis	Code	Count	Training	Test
URTICARIA	278	123	0	0
VAGINAL BLEEDING	279	383	0	0
VASOVAGAL SYNCOPE	280	422	0	0
VERTIGO (NEURO)	281	253	0	0
VIRAL ILLNESS	282	823	0	0
VOMITING	283	772	0	0
WEAKNESS NYD	284	870	0	0
WHIPLASH INJURY	285	124	0	0
WOUND INFECTION (PO)	286	302	0	0
Z-POST OP OTHER	287	255	0	0

Appendix B: R-code

Appendix: R code (marginCalculator)

```
> #Function that calculates the "distance" of two classes using an SVM
>
> #We define distance here as the sum of the Euclidian distance of the margin
> #2/||B||, where B is the vector of coefficients associated with the SVM in question,
> #and the proportion of misclassification.
>
> #Distance = size of margin + misclassification proportion
>
> #For the function need to specify an SVM object (from e1071 implementation),
> #the first and second class for comparison, and the data source.
>
> marginCalculator <- function(svm, class1, class2, data){
+
+   #margin size (from Elements)
+   margin <- 2 / sqrt(sum(svm$coefs^2))
+
+   #Rate of misclassification for the i,jth comparison
+   tab <- xtabs(~ predict(svm) + subset(data[,1],
+                                      xor(data[,1] == class1,
+                                          data[,1] == class2)))
+
+   #Correct Classification rate
+   error <- sum(diag(tab))/sum(tab)
+
+   if(error == 1){
+
+     return(margin + error)
+   }
+
+   else{
+
+     return(error)
+   }
+
+ }
```

Appendix: R code (hierarchyTrain)

```
> #Function that takes a given data set, and makes n-choose-2 SVM comparisons, and
> #places them in a list object for reference later to build dissidence matrix.
>
> #Three parameters:
>
> #classData: the data we are building the structure on
>
> #startClass: the first numbered class to start with
> #added so that one can divide the computationally
> #intensive task up between workers.
>
> #numClasses: the total number of classes in the dataset
> #so that we do not miss pairwise comparisons.
>
> #Requires the following functions:
>
> #marginCalculator: This calculates the value of interest,
> #the distance between the two class labels
>
> hierarchyTrain <- function(classData, startClass, endClass, numClasses){
+
+   #Empty matrix to place margins into
+   marginMat <- matrix(,nrow = choose(numClasses, 2))
+
+   #Initialize count variable for matrix reference
+   k = 1
+
+   #Initialize count variable to stop process when
+   #the prescribed number of classes is reached.
+   l = startClass
+
+   while(l < endClass){
+
+         for(i in l:(endClass - 1)){
+
+       for(j in (i + 1):(numClasses)){
+
+               #Train an SVM on the features, subsetting the data for training purposes
+
+         #Want to weight the observations to eliminate asymmetric effect of
+         #different class sizes
+
+         tempData <- subset(classData, xor(classData[,1] == i,
+                                           classData[,1] == j))
+
+         classOneNum <- nrow(subset(tempData, tempData[,1] == i))
```

Appendix: R code (hierarchyTrain)

```
+        classTwoNum <- nrow(subset(tempData, tempData[,1] == j))
+
+        maxNum <- max(classOneNum, classTwoNum)
+
+        #Sets the class with the most observations
+        #to a weight of one, and upweights the
+        #smaller class to equivalence.
+        wts <- maxNum /c(classOneNum, classTwoNum)
+
+        #Add in variables that are always present.
+        for(m in 2:4){
+
+          newFormula <- paste0(newFormula, names(tempData)[m], " + ")
+
+        }
+
+        for(m in 1:length(which(colSums(tempData[,5:195]) != 0))){
+
+          if(m == length(which(colSums(tempData[,5:195]) != 0))){
+
+            newFormula <- paste0(newFormula,
+                                 attr(which(colSums(tempData[,5:195]) != 0)[m],
+                                      "names"))
+
+          }
+
+          else{
+
+            newFormula <- paste0(newFormula,
+                                 attr(which(colSums(tempData[,5:195]) != 0)[m],
+                                      "names")," + ")
+
+          }
+        }
+
+
+        svmNow <- svm(subset(classData[,1],
+                             xor(classData[,1] == i,
+                             classData[,1] == j)) ~.,
+                      scale = TRUE,
+                      kernel = "linear",
+                      class.weights = c(i = wts[1], j = wts[2]),
+                      type = "C-classification",
+                      data = subset(classData,
+                                    xor(classData[,1] == i,
+                                    classData[,1] == j)))
```

Appendix: R code (hierarchyTrain)

```
+
+               #Place trained SVM in the list for access later.
+               marginMat[k] <- marginCalculator(svmNow, i,
+                                              j, classData)
+
+               #Indictator to see how far along the program is
+               print(k)
+
+         #Increment placement reference
+         k = k + 1
+
+         #Remove current svm model to free up memory
+         rm(svmNow)
+
+         #Run the garbage collector to ensure freed up
+         #RAM
+         gc()
+     }
+
+     l = l + 1
+
+    }
+
+   }
+
+         return(marginMat)
+
+ }
```

Appendix: R code (dissidenceMatrix)

```
> #Function to calculate distance matrix for hClust
> #Requires the matrix of the distances that come from
> #the hierarchyTrain function. The return value
> #from this is passed into the dist() function for
> #preparation to use in the hclust() function.
>
> distanceMatrix <- function(marginMat, numClasses){
+
+   distance <- matrix(,nrow = numClasses, ncol = numClasses)
+
+   diag(distance) <- 0
+
+   k <- 1
+
+   for(i in 1:(numClasses-1)){
+
+     for(j in (i+1):numClasses){
+
+       if(i != j){
+
+         distance[j, i] <- marginMat[k]
+
+         distance[i, j] <- marginMat[k]
+
+         k <- k + 1
+       }
+
+       if(i == j){
+
+         distance[j, i] <- 0
+
+       }
+
+     }
+
+   }
+
+   return(distance)
+ }
```

Appendix: R code (tracer)

```
> #Function that gives the node membership of a particular classification
> #in the hierarchy
>
> tracer <- function(hclustObject, class){
+
+   trMatrix <- matrix(, nrow=length(hclustObject$merge[,1]))
+
+   k = 1
+
+   #Need an indicator variable to tell us when we've reached the terminal node
+   stopRule=1
+
+   while(length(stopRule) != 0){
+
+       #Set class to the index in the merge list. This is the same as
+       #node membership
+       class <- which(hclustObject$merge == class, arr.ind = TRUE)[,1]
+
+       #Place in a matrix that we can use as a list
+       trMatrix[k] <- class
+
+       #Update stop value
+       stopRule <- which(hclustObject$merge==class, arr.ind=TRUE)[,1]
+
+       k = k+1
+   }
+   return(rev(na.omit(trMatrix)))
+ }
```

Appendix: R code (structTracer)

```
> #Modified function that uses modified merge list to assess structure effect.
>
> structTracer <- function(merge, class){
+
+   trMatrix <- matrix(,nrow = length(merge[,1]))
+   k <- 1
+
+   #Need an indicator variable to tell us when we've reached the terminal node
+   stopRule <- 1
+
+   #Make sure we're not at the end of the list, so it does not throw an error
+   while(length(stopRule) != 0){
+
+     #Set class to the index in the merge list. This is the same as node membership
+     class <- which(merge == class, arr.ind = TRUE)[,1]
+
+     #Place in a matrix that we can use as a list
+     trMatrix[k] <- class
+
+     #Update stop value
+     stopRule <- which(merge == class,arr.ind=TRUE)[,1]
+     k <- k+1
+   }
+
+   #Get rid of parts of the list that do not matter
+   return(rev(na.omit(trMatrix)))
+ }
```

Appendix: R code (randomization)

```
> #EDIS data randomization
>
> edis <- read.csv("filteredData.csv")
> #Set diagnostic labels as numbers to work
> #in the functions
>
> edis$Diagnosis <- as.numeric(edis$Diagnosis)
> #Create list of diagnosises and their numeric code
> write.table(edis$Diagnosis, file = "numLabel.txt", sep = " ")
> #Split data into training set and test set
>
> #Set reproducible random seed
> set.seed(256)
> #Empty dataframes with same column names
> #$Diagnosis$ is a value known to not be true
> #here
>
> train.edis <- subset(edis, Diagnosis == 304)
> test.edis <- subset(edis, Diagnosis == 304)
> #Iterate through each diagnosis label
>
> for(i in 1:287){
+
+   tempData <- subset(edis, Diagnosis == i)
+
+   #Sample indexes and use these
+   #to subset and separate data so there are an
+   #even distribution of the diagnostic labels
+
+   indexes <- sample(1:nrow(tempData), size = 0.2*nrow(tempData))
+
+   test.edis <- rbind(test.edis, tempData[indexes,])
+
+   train.edis <- rbind(train.edis, tempData[-indexes,])
+ }
```

Appendix: R code (modelFit)

```
> #Function that automatically fits SVM models based on the merge list,
> #and places them in a structure to refer to later.
>
> #This function is specific to the emergency department data.
> #Requires the dataframe produced by the prepareMatrix() function
>
> #Other arguments required:
>
> #mergeList: the merge list from the hclust object fit during the
> #initial structural learning.
>
> #numCol: the number of columns of the original data. This is to
> #point the model to classify on the right outcome variable.
>
> #start: Tells the function which node to focus on first.
> #finish: Tells the function the last node to focus on.
> #These two values allow you to split the computations up amonst
> #different workers to finish faster.
>
> #Note that it took almost a month to train all of the EDIS data,
> #so only run a large fitting if one is comfortable waiting.
>
> modelFit <- function(dataframe, mergeList, numCol, start, finish){
+
+   k = start
+
+   for(i in start:finish){
+
+     #Subset the data that is appropriate for the pair-wise comparison
+     tempData <- subset(dataframe, dataframe[,(numCol+i)] == TRUE)
+
+     #if clause to account for having a superficial split
+
+     if(mergeList[i,1]<0){
+
+       #Define weights to counteract unbalanced data
+       classOneNum <- sum(tempData[, numCol + (numClasses - 1) - mergeList[i,1]] == TRUE)
+       classTwoNum <- sum(tempData[, numCol + (numClasses - 1) - mergeList[i,1]] == FALSE)
+
+       maxNum <- max(classOneNum, classTwoNum)
+
+       wts <- maxNum/c(classOneNum, classTwoNum)
+
+       #Make formula
+       newFormula <- paste0(names(tempData)[numCol + 286 - mergeList[i, 1]], " ~ ")
+
```

Appendix: R code (modelFit)

```
+       #Add in variables that are always present.
+       for(j in 2:4){
+
+         newFormula <- paste0(newFormula, names(tempData)[j], " + ")
+
+       }
+
+       for(l in 1:length(which(colSums(tempData[,5:195]) != 0))){
+
+         if(l == length(which(colSums(tempData[,5:195]) != 0))){
+
+           newFormula <- paste0(newFormula,
+                                attr(which(colSums(tempData[,5:195]) != 0)[l],
+                                     "names"))
+           newFormula <- as.formula(newFormula)
+         }
+
+         else{
+
+           newFormula <- paste0(newFormula,
+                                attr(which(colSums(tempData[,5:195]) != 0)[l],
+                                     "names")," + ")
+           newFormula <- as.formula(newFormula)
+         }
+       }
+
+       #Utilize this formula to train model
+       trainedModel <- svm(newFormula,
+                           data = tempData, kernel = "linear",
+                           type = "C-classification", probability = TRUE,
+                           class.weights = wts)
+
+       #Save the model to the hard memory for reference later
+       save(trainedModel, file = paste0("Model", k,".rda"))
+
+       #Print the iteration so we know the progress
+       print(paste0("Model", k, ".rda"))
+       k = k + 1
+
+       #Free up RAM by removing the model from the workspace
+       rm(trainedModel)
+
+     }
+
+     #Fit a model at each node
+     else{
```

Appendix: R code (modelFit)

```
+
+       #Define weights to counteract unbalanced data
+       classOneNum <- sum(tempData[, numCol + (numClasses - 1) - mergeList[i,1]] == TRUE)
+       classTwoNum <- sum(tempData[, numCol + (numClasses - 1) - mergeList[i,1]] == FALSE)
+
+       maxNum <- max(classOneNum, classTwoNum)
+
+       wts <- maxNum/c(classOneNum, classTwoNum)
+
+       #Make up the formula for the model
+       newFormula <- paste0(names(tempData)[195 + mergeList[i, 1]], " ~ ")
+
+       #Add in variables that are always present.
+       for(j in 2:4){
+
+         newFormula <- paste0(newFormula, names(tempData)[j], " + ")
+
+       }
+
+       for(l in 1:length(which(colSums(tempData[,5:195]) != 0))){
+
+         if(l == length(which(colSums(tempData[,5:195]) != 0))){
+
+           newFormula <- paste0(newFormula,
+                                attr(which(colSums(tempData[,5:195]) != 0)[l],
+                                     "names"))
+           newFormula <- as.formula(newFormula)
+         }
+
+         else{
+
+           newFormula <- paste0(newFormula,
+                                attr(which(colSums(tempData[,5:195]) != 0)[l],
+                                     "names")," + ")
+           newFormula <- as.formula(newFormula)
+         }
+       }
+       #Utilize this formula to train model
+       trainedModel <- svm(newFormula,
+                           data = tempData, kernel = "linear",
+                           type = "C-classification", probability = TRUE,
+                           class,weights = wts)
+
+       #Save the model to the hard memory for reference later
+       save(trainedModel, file = paste0("Model", k,".rda"))
+
```

Appendix: R code (modelFit)

```
+       #Free up RAM by removing the model from the workspace
+       rm(trainedModel)
+
+       #Print the iteration so we know the progress
+       print(paste0("Model", k, ".rda"))
+       k = k + 1
+     }
+
+   }
+
+ }
```

Appendix: R code (probMatrixMake)

```
> #Collecting the predictions from the models
> #To not use up all the RAM, need to bring the
> #model in, take the predictions (probabilities)
> #and then remove the model from the workspace.
>
> #Need to tell the function how many node memberships
> #there are, number of observations to predict,
> #which node membership you want to start on, and
> #which node to stop (so you can parcel out the
> #predictions in the high-computational case.
>
> #Also requires df, which is the dataframe of observations
> #to make the predictions on.
>
> probsMatrixMake <- function(numCol, numRow, startCol, endCol, mergeList, df){
+
+   #Create matrix to contain the predicted probs
+   probMat <- matrix(ncol = numCol, nrow = numRow)
+
+   #Cycle through all of the membership models
+   for(i in startCol:endCol){
+
+     #Since class labels are negative, test for less than zero here.
+
+     for(j in 1:2){
+
+       #Read in model structure from outside source, all are named
+       #"trainedModel", so can refer to and overwrite as needed
+       load(paste0("Model", i, "c", j,".rda"))
+
+       print(paste0("Model", i, "c", j,".rda"))
+
+       preds <- attr(predict(trainedModel, df, probability = TRUE), "probabilities")
+
+
+       if(mergeList[i,j] < 0){
+
+         #Places predictions of node membership in same column as is in the
+         #dataframe
+         if(dimnames(preds)[[2]][1] == TRUE){
+
+           probMat[,(length(mergeList[,1]) - 1) - mergeList[i, j]] <- preds[,1]
+         }
+
+         else{
+
```

Appendix: R code (probMatrixMake)

```
+           probMat[, (length(mergeList[,1]) - 1) - mergeList[i, j]] <- preds[,2]
+
+         }
+       }
+
+       #If membership of node rather than class label, place in same relative position
+       #as the dataframe.
+       else{
+
+         if(dimnames(preds)[[2]][1] == TRUE){
+
+           probMat[, mergeList[i, j]] <- preds[,1]
+
+         }
+
+         else{
+
+           probMat[, mergeList[i, j]] <- preds[,2]
+
+         }
+       }
+     }
+
+     #Sometimes takes a great deal of time, so a counter
+     #to mark progress
+
+     print(i)
+   }
+
+   return(probMat)
+ }
```

Appendix: R code (makePredictions)

```
> #Code to calculate how probable each class is for each observation
>
> #Requires the following arguments:
>
> #OriginalData: the dataframe produced by the PrepareMatrix function
>
> #ProbabilityMatrix: the matrix of the probabilities of node membership
> #                   for each observation produced by the probMatrixMake
> #                   function.
>
> #mergeList: the list of merges from the hierarchical structure of the data
> #           which is given by the hierarchyTrain function
>
> #Requires the following functions in your workspace:
>
> #makeConfusionMatrix, structTracer
>
> makePredictions <- function(originalData, probabilityMatrix,
+                             numCol, mergeList, endM){
+
+   #Saving number of rows of the data for use later
+   numRow <- nrow(originalData)
+
+   #Empty matrix to store probabilities of a given
+   #class label, with each row an observation
+   predictionMatrix <- matrix(ncol = max(originalData[,1]), nrow = numRow)
+
+   for(i in 1:max(originalData[,1])){
+
+     #Path gives the references for the predicted
+     #probability matrix. Take off the first entry
+     #since it refers to the probability of being
+     #in the most superior cluster, which is unity.
+
+     path <- c(structTracer(mergeList, -i),
+               length(mergeList[,1]) + i -1)[-1]
+
+     #Now simply place the product of these probabilities
+     #into the prediction matrix for each observation
+
+     for(j in 1:numRow){
+
+       predictionMatrix[j,i] <- prod(probabilityMatrix[j,path])
+
+     }
+
```

Appendix: R code (makePredictions)

```
+   }
+
+   #Now that the probability of each class label is given
+   #for each observation, order them to give the rankings
+   #of each class label
+
+   orderedPredictions <- matrix(ncol = numCol, nrow = numRow)
+
+   for(k in 1:numRow){
+
+     orderedPredictions[k,] <- order(predictionMatrix[k,], decreasing = TRUE)
+   }
+
+   #Create confusion matrix for the rest of the program
+   conf.matrix <- makeConfusionMatrix(originalData[,1],
+                                      orderedPredictions[,1])
+
+   #Gives the total number correct
+   totalFirstCorrect <- sum(diag(conf.matrix))
+
+   #Initialize variable to collect number correct
+   totalTenCorrect <- 0
+
+   #Loop collects the first most probable class label, the next, and so on.
+   responseCurveData <- matrix(nrow = 287)
+
+   for(m in 1:endM){
+
+     totalTenCorrect <- totalTenCorrect +
+                        sum(diag(makeConfusionMatrix(originalData[,1],
+                                                     orderedPredictions[,m])))
+
+     responseCurveData[m] <- totalTenCorrect/numRow
+   }
+
+   #Matrix for presentation of the results. Store in return list for reference.
+   returnMatrix <- matrix(c(totalFirstCorrect, totalFirstCorrect/numRow,
+                            totalTenCorrect, totalTenCorrect/numRow),
+                          nrow = 2)
+
+   colnames(returnMatrix) <- c("Correct first rank",
+                               "Correct in ten ranks")
+   rownames(returnMatrix) <- c("Absolute number",
+                               "Correct rate")
+
+   #Place everything into a list for the return statement
```

Appendix: R code (makePredictions)

```
+   returnList <- list()
+   returnList[[1]] <- returnMatrix
+   returnList[[2]] <- conf.matrix
+   returnList[[3]] <- responseCurveData
+
+   return(returnList)
+ }
```

Appendix: R code (randomMerge)

```
> #Script to randomly modify the tree structure, retrain the models,
> #and give the predictions
>
> #Note: takes a long time to run, so do not execute unless one is
> #sure.
>
> #Place predictions into a matrix
> newErrors <- function(){
+
+   matrix(,nrow=30)
+
+   for(h in 1:30){
+
+     #Randomize the predictions in the merge list to randomize terminal
+     #experts
+     newMer <- randomMerge(zip.Cluster)
+
+     #Append outcome variables to the dataframe
+     newTrainingData <- prepareMatrix(zip.train, 10, newMer)
+
+   #For the sake of reference, give the names of the variables in the
+   #dataframe
+   names(newTrainingData)[258:276] <- names(test.df)[258:276]
+
+   #Refit the SVMs with said models
+   newModels <- modelFit(newTrainingData, newMer, 257)
+
+   #Prepare the test data
+   newTestSet <- prepareMatrix(zip.test, 10, newMer)
+
+   #Rename the columns for the sake of reference
+   names(newTestSet)[258:276] <- names(test.df)[258:276]
+
+   #Calculate the probability of each class for each observation
+   newProbs <- predictionProbs(newTestSet, newMer, newModels, 10)
+
+   #Empty matrix to contain predicted class labels
+   predictionsMat <- matrix(nrow=2007,ncol=10)
+
+   #Placing predictions in said matrix
+   for(j in 1:2007){
+
+     predictionsMat[j,] <- order(-newProbs[j,])-1
+
+   }
+
```

Appendix: R code (randomMerge)

```
+   #Place classification rate in the errors matrix
+   newErrors[h] <- sum(diag(xtabs(~ newTestSet[,1] + predictionsMat[,1])))/2007
+
+   }
+ }
```

Appendix: R code (newErrors)

```
> #Script to randomly modify the tree structure, retrain the models,
> #and give the predictions
>
> #Note: takes a long time to run, so do not execute unless one is
> #sure.
>
>
> newErrors <- function(){
+
+   #Place predictions into a matrix
+   matrix(,nrow=30)
+
+   for(h in 1:30){
+
+     #Randomize the predictions in the merge list to randomize terminal
+     #experts
+     newMer <- randomMerge(zip.Cluster)
+
+     #Append outcome variables to the dataframe
+     newTrainingData <- prepareMatrix(zip.train, 10, newMer)
+
+     #For the sake of reference, give the names of the variables in the
+     #dataframe
+     names(newTrainingData)[258:276] <- names(test.df)[258:276]
+
+     #Refit the SVMs with said models
+     newModels <- modelFit(newTrainingData, newMer, 257)
+
+     #Prepare the test data
+     newTestSet <- prepareMatrix(zip.test, 10, newMer)
+
+     #Rename the columns for the sake of reference
+     names(newTestSet)[258:276] <- names(test.df)[258:276]
+
+     #Calculate the probability of each class for each observation
+     newProbs <- predictionProbs(newTestSet, newMer, newModels, 10)
+
+     #Empty matrix to contain predicted class labels
+     predictionsMat <- matrix(nrow=2007,ncol=10)
+
+     #Placing predictions in said matrix
+     for(j in 1:2007){
+
+       predictionsMat[j,] <- order(-newProbs[j,])-1
+
+     }
```

Appendix: R code (newErrors)

```
+
+     #Place classification rate in the errors matrix
+     newErrors[h] <- sum(diag(xtabs(~ newTestSet[,1] +
+                                      predictionsMat[,1])))/2007
+
+   }
+ }
```

Appendix: R code (makeConfusionMatrix)

```
> # Create a confusion matrix from the given outcomes, whose rows correspond
> # to the actual and the columns to the predicted class labels.
> makeConfusionMatrix <- function(dataLabels, predictedLabels) {
+
+   #Declare max number of classes to make the matrix
+   numClasses <- max(dataLabels, predictedLabels)
+
+   #Order the labels for the rows and columns
+   predictedLabels <- predictedLabels[order(dataLabels)]
+   dataLabels  <- dataLabels[order(dataLabels)]
+
+   sapply(split(predictedLabels, dataLabels), tabulate, nbins = numClasses)
+ }
```