MACHINE LEARNING METHODS WITH TIME SERIES DEPENDENCE

Blakeley Barton McShane

A DISSERTATION

in

Statistics

For the Graduate Group in Managerial Science and Applied Economics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2010

Supervisor of Dissertation

_____

Abraham J. Wyner, Associate Professor, Statistics

Graduate Group Chairperson

_____

Eric T. Bradlow, Professor of Marketing, Statistics, and Education

Dissertation Committee:
Abraham J. Wyner, Associate Professor, Statistics
Eric T. Bradlow, Professor of Marketing, Statistics, and Education
Shane T. Jensen, Assistant Professor, Statistics
Abba M. Krieger, Professor of Statistics and Operations Research, Operations and Information Management, and Marketing

Machine Learning Methods with Time Series Dependence

Sancto Patri Nostro Papae Benedicto XVI

Precamur ne lupos fugiatis
in angustiis praesentibus et in temporibus omnibus.

Acknowledgements

Many thanks go to my advisor Abraham Wyner for his collaboration on this and several other projects. I have learned far too much from him to cover in this small space and have had tremendous fun doing so. I look forward to our continued work together.

I also owe a great deal to the three members of my committee. Eric Bradlow has been an incredible mentor both academically, professionally, and personally. Shane Jensen has been a terrific friend and coauthor on this and other work. Finally, Abba Krieger imparted a great deal of wisdom to me over the last few years and provided valuable help with this manuscript.

I am also indebted to Allan Pack and the entire Center for Sleep Research and Respiratory Neurobiology for data, funding, and intellectual stimulation over the past three years.

I have been extremely fortunate to have been a student in the Statistics Department at Wharton. I have learned a great deal from the entire faculty, but in particular from those who have taught me, coauthored papers with me, or have oth-

erwise assisted me including Andreas Buja, Dylan Small, Dean Foster, Mike Steele, and Mark Low. I have also benefited greatly from interaction with my fellow graduate students, in particular Alex Braunstein, Sivan Aldor-Noiman, Mike Baiocchi, Oliver Entine, and James Piette.

I am indebted to numerous faculty outside of our department, particularly from my undergraduate days. William Whitney and Martin Asher were fabulous professors, advisors, and mentors to me in the Joseph Wharton Scholars program. Jamshed Ghandhi, Pete Fader, Jonathan Block, and Jack Siler exposed me to among the most rigorous and interesting coursework I have seen and thus have had a profound impact on me.

I would like to thank my friends. I owe untold amounts to Mia Adelberg and Rya Conrad-Bradshaw. Amy Vegari, Andrew Torre, J. J. McKeever, Andrew Rogers, Trey Best, and C. J. Walsh from the Episcopal Academy are among my oldest and dearest friends and I am greatly indebted. I couldn't possibly leave out the gentlemen of Pennsylvania Six-5000 and in particular Alex "Woody" Hayden, Adam "Tesh" Yanoff, Josh "Kramer" Palley, and Dave Katowitz. My Penn friends–Oliver Watson and Alex Cowan as well as John De Palma, Jeff Smith, and Alex Hardin–deserve special mention as do the friends who stem from my time in England–Billy Newton, Will Chapman, Chris Goff, and Bastian Meyenburg as well as Anton Webb and Sushma Shankar. Ian Doherty, John Wynn, Mike Wynn, Dean Costalas, and John Middleton are among the best friends I could ask for. I also owe the staff at

Tria, particularly Lauren Daprile and Stephanie Lehmann LeMay. Finally, many thanks to my friends from Earle Mack (or should I say Drexel?) School of Law who have adopted me–Sara Goldstein, Devon Morrissey, Tim Howett, and most especially Eva Miller.

Last, but certainly not least, I would like to thank my family. Mom, Dad, Ryan, Kyle, Kerri, Bridget, Haley, and Cullen, I love you greatly and am lucky to have you.

ABSTRACT

MACHINE LEARNING METHODS WITH TIME SERIES DEPENDENCE

Blakeley B. McShane

Abraham J. Wyner (Advisor)

We introduce the PrAGMaTiSt: Prediction and Analysis for Generalized Markov Time Series of States, a methodology which enhances classification algorithms so that they can accommodate sequential data. The PrAGMaTiSt can model a wide variety of time series structures including arbitrary order Markov chains, generalized and transition dependent generalized Markov chains, and variable length Markov chains. We subject our method as well as competitor methods to a rigorous set of simulations in order to understand its properties. We find, for very low or high levels of noise in $Y_t|X_t$, complexity of $Y_t|X_t$, or complexity of the time series structure, simple methods that either ignore the time series structure or model it as first order Markov can perform as well or better than more complicated models even when the latter are true; however, in moderate settings, the more complicated models tend to dominate. Furthermore, even with little training data, the more complicated models perform about as well as the simple ones when the latter are true. We also apply

the PrAGMaTiSt to the important problem of sleep scoring of mice based on video data. Our procedure provides more accurate differentiation of the NREM and REM sleep states compared to any previous method in the field. The improvements in REM classification are particularly beneficial, as the dynamics of REM sleep are of special interest to sleep scientists. Furthermore, our procedure provides substantial improvements in capturing the sleep state bout duration distributions relative to other methods.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction: Classification and Application

1.1 Statistical Learning: Classification

Classification is the task of using a set of *input* variables, whose values are known, to predict one or more *output* variables, whose values are known in sample but are unknown out of sample. Statisticians have typically called the input variables the covariates, the predictors, or the independent variables while they have typically called the output variable(s) the response(s) or dependent variable(s). Regardless of the terminology, the goal is the same: use the inputs to predict the value of the output(s).

While output variable(s) can be continuous, *classical machine learning* focuses

on the problem of predicting a single, unordered categorical output from a set of continuous and/or categorical inputs (categorical variables are interchangeably referred to as qualitative variables, discrete variables, factors, or classes). The canonical example in statistics would be R. A. Fisher's work on Iris species discrimination (Anderson, 1935; Fisher, 1936). In this example, the output is a random variable Y which takes on values $y \in S \equiv \{$*Virginica*, *Setosa*, *Versicolor*$\}$ (we often assume $y \in S \equiv \{1, 2, ..., k\}$ except when $k = 2$ in which case we assume $S \equiv \{0, 1\}$ or $S \equiv \{\pm 1\}$; however, this is not always the case as shown by this Iris example). Input variables are denoted by X and in this example X takes on values $X = x$, a vector of four continuous variables corresponding to measurements of a particular flower's petal length, petal width, sepal length, and sepal width (in general, we let X and x denote a scalar or vector of length p; all vectors are assumed to be column vectors). The training data consists of $N = 150$ $(y_i, x_i)_{i=1}^N$ pairs where y_i is the species of a particular flower and x_i are its covariates (in general, we will denote by $\boldsymbol{y}$ the column vector $(y_1, \ldots, y_N)^T$ of outputs; we assume $\boldsymbol{y}$ is a realization of the vector random variable $Y_{1:N}$. We denote by $\boldsymbol{X}$ the N x p matrix whose rows are each of the x_i^T covariate vectors and also assume it is a realization of the matrix random variable $X_{1:N}$; when necessary, we will use $\boldsymbol{x}^j$ to refer to vector of N observations of the j^{th} covariate, similarly a realization of the random variable $X_{1:N}^j$.). Fisher assumed each of the (y_i, x_i) were drawn *i.i.d.* from a joint distribution $\mathbb{P}(Y, X)$ and modeled them via linear discriminant analysis.

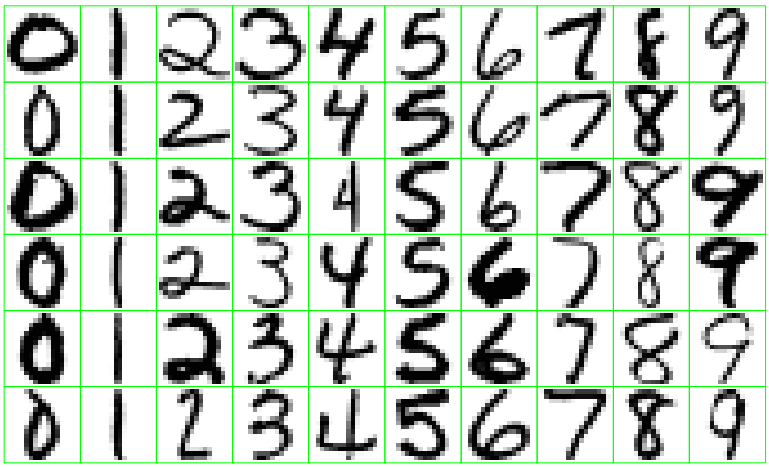

Figure 1.1: Examples of handwritten digits from U.S. postal envelopes. Source: Figure 1.2 in Hastie *et al.* (2001).

A standard machine learning example would be correctly identifying the letter $y \in S \equiv \{A,\ B,\ ...,\ Z\}$ or $y \in S \equiv \{0,\ 1,\ ...,\ 9\}$ based on covariates culled from a hand-written image X as in Figure 1.1 where the goal is to automatically read zip code digits from envelopes for the purpose of automated mail-sorting. What makes this example stand out as a classic machine learning one are three things. First, the number of covariates is potentially enormous and could include the grayscale values of the image over a large grid of pixels plus any other covariates one might create from these. Second, a "complex" function is likely required to map from the covariates to the predicted class; simple linear methods are unlikely to be suitable. Third, and perhaps most important and most in contrast with statistical problems, the signal to noise ratio for this problem is extremely high, possibly even infinite. This is clear because a human being could perform this task with near 100% accuracy. In contrast, in many statistical problems, the signal to noise ratio

is not nearly so high and often the noise dominates the signal.

A *classifier* is a function $f : X \to Y$ which maps from covariates X to classes Y and the goal of *classification* is to find a function $\hat{f}$ which correctly predicts the class of new (y, x) pairs via $\hat{y} = \hat{f}(x)$. This is usually accomplished by restricting oneself to a class of functions $\hat{f} \in \mathcal{F}$ and choosing a classifier which performs well on the training data without overfitting. For instance, in Fisher's example, he restricted himself to linear classifiers arising from the assumption that the covariates came from a three-component mixture of four-dimensional Gaussian distributions (three referring to the number of classes and four to the number of covariates).

1.2 Statistical Learning: Sequential Classification

While the classification paradigm is rich and fits many examples, it is also quite limited. Consider the example of predicting letters based on an image. If these are random letters, then the classification paradigm holds. However, if we suppose that these letters come in order from a word, then they follow a natural sequence and the *i.i.d.* assumption is clearly false. Rather than being drawn *i.i.d.* from $\mathbb{P}(Y, X)$, the (y, x) pairs are *sequences* which exhibit correlation from item to item. This case, where the Y random variables form a categorical sequence or time series $Y_1, Y_2, ...,$ is the *sequential learning* paradigm.

Many popular machine learning applications such as part-of-speech tagging, text-to-speech mapping, biological sequence analysis, and information extraction

from web pages fit this case. For instance, one might be interested in part-of-speech tagging where each X random variable is a word in a sentence and each Y random variable is the part of speech of that word (e.g., noun, adjective, verb, etc.). Or, perhaps one is interested in fraud detection where each X is the use of a credit card and Y takes on the value zero or one depending on whether the use was legitimate or fraudulent. Finally, Y could be one of the three sleep states (NREM, REM, and WAKE) and X could be data from video recordings, electroencephalographic (EEG) recordings, electromyographic (EMG) recordings, and piezoelectric recordings. In the sequel, we will focus on this problem using video recordings as our covariates.

In the part-of-speech example, English grammar (or that of any other language) enforces certain patterns on the data and makes others impossible or unlikely (e.g., the pattern "verb verb noun verb" is highly unlikely in English). In the fraud case, the sequence is likely to be all zeroes before a credit card is stolen and all ones after. For sleep states, biological mechanisms rule out certain sequences (e.g., no transitions from WAKE to REM). These temporal correlations can be harnessed to improve predictions.

Formally, the *sequential classification* task is defined as follows. We assume we have a set of N training sequences $\{(\boldsymbol{y}_i, \boldsymbol{X}_i)_{i=1}^N\}$ (often, $N = 1$) where training sequence i is measured for T_i time periods. For example, we might have consecutive sleep states and video recordings for several mice. Each training sequence consists of a sequence of outputs $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,T_i})^T$ and covariates $\boldsymbol{X}_i$, the T_i x p

matrix whose rows are the $x_{i,t}^T$ (where $x_{i,t}$ is the vector of covariates from training sequence i at time t). Here, the goal is to find a classifier $\hat{f}$ that correctly predicts a new sequence $\boldsymbol{y}$ given a new covariate sequence $\boldsymbol{X}$.

We assume each training sequence $\boldsymbol{y}_i$ is a realization of the vector valued random variable $Y_{i,1:T_i} = (Y_{i,1}, ..., Y_{i,T_i})^T$. Likewise, we assume the covariate matrix $\boldsymbol{X}_i$ is a realization of the random variable $X_{i,1:T_i}$ which has rows equal to $X_{i,t}^T$. In the sequential case, we assume $Y_{i,1:T_i}$ and $X_{i,1:T_i}$ are drawn jointly from the distribution $\mathbb{P}(Y_{1:T}, X_{1:T})$ which returns the $(1+p) \cdot T_i$ random variables that make up $Y_{1:T_i}$ and $X_{1:T_i}$.

The sequential learning task differs in important ways from two closely-related tasks. The first is time-series prediction where the typical goal is to predict Y_{t+1} given $Y_1, ..., Y_t$ (and possibly covariates $X_1, ..., X_{t+1}$). There are two key differences. First, in the sequential case, one has the whole time series of covariates $X_{1:T}$ with rows $\{X_t^T\}_{t=1}^T$ available for predicting future values Y_{t+1}; in the time series case, one typically only has $X_1, ..., X_{t+1}$ available for predicting Y_{t+1}. Utilization of the entire sequence $X_{1:T}$ can improve performance. Second, in sequential learning, we know none of the true values $Y_1, ..., Y_T$ (except in our training data) whereas, in time series forecasting, one typically knows the true $Y_1, ..., Y_t$ before predicting Y_{t+1} (or at least before predicting Y_{t+k}).

The second related task is *sequence classification*: predicting a single label Y for a whole sequence $X_{1:T}$. For example, $X_{1:T}$ could be a sequence of images of

hand-written letters and Y could be the word those letters form. Or, $X_{1:T}$ could be a sequence of images of hand-written letters and Y could be the person who wrote them (i.e., hand-writing identification). Clearly these problems are related to the problem of sequential learning. In fact, in the former case, a successful strategy for classifying words might be to sequentially classify letters and then string them together to form a word. However, no such strategy would work for the hand-writing identification task.

We note that both the generic learning task as well as the sequential learning task have *unsupervised* equivalents. That is, Y_t is assumed to belong to $S \equiv \{1, 2, ..., k\}$ however it is unknown and latent, even in the training data. Often, some form of the Expectation-Maximization (EM) algorithm (Dempster *et al.*, 1977) is used on the training data. The latent Y_t are estimated in the E-step and then the likelihood is maximized with respect to these estimates in the M-step. In the sequel, Y_t will be assumed to be known for the training data and hence we will be working in the supervised case.

1.3 Application to Sleep Data

Approximately 40 million Americans are afflicted with various sleep disorders such as insomnia, sleep apnea, and narcolepsy. As knowledge of these disorders grows amongst the populace, sleep medicine is becoming an increasingly important field of medical inquiry. Sleep scientists wish to establish the genetic basis of sleep behavior

in humans, but a typical sleep study using human subjects is incredibly expensive and time-consuming.

Mice are increasingly becoming the animal model for studying sleep. The advantages of using mice are the accessibility of many inbred strains as well as many recombinant inbreds which facilitate the identification of quantitative trait loci. Other advantages are the availability of congenics and consomics to facilitate gene identification and large-scale ethylnitrosourea (ENU) mutagenesis projects that have been and are being conducted in mice. All of these strategies are being undertaken to identify genes regulating biological processes such as sleep.

Genetic differences cause strains of mice to differ in both sleep behavior and prevalence of sleep disorders. Sleep researchers are particularly interested in which genes contribute to wakefulness versus sleep and, within sleep, REM sleep versus non-REM sleep. Sleep scientists are especially interested in the REM state of sleep which occurs much less frequently than either non-REM sleep or wakefulness.

Large-scale sleep studies of mice are not currently feasible because the gold standard methodology for the study of sleep behavior is invasive, expensive, and time-consuming. First, researchers implant a wire into the mouse's head for the purpose of electroencephalographic (EEG) and electromyographic (EMG) recordings. After waiting ten to fourteen days for the mouse to recover from surgery, EEG and EMG signals are recorded for twenty-four hours at 256Hz. These continuous EEG and EMG recordings are broken into ten second blocks (termed "epochs"). Each of

the 8,640 epochs that make up a day are then manually classified by trained technicians into three stages: REM sleep, non-REM (NREM) sleep, and wakefulness (WAKE).

The EEG/EMG system is besot with problems in addition to high cost and impracticality. Manual-scoring is internally and externally inconsistent. That is, there are disagreement rates between scorers: different scorers can disagree by as much as 8% overall and up to 15% within the important REM sleep stage. Moreover, the same person frequently scores the same epoch differently when he revisits the data at different times. Consequently, an automated procedure would have the additional benefit of following a fixed set of rules and producing internally consistent scores. Still, the high "human error" rate is a feature of this problem generally not encountered in the sequential classification literature (in contrast, a difficult sequential classification problem is part-of-speech tagging, where the human or Bayes error rate is essentially zero).

There is substantial interest in replacing this invasive, manual process with a high-throughput system based on video recordings of mice. In addition to vast savings in time and money, use of video data would avoid the costly surgery, recovery time, and confounding effects of the wire implantation. This is a very important problem for scientists who require accurate phenotypic screening of different genetic strains of mice.

The accurate classification of sleep states based on this video data is a daunting

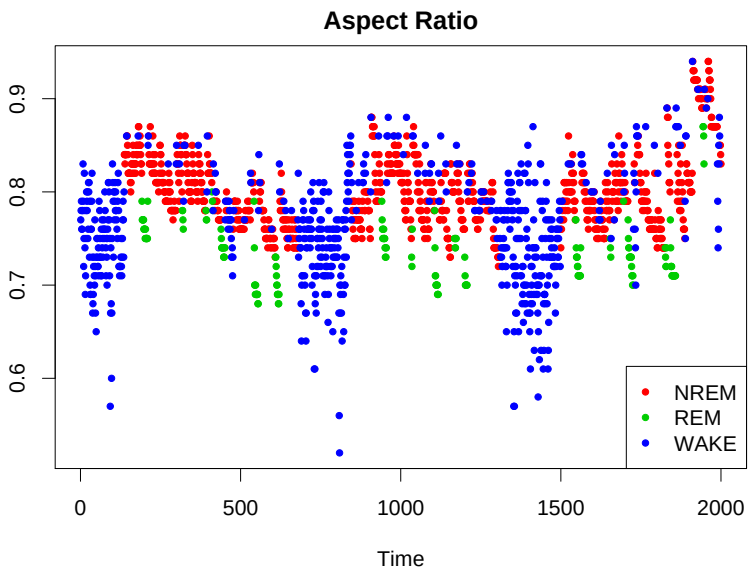

Figure 1.2: A time-series plot of the mean aspect ratio for one mouse, with colors corresponding to the manually scored sleep state.

proposition. Sleep scientists have already made initial efforts towards replacing manual EEG/EMG scoring with automated classification of sleep states based on video data alone using the so-called "40-second Rule"(Pack *et al.*, 2007). These results are useful for a limited set of purposes but are nevertheless quite inaccurate. These attempts have only been able to differentiate between sleep and wake, whereas the main interest of sleep researchers is differentiation between the NREM and REM sleep states. Sleep researchers are especially interested in the dynamics of the REM sleep state in different strains of mice. Unfortunately, REM and NREM behavior are very similar, a problem that is compounded by the fact that the REM sleep state is rare, occurring in only 5% of the manually-scored epochs. We shall show that with a more sophisticated analysis, this differentiation can be accomplished.

Despite these challenges, there is some hope that aspects of the data that could be used to differentiate NREM and REM. In Figure 1.2, the aspect ratio of the mouse gives evidence of subtle signal between REM and NREM. For instance, REM epochs appear to have lower aspect ratios than the corresponding NREM epochs just before and after it. Marginally, the discriminatory power of this feature is very limited, but there is clearly a time dependent error structure.

1.4 Overview

In the sequel, we will provide an overview of various statistical methods currently in use (Chapter 2). We will then discuss various difficulties these methods encounter in practice (Chapters 3 and 4). Next, we will introduce the PrAGMaTiSt: **Pr**ediction and **A**nalysis for **G**eneralized **Ma**rkov **Ti**me **Se**ries of **St**ates in Chapter 5 and discuss how it overcomes some of these difficulties.

We subject our methods and various competitors to rigorous tests on both simulated data (Chapter 6) and sleep data (Chapter 7). Finally, in Chapter 8, we provide a brief discussion of the main results of this work and areas for future research.

Chapter 2

Extant Methods

There is a vast literature on classification methods which we briefly review in this section. This review is by no means exhaustive. For a more thorough review on classification in general, we suggest Hastie *et al.* (2001) and Duda *et al.* (2001). For sequential classification, no classic texts exist though Dietterich (2002) provides a brief review.

2.1 Classification Methods

2.1.1 Classical Methods

The classical statistical workhouse for classification is **logistic regression**. Logistic regression models the probability of each of the k classes via linear functions of the covariates x. The model arbitrarily chooses one class (here the k^{th}) as the base class

and linearly models the log-odds of the other classes with respect to this class

$$\begin{aligned}
\log \frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=k|X=x)} &= x^T \cdot \beta_1 \\
\log \frac{\mathbb{P}(Y=2|X=x)}{\mathbb{P}(Y=k|X=x)} &= x^T \cdot \beta_2 \\
&\dots \\
\log \frac{\mathbb{P}(Y=k-1|X=x)}{\mathbb{P}(Y=k|X=x)} &= x^T \cdot \beta_{k-1}.
\end{aligned}$$

Here, a column of ones is usually joined to the observed covariate matrix $\boldsymbol{X}$ so that each β_k is of length $p+1$. Under this model, we can recover the conditional class probabilities as

$$\begin{aligned}
\mathbb{P}(Y=i|X=x) &= \frac{\exp(x^T \cdot \beta_i)}{1+\sum_{j=1}^{k-1}\exp(x^T \cdot \beta_j)},\ i=1...,k-1 \\
\mathbb{P}(Y=k|X=x) &= \frac{1}{1+\sum_{j=1}^{k-1}\exp(x^T \cdot \beta_j)}.
\end{aligned}$$

Logistic regression is typically estimated via maximization of its multinomial likelihood.

Though we do not use it here, it is also worth mentioning another classical statistical method, **linear discriminant analysis** (LDA), for it offers a parallel we will see when we discuss sequential methods. LDA models the class-conditional distribution of the covariate X as multivariate Gaussian with a covariance matrix

that is common across all classes. That is,

$$X|Y = i \sim N(\mu_i, \Sigma),\ i = 1, ..., k$$

where μ_i is a mean vector of length p and Σ is the common p x p covariance matrix. It also assumes the prior probability of each class is $\mathbb{P}(Y = i) = \pi_i$, $i = 1, ..., k$. By Bayes Theorem,

$$\mathbb{P}(Y = i|X = x) = \frac{\phi(x|\mu_i, \Sigma) \cdot \pi_i}{\sum_{j=1}^{k} \phi(x|\mu_j, \Sigma) \cdot \pi_j}$$

where $\phi(.|\mu, \Sigma)$ is the multivariate normal probability density function with mean μ and covariance matrix Σ. As in logistic regression, the model is typically fit via maximum likelihood (equivalent to moment estimation) and a linear decision boundary is formed.

Whereas logistic regression is a *discriminative* model, linear discriminant analysis is a *generative* model (despite the name). The difference is as follows. Generative classifiers model the joint distribution $\mathbb{P}(Y, X)$ of the classes and covariates and factor that distribution as $\mathbb{P}(Y)\mathbb{P}(X|Y)$. Typically, models are fit via maximization of the *joint* likelihood $\mathbb{P}(Y)\mathbb{P}(X|Y)$. On the other hand, discriminative classifiers such as logistic regression model only the *conditional* distribution of the classes given the covariates $\mathbb{P}(Y|X)$ and are typically fit via maximization of this *conditional* likelihood $\mathbb{P}(Y|X)$.

More specifically, in the former, the distribution of the covariates is a mixture

of Gaussians, $\mathbb{P}(X) = \sum_{i=1}^{k} \pi_i \cdot \phi(X|\mu_i, \Sigma)$. In the latter, though it appears $\mathbb{P}(X)$ is ignored, it is not: "we can think of this marginal density as being estimated in a fully nonparametric and unrestricted fashion, using the empirical distribution which places mass $1/N$ at each observation" (Hastie *et al.*, 2001). The former, by relying on these additional assumptions, can provide more efficient estimates of the model parameters. In fact, logistic regression requires 30% more data to do as well as LDA when the conditional covariate distributions are Gaussian (Efron, 1975).

As a prelude, we will see a similar difference between generative Hidden Markov Models (HMMs) on the one hand and discriminative Maximum Entropy Markov Models (MEMMs) and Conditional Random Fields (CRFs) on the other hand. A pivotal contribution of this research is to fit Markov models in a discriminative fashion, thus allowing the addition of assumptions on the state duration distributions not possible in MEMMs and CRFs and also providing increased efficiency.

2.1.2 CART

Tree-based methods split the space of the predictors into hyper-rectangles and then fit simple models to each one[1]. We will briefly discuss **CART** trees (Breiman *et al.*, 1984) in this section. Though we do not use CART for classification, CART trees form the basis for many of the algorithms we consider in the sequel. Furthermore, while CART trees are one of the most popular tree-growing algorithms, we note

[1]This section relies substantially on Hastie *et al.* (2001)

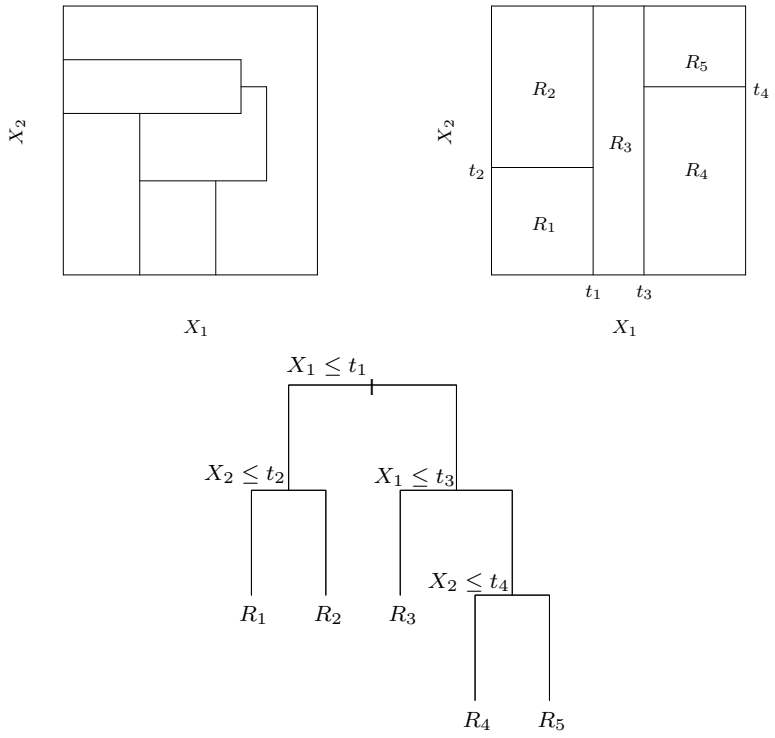

Figure 2.1: Partitions and CART. The top left panel shows a partition of a two-dimensional covariate space which cannot be obtained from recursive binary splitting. The top right shows one which can. The bottom panel shows the tree corresponding to the partition in the top right. Source: Figure 9.2 in Hastie *et al.* (2001).

that others exist (e.g., C4.5 (Quinlan, 1993)).

We start with a simple example. Consider a categorical response Y and two predictors X_1 and X_2. Figure 2.1 shows partitions of the covariate space. Within each partition, one could model the probability that Y takes on the various classes differently. For example, suppose there are N_m datapoints within partition R_m (corresponding to node m of a tree). Then, we could model $\mathbb{P}(Y = i|X \in R_m)$, the probability of that Y equals class i when the covariates are found in partition R_m

as

$$\hat{p}_{mi} = \frac{1}{N_m} \sum_{y_j | x_j \in R_m} I(y_j = i). \tag{2.1.1}$$

We can also classify all observations in partition R_m to class $\hat{i}(m) = \text{argmax}_i \, \hat{p}_{mi}$.

For simplicity, we only consider recursive binary partitions as in the top right panel of Figure 2.1. That is, we first split the space on one variable into two regions and then model Y within each of those two regions. The variable and the split point are both chosen to provide the best fit. Then, one or both of these regions is split into two regions and the process is continued until some stopping rule is applied.

Such a procedure can be viewed in partition form in the top right panel of Figure 2.1 and in tree form by the lower panel. First, X_1 is chosen as the split variable at split point t_1. Then, the partition corresponding to $X_1 \leq t_1$ is split again at $X_2 = t_2$ while the partition corresponding to $X_1 > t_1$ is split at $X_1 \leq t_3$. Finally, the partition given by $X_1 \leq t_3$ is split at $X_2 = t_4$.

A major question is how to determine which variable to split on and where to split. Usually, this is done in a greedy fashion. Given a dataset and no tree, we form the first split as follows. First, we restrict ourself to only splitting at values that are observed in our dataset. We can then consider all possible splits on all of our p variables (assuming continuous predictors with no ties, we would consider $p \cdot N$ splits). Each variable / split-point combination induces two nodes for which we can obtain probability predictions $\hat{p}_{mi}$ and class estimates as above.

In order to select among the $p \cdot N$ variable / split-point combinations, we utilize

an "impurity function" $Q_m(T)$ (to be defined below) for terminal node m of our tree T (here, since we are considering the first split, there are two terminal nodes). We then define the total tree impurity as $Q(T) = \sum_{m=1}^{|T|} N_m Q_m(T)$ where $|T|$ is the number of terminal nodes of tree T (again, since we are considering the first split, $|T| = 2$). Then, we select the variable / split point combination that minimizes total impurity. Finally, we repeat this process on the resulting two regions and repeat again on all the resulting regions.

This leaves two questions: what should our impurity function be and when do we stop growing the tree? There are three commonly used impurity measures:

$$\begin{aligned}
\text{Misclassification Error:} \quad & \frac{1}{N_m} \sum_{y_j | x_j \in R_m} I(y_j \neq \hat{i}(m)) = 1 - \hat{p}_{m\hat{i}(m)} \\
\text{Gini Index:} \quad & \sum_{i \neq i'} \hat{p}_{mi} \hat{p}_{mi'} = \sum_{i=1}^{k} \hat{p}_{mi}(1 - \hat{p}_{mi}) \\
\text{Cross-entropy or Deviance:} \quad & \sum_{i=1}^{k} \hat{p}_{mi} \log \hat{p}_{mi}.
\end{aligned}$$

A more extensive discussion of impurity measures is beyond the scope of this work. However, we note the latter two are typically employed because they are differentiable and therefore more tractable for numerical optimization (Hastie *et al.*, 2001).

As for stopping, there are several competing approaches. One simple approach involves recursively splitting until a split will produce a terminal node with fewer than some pre-specified number of observations. Another approach involves specifying a maximum number of terminal bins or a maximum depth size for the tree.

Finally, the *cost-complexity approach* grows a very large tree and then "prunes" it backwards. If we let T_0 be a large tree, we can define any subtree $T \subset T_0$ as a tree which can be obtained from T_0 by collapsing any number of non-terminal nodes. We can then define the cost-complexity of a subtree as

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T| = Q(T) + \alpha|T|. \tag{2.1.2}$$

$\alpha \geq 0$ is a parameter that governs the penalty due to the size of the tree, thus enforcing a tradeoff between goodness of fit to the training data and complexity of the tree. The goal is to find, for each α, the subtree $T_\alpha \subseteq T_0$ which minimizes $C_\alpha(T)$. Typically, the value $\alpha = \hat{\alpha}$ is chosen via cross-validation and then $T_{\hat{\alpha}}$ is chosen as the final tree (see Breiman *et al.* (1984) and Ripley (1996) for more details). For more technical issues such as categorical predictors, unequal misclassification costs, missing data, non-binary splits, splitting on linear combinations of predictors, and issues examining instability of trees, their lack of smoothness, and their difficulty in capturing additive structure, see Hastie *et al.* (2001).

2.1.3 Ensemble Methods

CART trees can form the building blocks to the four methods discussed in this section: bagged trees (Breiman, 1996), Adaboost (Freund and Schapire, 1996), LogitBoost (Friedman *et al.*, 2000a), and Random forests (Breiman, 2001). These methods form an "ensemble" or "forest" of trees and allow the trees to "vote" in

order to estimate probabilities and classify. They all work in different ways and are reviewed.

Two caveats are in order. First, these methods are primarily ensemble methods that consist of generating many "weak" or "base" learners. CART trees are one such base learner from which an ensemble can be formed but they are by no means the only one even if they are quite common. Second, this is not an exhaustive list of ensemble methods though they do include two of the most common kinds: randomized algorithms which use repeated bootstrap samples to reduce variance and avoid overfitting (bagged trees and Random forests) and adaptive, recursively weighted algorithms which minimize loss functions (AdaBoost and LogitBoost). Such methods stand in stark contrast to model-based approaches like logistic regression and LDA.

We begin with the simplest method, **bagged trees**. Before discussing this method, we must discuss the non-parametric bootstrap (Efron, 1979; Efron and Tibshirani, 1994). Given data $\boldsymbol{Z} = \{(y_1, x_1), ..., (y_N, x_N)\}$, a bootstrap sample $\boldsymbol{Z}^*$ is obtained by taking a sample of (x, y) pairs from $\boldsymbol{Z}$. Typically (and here), the the sample is of size N and sampling is done with replacement (see Buja and Stuetzle (2006) for other possibilities); furthermore, by convention we usually let the first bootstrapped sample be the original data $\boldsymbol{Z}$ itself.

In bagged trees, B bootstrap samples $\boldsymbol{Z}^{*b}, b = 1, ..., B$ are taken and a CART tree is fit to each one yielding an estimator $\hat{f}^{*b}(x)$ ($\hat{f}(x)$ returns the class estimate $\hat{i}(m)$

for the terminal node m in which datapoint x lands). The bagged tree estimator $\hat{f}_{bag}(x)$ is a vector of length k giving the proportion of the B trees which predicted class i, $i = 1, ..., k$. That is, each bootstrapped tree is given one "vote": the estimated class is the class that obtains the most votes and the estimated probabilities are the votes normalized by B, the number of bootstrap samples[2].

Now, each bootstrapped tree will typically use different variables and splitting points. Furthermore, they are also likely to have varying numbers of terminal bins. Hence, even though each tree is a "step function" (i.e., it returns a fixed class estimate or probability vector for each region), the bagged estimator which averages them can form complex, non-linear patterns. This can have the effect of reducing the variance of the procedure and thus improving prediction. That said, one of the primary advantages of CART trees is their interpretability and this is lost by bagging.

Where bagging forms a democratic committee whose members are all the same (i.e., they are all bootstrapped trees who each get one vote), *boosting algorithms* attempt to to combine many "weak learners" into a strong committee by varying both the individual committee members themselves as well as the number of votes each receives (Schapire, 1990; Freund, 1995; Freund and Schapire, 1997). The most popular such algorithm is **AdaBoost** (Freund and Schapire, 1996) which is also called *AdaBoost.M1* and is termed *Discrete AdaBoost* in the seminal Friedman *et al.*

[2]Alternatively, rather than using the class predictions $\hat{i}(m)$ at the terminal bin of each of the bootstrapped trees, one can use the probability predictions $\hat{p}_{mi}, i = 1, ..., k$ and average these over the B trees. For large B, these typically give similar results.

Algorithm 2.1 AdaBoost.M1 (Source: Hastie *et al.* (2009)).

1. Initialize weights $w_i = 1/N$, $i = 1, ..., N$.
2. For $m = 1$ to M:
   (a) Fit classifier f_m to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq f_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq f_m(x_i))], i = 1, ..., N$.
3. Output $F_M(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m f_m(x)]$.

(2000a) paper[3]. Though various modifications and improvements to the AdaBoost procedure have been proposed (Freund and Schapire, 1996; Breiman, 1998; Schapire and Singer, 1998; Friedman *et al.*, 2000a), we will focus on Discrete AdaBoost for the two-class problem and generalize later.

AdaBoost has had tremendous success in classification in real-world data settings, leading Breiman (1996) to call it the "best off-the-shelf classifier in the world". In addition to this, it has shown a remarkable resistance to overfitting in both simulated and real world datasets (Mease and Wyner, 2008; McShane, 2007). In fact, this was mentioned by three of the discussants (Breiman, 2000; Freund and Schapire, 2000; Buja, 2000) of Friedman *et al.* (2000a) (and one other discussant worked to construct counter-examples where it did not hold (Ridgeway, 2000)).

The AdaBoost algorithm is presented in Algorithm 2.1. It works by repeatedly applying a "weak" classifier to re-weighted versions of the data and thereby pro-

[3]The algorithm is called "discrete" because each weak learner returns a class estimate $y \in \{\pm 1\}$. In the case that the weak learner returns a probability estimate $\hat{p}(x) = \mathbb{P}(Y = 1|X = x)$, there is a modification to the algorithm known as *Real AdaBoost* (Friedman *et al.*, 2000a).

ducing classifiers $f_1(x), ..., f_M(x)$. These are combined via weighted majority vote to form $F_M(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m f_m(x)]$ where each classifier gets weight α_m.

Boosting is successful because, at each iteration, the datapoints which were incorrectly classified have their weights increased while those which were correctly classified have their weights decreased. Therefore, observations which are difficult to classify receive successively more weight and each new classifier focuses more heavily on those observations which have been missed by the previous ones. In addition to re-weighting observations, the individual classifiers are themselves weighted (by α_m) such that the more successful classifiers in the sequence (i.e., those with lower overall error) receive more votes and those that are less successful receive less.

Friedman *et al.* (2000a) show a number of important theoretical results about boosting which we briefly review. An additive model is one that can expressed as a basis expansion, $F(x) = \sum_{m=1}^{M} \beta_m b(x|\gamma_m)$ where β_m are the basis coefficients, $b(x|\gamma)$ is a simple real-valued function, and γ are parameters of that function. In boosting, we typically let $b(x|\gamma)$ be a CART tree; therefore, γ gives the split variables, split points, and terminal node probabilities $\hat{p}_{mi}$ and/or classifications $\hat{i}(m)$.

Since optimizing over all $\{\beta_m\}_{m=1}^{M}$ and $\{\gamma_m\}_{m=1}^{M}$ is computationally infeasible, boosting fits the additive model in a forward, stagewise fashion. That is, rather than minimizing

$$(\{\beta_m\}_{m=1}^{M}, \{\gamma_m\}_{m=1}^{M}) = \underset{\{\beta_j\},\{\gamma_j\}}{\text{argmin}} \sum_{i=1}^{N} L(y_i, F(x_i))$$

where $F(x) = \sum_{m=1}^{M} \beta_m b(x|\gamma_m)$, boosting proceeds incrementally and inductively. It sets $f_0(x) = 0$ and, at each iteration $m = 1, ..., M$, it minimizes

$$(\beta_m, \gamma_m) = \underset{\beta,\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta b(x_i|\gamma))$$

where $F_m(x) = F_{m-1}(x_i) + \beta b(x_i|\gamma)$.

Given that boosting is forward, stagewise additive modeling, a natural question is to ask what loss function is being minimized. Friedman *et al.* (2000a) show that AdaBoosts minimizes the exponential loss function, $L(y, f(x)) = \exp(-yf(x))$. At each stage, AdaBoost solves

$$(\beta_m, f_m) = \underset{\beta,f}{\operatorname{argmin}} \sum_{i=1}^{N} \exp[-y_i(F_{m-1}(x_i) + \beta f(x_i))] \tag{2.1.3}$$

$$= \underset{\beta,f}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^m \exp[-\beta y_i f(x_i)] \tag{2.1.4}$$

where f is a "weak learner" CART tree and $w_i^m = \exp(-y_i F_{m-1}(x_i))$. This is because, for any $\beta > 0$, the solution to Equation 2.1.3 is

$$f_m(x) = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^m I(y_i \neq f(x_i))$$

and, for this f_m, the solution for β is

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i^m I(y_i \neq f_m(x_i))}{\sum_{i=1}^{N} w_i^m}.$$

Hence, AdaBoost approximately minimizes exponential loss in a forward, stagewise, additive manner.

Naturally, one might ask why the exponential loss function and not some other one. Friedman *et al.* (2000a) also posed this question, leading them to develop a new algorithm, **LogitBoost**. It can be easily shown that the population minimizer of exponential loss is

$$F(x) = \underset{G}{\operatorname{argmin}}\, \mathbb{E}(e^{-YG(x)}|X = x) = \frac{1}{2} \log \frac{\mathbb{P}(Y = 1|x)}{\mathbb{P}(Y = -1|x)}$$

implying $\mathbb{P}(Y = 1|x) = e^{2F(x)}/(1 + e^{2F(x)})$. This provides justification for the use of exponential loss for classification.

However, there is another popular loss function which has exactly the same minimizer: the negative of the binomial log likelihood (equivalent to the deviance or cross-entropy). If $\tilde{Y} = Y/2+1/2 \in \{0, 1\}$, then this loss is $L(y, p(x)) = \tilde{Y} \log p(x)+ (1 - \tilde{Y}) \log(1 - p(x))$ where $p(x) = e^{2F(x)}/(1 + e^{2F(x)})$. Though both binomial and exponential loss have the same population minimizer, they clearly will not have the same minimizer in a finite population, particularly when approximated via forward, stagewise additive modeling.

In classification when $Y \in \{\pm 1\}$, the margin $yf(x)$ plays a similar role to regression residuals $(y - f(x))$. Positive margin datapoints are classified correctly whereas negative margin ones are classified incorrectly. Since the goal of classification is to produce good class estimates, negative margins should receive a greater

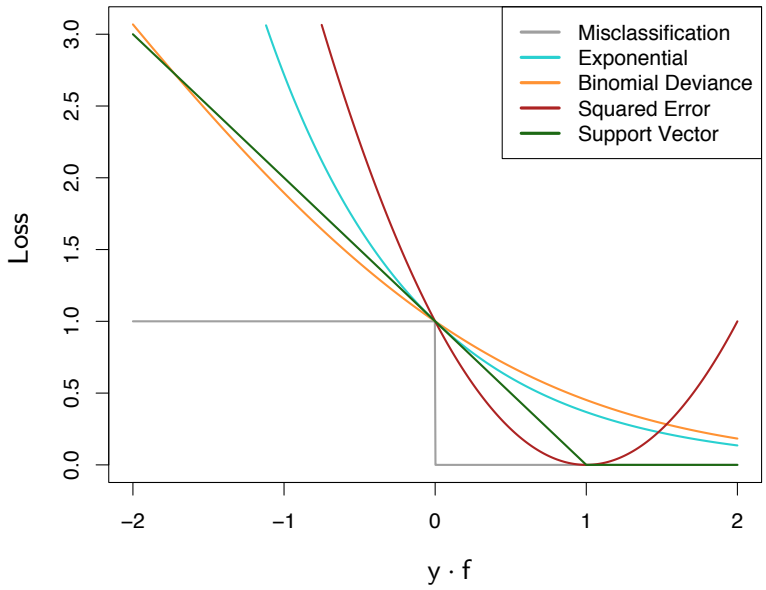

Figure 2.2: Loss functions for two-class classification. The response is $y = \pm 1$; the prediction is f, with class prediction $\text{sign}(f)$. The losses are misclassification: $I(\text{sign}(f) \neq y)$; exponential: $\exp(-yf)$; binomial deviance: $\log(1 + \exp(-2yf))$; squared error: $(y - f)^2$; and support vector: $(1 - yf)_+$ Each function has been scaled so that it passes through the point (0,1). Source: Figure 10.4 in Hastie *et al.* (2009).

Algorithm 2.2 LogitBoost (Source: Friedman *et al.* (2000a)).

1. Start with weights $w_i = 1/N$, $i = 1, ..., N$, $F_0(x) = 0$, and probability estimates $p(x_i) = 1/2$. Let $y_i \in \{\pm 1\}$ and $y_i^* = y_i/2 + 1/2 \in \{0, 1\}$.
2. Repeat for $m = 1, ..., M$:
   (a) Compute the working response and weights

$$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))}$$

$$w_i = p(x_i)(1 - p(x_i)).$$

   (b) Fit the function $f_m(x)$ by a weighted least-squares regression of z_i to x_i using weights w_i.
   (c) Set $F_m(x) = F_{m-1}(x) + \frac{1}{2} f_m(x)$ and update $p(x_i) \leftarrow e^{2F(x)}/(1 + e^{2F(x)})$.
3. Output the classifier $\text{sign}[F_M(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$.

penalty than positive ones. Various loss functions appear in Figure 2.2 and all can be viewed as convex approximations of misclassification loss, with all being continuous except for support vector loss and all being monotonically decreasing except squared error loss.

As can be seen from Figure 2.2, exponential loss places a much larger penalty on large negative margin observations than binomial deviance. Hence, it is suggested that it is sensitive to outliers. Binomial deviance will not place as much influence on such observations: "It is therefore far more robust in noisy settings where the Bayes error rate is not close to zero, and especially in situations where there is misspecification of the class labels in the training data. The performance of AdaBoost has been empirically observed to dramatically degrade in such settings" (Hastie *et al.*, 2009). Hence, Friedman *et al.* (2000a) suggest LogitBoost (Algorithm 2.2) as a robust alternative to AdaBoost which does not suffer from such features.

Before proceeding, we note that we have focused on the binary classification task. Both AdaBoost and LogitBoost can be extended to deal with multiple classes. One way to do this is simply to fit k classifiers where each is fit to response $y_{ik} = I(y_i = k)$. This is the "one versus every other" approach which is essentially equivalent to the augmented data approach of *AdaBoost.MH* (Schapire and Singer, 1998). In this case, one predicts the class which has the maximal $F_{Mi}(x)$. Another approach, which can be computationally demanding for large k, is the pairwise comparison approach where the classifier is fit to each (i, i') pair such that $i \neq i$; besides computational difficulty, this approach does not enforce transitivity and thus it is possible for one class to beat a second, the second a third, but the third to beat the first. For LogitBoost, Friedman *et al.* (2000a) propose the natural modification of the algorithm which uses multinomial loss instead of binomial.

Also, there are vast numbers of details we have omitted: why to boost trees, other forms of boosting and choices of gradients (gradient boosting, steepest descent, stochastic gradient boosting, etc.), regularization of the individual trees (via minimum bin size, maximum number of terminal nodes, penalized complexity functions, etc.), and regularization of the ensemble (via shrinkage of the step size, sampling or subsampling the data at each iteration, etc.). There is a wide literature on these topics but detailed treatment is beyond the scope of this paper.

The last algorithm we consider in depth is **Random forests** (Breiman, 2001). Random forests are a form of bagged trees grown in a very special way (see Algo-

Algorithm 2.3 Random Forests for Regression or Classification (Source: Hastie *et al.* (2009)).

1. For $b = 1, ..., B$:
   (a) Draw bootstrap sample $\boldsymbol{Z}^*$ of size N from the training data.
   (b) Grow a Random forest tree T_b to the bootsrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      (i) Select m variables at random from the p variables.
      (ii) Pick the best variable / split point among the m.
      (iii) Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.

To make a prediction at a new point x:

Regression: $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the b^{th} Random forest tree. Then $\hat{C}_{rf}^{B}(x) =$ majority vote $\{C_b(x)\}_{b=1}^{B}$. Similarly, probability estimates can be obtained by normalizing the votes by B.

rithm 2.3). The evolution of Random forests proceeded as follows. Bagging creates a committee of members trained in the same way, each receiving one vote. Boosting generalized this by training the committee members each in a different way (via the ever-changing weights) and allowing each member a different number of votes; it appears to dominate bagging on a panoply of applied problems. Random forests was invented as an implementation of bagging that appears to perform similarly to boosting but with the added advantage that it is easier to train and requires fewer tuning parameters.

Random forests are easier to train because, among other things, the trees can be grown in parallel (unlike in boosting where the iterative re-weighting requires the trees to be grown serially). Moreover, there are few tuning parameters: m, n_{min}, B. In fact, B is hardly a tuning parameter since the algorithm appears

relatively insensitive to changes in B provided it is "large enough". Furthermore, for classification, Breiman (2001) suggests using a default values of $n_{min} = 1$ and $m = p^{1/2}$ (for regression, he suggests $n_{min} = 5$ and $m = p/3$). They can be treated as tuning parameters or simply fixed to these defaults.

Further simplifying computational matters is the notion of *out of bag* (OOB) samples. For each sample datapoint (y_i, x_i), one can construct a Random forest predictor by only considering the trees for which it was omitted from the bootstrap samples. It turns out that OOB error estimates are very similar to those obtained by multiple K-fold cross-validation. Hence, unlike other algorithms, one can fit a single Random forest sequence and cross-validate along that sequence. Thus, one can fix B (i.e., stop the algorithm) once the OOB error rate flattens.

It is thought that the advantage of a Random forest over bagged trees comes from the fact that the trees in a Random forest are *less correlated* with one another than a set of bagged trees. This is because at each split point a random $m < p$ variables are chosen to split on. That is, there is an additional layer of randomness: not only is there the randomness of the bootstrap sample $\boldsymbol{Z}^*$ itself, but also there is randomness due to sampling m covariates at each split (of course there is also randomness due to the sampling distribution of $\boldsymbol{Z}$, the original sample data, itself). As in bagged trees, the bias of a Random forest ensemble is the same as that of any particular tree.

To conclude, though there are a vast number of algorithms (for example, the

gradient boosting machine (Friedman, 2001), stochastic gradient boosting (Friedman, 2002), MART (Friedman, 2001), and mboost (Hothorn and Buhlmann, 2002)) we have not considered in this section, we have considered several of the main contenders from the various classes: model based methods like logistic regression, iteratively re-weighted methods like AdaBoost and LogitBoost, and randomization algorithms like bagged trees and Random forests. In the sequel, we will apply these methods to various simulated and real world datasets.

2.2 Sequential Classification Methods

In this section, we return to the focus of this paper: classification for sequential data. To review, our dataset consists of N training sequences $\{(\boldsymbol{y}_i, \boldsymbol{X}_i)_{i=1}^N\}$ (often, $N = 1$) each of length T_i. Each training sequence consists of a sequence of outputs $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,T_i})^T$ and covariates given by the T_i x p matrix $\boldsymbol{X}_i$ whose rows are the $x_{i,t}^T$. We take $\boldsymbol{y}_i$ and $\boldsymbol{X}_i$ to be realizations of the random variables $Y_{i,1:T_i}$ and $X_{i,1:T_i}$ who have joint distribution $\mathbb{P}(Y_{1:T}, X_{1:T})$. Finally, the goal is to find a classifier $\hat{f}$ that correctly predicts a new sequence $\boldsymbol{y}$ given a new observed covariate sequence $\boldsymbol{X}$.

2.2.1 Data Augmentation Methods

The simplest approach to sequential classification involves augmenting the covariate space and applying standard classification methods. That is, the sequential learning

problem is converted into a classical learning problem. This approach is termed the **sliding window** method. Each covariate vector $X_{i,t} = x_{i,t}$ is augmented with the $d = (w-1)/2$ (w must be odd) preceding covariate vectors and the $d = (w-1)/2$ succeeding covariate vectors: $\tilde{x}_{i,t} = (x_{i,t-d}^T, x_{i,t-d+1}^T, ..., x_{i,t}^T, ..., x_{i,t+d-1}^T, x_{i,t+d}^T)^T$ (the first d and last d observations of the covariate vector can either be discarded or augmented with sensible values like the means). Hence, the covariates now follow a sliding window of window size w and the number of covariates increases from p to $w \cdot p$.

Now, *any* standard classification algorithm can be applied to the augmented dataset $\{(\boldsymbol{y}_i, \tilde{\boldsymbol{X}}_i)_{i=1}^N\}$ where $\tilde{\boldsymbol{X}}_i$ is the T_i x $p \cdot w$ matrix whose rows are the $\tilde{x}_{i,t}^T$. This will give a prediction for each $y_{i,t}$ which can be strung together to form a prediction for $\boldsymbol{y}_i$.

If one is concerned about the number of parameters being too large, one can modify this approach. For instance, instead of augmenting with all d preceding and succeeding values, one could consider functions of them. For example, one could consider using forward and backward moving averages of order d, thus only increasing the number of covariates from p to $3p$ rather than $w \cdot p$.

While major advantages of the sliding window approach include the ability to use any standard classification algorithm as well as the fact that standard classification algorithms in general have lower computational costs than sequential classification algorithms (even when the former are trained on the augmented covariate space and

the latter are not), there are some drawbacks. Mainly, sliding window approaches cannot capture any correlation among the Y_t which are independent of the neighboring X_t: the only intertemporal correlations among the Y_t that are captured are those which are predictable from local X_t.

A potential remedy to this is provided by **recurrent sliding windows**. In this case, one not only augments the covariates with local preceding and succeeding covariates, but also with local preceding predictions $\hat{y}_{i,t}$ such that $\tilde{x}_{i,t} = (\hat{y}_{i,t-d}, \hat{y}_{i,t-d+1}, ..., \hat{y}_{i,t-1}, x^T_{i,t-d}, x^T_{i,t-d+1}, ..., x^T_{i,t}, ..., x^T_{i,t+d-1}, x^T_{i,t+d})^T$ and $\tilde{\boldsymbol{X}}_i$ is now a matrix of size T_i x $(d + p \cdot w)$ (alternatively, one can use the d succeeding predictions $\hat{y}_{i,t+1}, ..., \hat{y}_{i,t+d}$; both preceding and succeeding predictions cannot be used). As with sliding windows, one can now apply standard classification algorithms.

A major question for recurrent sliding windows is what $\hat{y}_{i,t}$ should be used when training the algorithm. One popular approach is to use the actual $y_{i,t}$ themselves and then one can apply standard classification algorithms with no modifications whatsoever. Another approach is to first fit using a non-recurrent sliding window approach; then one uses the predictions $\hat{y}_{i,t}$ from that to train a recurrent sliding window classifier. Finally, one iterates until the $\hat{y}_{i,t}$ stabilize.

Algorithm 2.4 Generating Data From a Hidden Markov Model

1. Draw Y_1 from the initialization distribution, $\mathbb{P}(Y_1)$.
2. Draw X_1 from the covariate emission distribution, $\mathbb{P}(X_1|Y_1)$.
2. For $t = 2, ..., T$:
   (a) Draw Y_t from the transition probability distribution, $\mathbb{P}(Y_t|Y_{t-1})$.
   (b) Draw X_t from the covariate emission distribution, $\mathbb{P}(X_t|Y_t)$.

2.2.2 Hidden Markov Models

A Hidden Markov Model (HMM)[4] is a joint probability model for the distribution of the $Y_{i,1:T_i}$ and $X_{i,1:T_i}$ (going forward, for simplicity, we will assume we observe one sequence and drop the subscript i). HMMs decompose the joint distribution $\mathbb{P}(Y_{1:T}, X_{1:T})$ into three components: (i) an initialization distribution, $\mathbb{P}(Y_1)$; (ii) a time-homogeneous transition probability distribution, $\mathbb{P}(Y_t|Y_{t-1})$; and (iii) a set of (potentially multivariate) time-homogeneous conditional covariate distributions, $\mathbb{P}(X_t|Y_t)$, which are also known as the observation distributions or emission distributions. This is a *fully generative model* in the sense that, given these three distributions, one can generate sequences $\boldsymbol{y}$ and $\boldsymbol{X}$ from the model as visualized in Figure 2.3 and provided in algorithmic form in Algorithm 2.4.

In the supervised version of the sequential learning problem, estimation is relatively straightforward because the Y_t are only "hidden" in the test set but are observed in the training set[5]. We can estimate the initialization distribution, $\mathbb{P}(Y_1)$, by

[4]In this subsection, we will consistently refer to the model as a Hidden Markov Model (HMM) following Rabiner (1989). However, in the sequel, since the Y_t will be observed in sample, we will typically drop the "Hidden" and refer to it as a first order Markov model since the Y_t are only "hidden" out of sample.

[5]In the unsupervised case, the Y_t are hidden in both the training set as well the test set either because they are unavailable or are latent and therefore by definition unobservable.

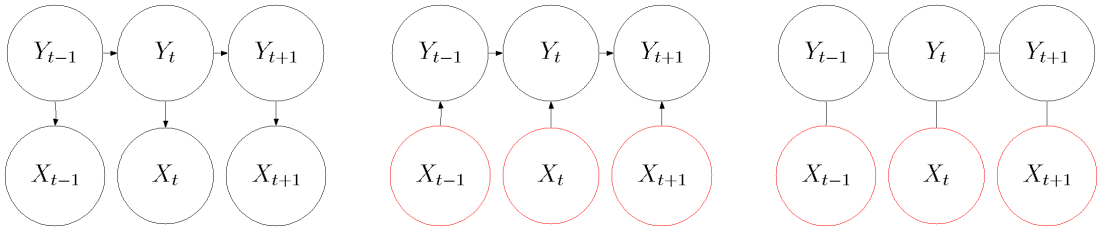

Figure 2.3: Graphical structures of HMMs (left), MEMMs (center), and CRFs (right) for sequences. A red circle indicates that the variable is not generated by the model.

the empirical frequency of each of the k classes (or by the empirical frequency of the $Y_{i,1}$'s if we observe multiple sequences) and the transition distribution, $\mathbb{P}(Y_t|Y_{t-1})$, by the empirical rate of transition from state i to state j (of course, more sophisticated strategies, for example shrinking to a common distribution, are possible).

Estimation of the covariate emission distribution is more complex. Oftentimes, one assumes the components X_{tj} of $X_t = (x_{t1}, ..., x_{tp})^T$ are independent. In this case, if X_{tj} is discrete, one can estimate the parameters of a multinomial distribution conditional on the observed Y_t. For continuous X_{tj}, a normal distribution is often assumed. More sophisticated strategies include estimating X_t conditional on Y_t as (i) multivariate normal, (ii) a mixture of multivariate normals (often with constrained covariance matrix), or (iii) via kernel density estimators. One is faced with a difficult tradeoff: estimate more complex, realistic models or estimate simpler, incorrect models with fewer parameters. This is particularly an issue when the number of observations of X_t conditional on Y_t is small, an issue which will not be mitigated in large datasets if there are one or more rare classes.

We now introduce some notation. First, we let $\pi = (\pi_1, ..., \pi_k)^T$ denote the vector of initial probabilities, $\mathbb{P}(Y_1 = i)$. Second, we let $\boldsymbol{A} = [a_{i,j}]_{i=1,...,k;j=1,...,k}$ be the transition probability matrix whose entries are $a_{i,j} = \mathbb{P}(Y_{t+1} = j|Y_t = i)$. Finally, let $\boldsymbol{\mu} = (\mu_1(x), ..., \mu_k(x))^T$ be a vector of probability measures on the covariate space such that each μ_i gives the conditional covariate probability $\mu_i(x) = \mathbb{P}(X_t = x|Y_t = i)$. We let $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu})$ denote the collection of these distributions.

In the classic reference on HMMs, Rabiner (1989) identifies three basic problems for HMMs:

1. Given an observed covariate sequence $x_1, ..., x_T$ and a model $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu})$, how does one efficiently compute $\mathbb{P}(\boldsymbol{X}|\boldsymbol{\Theta})$, the probability of observing the covariates one did given the model? This allows one to judge how well a model matches the covariate sequence and thus compare competing models.

2. Given an observed covariate sequence $x_1, ..., x_T$ and a model $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu})$, how does one choose a state sequence $\boldsymbol{y} = (y_1, ..., y_T)^T$ which is optimal in some meaningful sense? This allows us to predict the $\boldsymbol{y}$ (on the training data for the unsupervised case and on new test data for the supervised case).

3. How do we adjust the model parameters $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu})$ to maximize $\mathbb{P}(X_{1:T} = \boldsymbol{X}|\boldsymbol{\Theta})$? This allows us to train the HMM to best reflect the covariate sequence.

The solutions to the first two problems are critical to our work and will be presented below. The third problem is less relevant because it is exclusively for the

unsupervised case where the Y_t are not even known in the training sample (Rabiner (1989) solves it, via the EM algorithm, only for the case of scalar, discrete X_t). We will discuss the analogue for the supervised case in Chapter 5.

Calculating $\mathbb{P}(X_{1:T}|\boldsymbol{\Theta})$ is a daunting proposition. However, calculating $\mathbb{P}(X_{1:T}|\boldsymbol{y}, \boldsymbol{\Theta})$ is a seemingly trivial matter

$$\begin{aligned}\mathbb{P}(X_{1:T}|\boldsymbol{y}, \boldsymbol{\Theta}) &= \prod_{t=1}^{T} \mathbb{P}(X_t|Y_t, \boldsymbol{\Theta}) \\ &= \prod_{t=1}^{T} \mu_{Y_t}(X_t).\end{aligned}$$

Furthermore, calculating $\mathbb{P}(Y_{1:T}|\boldsymbol{\Theta})$ is also easy

$$\mathbb{P}(Y_{1:T}|\boldsymbol{\Theta}) = \pi_{Y_1} \cdot a_{Y_1,Y_2} \cdot a_{Y_2,Y_3} \cdots a_{Y_{T-1},Y_T}.$$

Now, since $\mathbb{P}(X_{1:T}|\boldsymbol{\Theta}) = \mathbb{P}(X_{1:T}|Y_{1:T}, \boldsymbol{\Theta})\mathbb{P}(Y_{1:t}|\boldsymbol{\Theta})$, then it clear that

$$\begin{aligned}\mathbb{P}(X_{1:T}|\boldsymbol{\Theta}) &= \sum_{\text{all } Y_{1:T}=\boldsymbol{y}} \mathbb{P}(X_{1:T}|Y_{1:T} = \boldsymbol{y}, \boldsymbol{\Theta})\mathbb{P}(Y_{1:T} = \boldsymbol{y}|\boldsymbol{\Theta}) \\ &= \sum_{\text{all } Y_{1:T}=\boldsymbol{y}} \pi_{Y_1}\mu_{Y_1}(X_1)a_{Y_1,Y_2}\mu_{Y_2}X_2 \cdots a_{Y_{T-1},Y_T}\mu_{Y_T}X_T.\end{aligned}$$

Unfortunately, this is computationally unfeasible (it requires $\approx 2T \cdot k^T$ calculations).

Thus, a more efficient approach is requisite. Fortunately, such an approach, the **Forward-Backward Algorithm**, exists (Rabiner, 1989) and is presented in Algorithms 2.5 and 2.6. Technically, only the forward portion is required to compute

Algorithm 2.5 The Forward Algorithm (Source: Rabiner (1989)).

Define $\alpha_t(i) = \mathbb{P}(X_1 = x_1, ..., X_T = x_t, Y_t = i|\boldsymbol{\Theta})$.

1. Initialization: $\alpha_1(i) = \pi_i \mu_i(x_1),\ i = 1, ..., k.$
2. Induction: $\alpha_{t+1}(j) = \left[\sum_{i=1}^{k} \alpha_t(i) a_{i,j}\right] \mu_j(x_{t+1}),\ t = 1, ..., T-1;\ i = 1, ..., k.$
3. Termination: $\mathbb{P}(X_{1:T} = \boldsymbol{X}|\boldsymbol{\Theta}) = \sum_{i=1}^{k} \alpha_T(i).$

Algorithm 2.6 The Backward Algorithm (Source: Rabiner (1989)).

Define $\beta_t(i) = \mathbb{P}(X_{t+1} = x_{t+1}, ..., X_T = x_T|Y_t = i, \boldsymbol{\Theta})$.

1. Initialization: $\beta_T(i) = 1,\ i = 1, ..., k.$
2. Induction: $\beta_t(i) = \sum_{j=1}^{k} a_{i,j} \mu_j(x_{t+1}) \beta_{t+1}(j),\ t = T-1, ..., 1;\ i = 1, ..., k.$

$\mathbb{P}(X_{1:T}|\boldsymbol{\Theta})$ (the backward portion is used for the solution to problem two but it makes sense to present them together). Each portion of the algorithm requires only $\approx k^2 T$ calculations, thus resulting in substantial savings over the naive method.

Unlike with the first problem where there is an exact solution, the second problem is more tricky. Not only will what is "optimal" depend on context (i.e., loss function), but it also might be the case that several sequences $Y_{1:T} = \boldsymbol{y}$ are equally optimal. That said, two approaches are usually considered. The first finds the state which is most likely at each time t (i.e., yielding the pointwise modal sequence) and the second finds *a* most likely sequence (i.e., a sequence $\boldsymbol{y}$ which maximizes $\mathbb{P}(Y_{1:T} = \boldsymbol{y}|X_{1:T} = \boldsymbol{X}, \boldsymbol{\Theta})$ and which may or may not be unique).

To find the pointwise modal sequence, we first define $\gamma_t(i) = \mathbb{P}(Y_t = i|\boldsymbol{\Theta})$. Once one has run the forward-backward algorithm, it is easy to calculate $\gamma_t(i)$ because

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\mathbb{P}(X_{1:T} = \boldsymbol{X}|\boldsymbol{\Theta})} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{k} \alpha_t(j)\beta_t(j)}.$$

Algorithm 2.7 The Viterbi Algorithm (Source: Rabiner (1989)).

Define $\delta_t(i) = \max_{y_1,...,y_{t-1}} \mathbb{P}(Y_1 = y_1, Y_2 = y_2, ..., Y_{t-1} = y_{t-1}, Y_t = i, X_1 = x_1, X_2 = x_2, ..., X_T = x_t|\boldsymbol{\Theta})$.

1. Initialization:
$$\begin{aligned} \delta_1(i) &= \pi_i \mu_i(x_1) \quad i = 1, ..., k \\ \psi_1(i) &= 0 \qquad\quad i = 1, ..., k. \end{aligned}$$

2. Recursion:
$$\begin{aligned} \delta_t(j) &= \max_{i=1,...,k}[\delta_{t-1}(i)a_{i,j}]\mu_j(x_t) \quad t = 2, ..., T;\ j = 1, ..., k \\ \psi_t(j) &= \text{argmax}_{i=1,...,k}[\delta_{t-1}(i)a_{i,j}] \qquad\quad t = 2, ..., T;\ j = 1, ..., k. \end{aligned}$$

3. Termination:
$$\begin{aligned} P^* &= \max_{i=1,...,k} \delta_T(i) \\ y_T^* &= \text{argmax}_{i=1,...,k} \delta_T(i). \end{aligned}$$

4. Path (state sequence) backtracking: $y_t^* = \psi_{t+1}(y_{t+1}^*)$, $t = T-1, ..., 1$.

Using this, the most likely state for each t is simply $y_t = \text{argmax}_{i=1,...,k} \gamma_t(i)$.

A sequence $\boldsymbol{y}^*$ which maximizes the conditional likelihood $\mathbb{P}(Y_{1:T} = \boldsymbol{y}|X_{1:T} = \boldsymbol{X}, \boldsymbol{\Theta})$ can be found by the Viterbi Algorithm (Algorithm 2.7). Because this sequence is "most likely" in a reasonable sense, it is useful for many problems. However, it is not a complete panacea because, depending on one's loss function, the Viterbi sequence might not be optimal.

Now, because an HMM is a full representation of the joint probability $\mathbb{P}(Y_{1:T}, X_{1:T})$, one can compute the probability of any sequence $\boldsymbol{y}$ given a covariate sequence $\boldsymbol{X}$ using $\mathbb{P}(Y_{1:T}, X_{1:T})$. Equipped with this, for any arbitrary loss function $L(Y_{1:T}, \hat{\boldsymbol{y}})$, one can predict the optimal sequence $\hat{\boldsymbol{y}}$ by

$$\hat{\boldsymbol{y}} = \underset{\tilde{\boldsymbol{y}}}{\text{argmin}} \sum_{Y_{1:T}=\boldsymbol{y}} \mathbb{P}(Y_{1:T} = \boldsymbol{y}|X_{1:T} = \boldsymbol{X}) L(Y_{1:t} = \boldsymbol{y}, \tilde{\boldsymbol{y}})$$

Unfortunately, this requires $\approx k^T$ calculations which is usually infeasible. Hence, we must rely on more computationally feasible $\hat{\boldsymbol{y}}$'s such as the modal or Viterbi sequence.

Although HMMs are a clean and elegant model, they have a number of drawbacks which stem from the fact that their structure is often not indicative of the data generating processes encountered in applied problems. For instance, due to the first order Markov property, long-term dependencies in the Y_t's cannot be captured: any relationship between Y_t and Y_{t+k} must be "communicated" via $Y_{t+1}, Y_{t+2}, ..., Y_{t+k-1}$. Sliding windows are able to avoid this problem somewhat by using a window of X_t values to predict Y_t but this cannot be done with an HMM: since an HMM generates X_t conditional on Y_t it is difficult to use a sliding window. While one could consider replacing the conditional covariate distribution $\mathbb{P}(X_t|Y_t)$ by a more complicated distribution $\mathbb{P}(X_t|Y_{t-d}, ..., Y_t, ..., Y_{t+d})$ (which would imply a sliding window of covariates), it would be difficult to model and estimate such a distribution. A final difficulty is that, when the dimension of the covariate space p is large (even without sliding windows), it is difficult to model and estimate $\boldsymbol{\mu}$, the k vector of p-variate probability measures.

In the sequel, we will overcome these difficulties in several ways. First, we will allow richer dependence structures among the Y_t: in addition to first order Markov chains, we will allow for higher order Markov chains, generalized Markov chains, and variable length Markov chains. Second, by training our model in a *discriminative*

fashion rather than a *generative* fashion, we will avoid the issues involved with estimating a k vector of p-variate probability measures as well as the intractability of adding sliding windows of covariates.

2.2.3 Conditional Approaches

The methods presented in this section, Maximum Entropy Markov Models (MEMMs) (McCallum *et al.*, 2000) and Conditional Random Fields (CRFs) (Lafferty *et al.*, 2001), attempt to overcome some of the limitations of HMMs by maximizing a *conditional* likelihood rather than the *joint* likelihood $\mathbb{P}(Y_{1:T} = \boldsymbol{y}, X_{1:T} = \boldsymbol{X})$ that HMMs maximize.

The distinction is similar to that between logistic regression (which maximizes the conditional likelihood $\mathbb{P}(Y|X)$) on the one hand and linear discriminant analysis (which maximizes the joint likelihood $\mathbb{P}(Y, X) = \mathbb{P}(Y)\mathbb{P}(X|Y)$) on the other. We have seen that the extra distributional assumptions on the covariates presumed by linear discriminant analysis and the concomitant maximization of the joint likelihood can yield large efficiency gains in appropriate settings. Something similar holds here with HMMs as compared to both MEMMs and CRFs but obviously depends on the those assumptions being correct.

Maximum Entropy Markov Models (McCallum *et al.*, 2000) maximize the conditional likelihood $\mathbb{P}(Y_{1:T}|X_{1:T})$ via learning of $\mathbb{P}(Y_t|Y_{t-1}, X_t)$. Hence, it has different dependency structure than HMMs (compare the first and second panels of

Figure 2.3). In particular, MEMMs model $\mathbb{P}(Y_t|Y_{t-1}, X_t)$ as

$$\mathbb{P}(Y_t|Y_{t-1}, X_{1:T}) = \frac{1}{Z(Y_{t-1}, X_{1:T})} \exp\left[\sum_{j=1}^{\tilde{p}} \beta_j f_j(Y_t, X_{1:T})\right]$$

where $Z(Y_{t-1}, X_{1:T})$ is a normalizing constant which ensures the probabilities sum to one.

Of note and unlike HMMs, MEMMs are not constrained to use only X_t to predict Y_t because they maximize the conditional rather than joint likelihood. In fact, the entire $X_{1:T} = \boldsymbol{X}$ sequence (or, typically, functions extracted from it) can be used to predict Y_t. In particular, the $f_j(Y_t, X_{1:T})$ are functions that can depend on Y_t and the entire covariate sequence $X_{1:T} = \boldsymbol{X}$ and are the "real" covariates used by the MEMM at time t (i.e., they take the place of $X_t = x_t$ in an HMM thus the number of f_j are denoted by $\tilde{p}$ in place of the usual p). This is important because it may be that "long-distance features" are important for predicting Y_t (for instance, in classifying letters which occur within words which occur within sentences, it might matter whether that word starts with a capital letter or whether the sentence ends with a question mark). Finally, the β_j are analogous to regression slopes on the covariates f_j and are found by maximizing the conditional likelihood $\mathbb{P}(Y_{1:T}|X_{1:T}) = \mathbb{P}(Y_1|X_{1:T}) \prod_{t=2}^{T} \mathbb{P}(Y_t|Y_{t-1}, X_{1:T})$.

MEMMs have fallen out of favor because they suffer from something known as the *label bias problem* (Bottou, 1991; Lafferty *et al.*, 2001). The label bias means that any probability that "arrives" at one state Y_t must be "passed on" to all k

possible successors Y_{t+1}. That is, covariates can affect how much probability each $Y_{t+1} = i$ gets, but they cannot affect total probability given across all states, causing a bias towards states with fewer outgoing transitions. Unlike in HMMs, the Viterbi algorithm cannot downweight an entire sequence of states based on covariates.

The label bias problem is perhaps best understood in the context of an example and the classic one involves two input or covariate strings: $X_{1:T} = \boldsymbol{X} = rib$ which is correctly classified by $Y_{1:T} = \boldsymbol{y} = 111$ and $X_{1:T} = \boldsymbol{X} = rob$ which is correctly classified by $Y_{1:T} = \boldsymbol{y} = 222$. After observation of $X_1 = r$, the probability of Y_1 is evenly split amongst the two sequences: $\mathbb{P}(Y_1 = 1|X_1 = r) = \mathbb{P}(Y_1 = 2|X_1 = r) = 1/2$. Now, suppose we observe $X_2 = i$; then we know $X_{1:T} = rib$ and $Y_{1:T} = 111$. However, according to the MEMM, the probability is still evenly split: because the transition from $Y_1 = 1$ to $Y_2 = 2$ has probability zero, the 50% probability of $\mathbb{P}(Y_1 = 1|X_1 = r)$ must be passed on to $\mathbb{P}(Y_2 = 1|X_1 = r, X_2 = i)$ (and likewise for $Y_1 = 2$ and $Y_2 = 2$ respectively). Finally, even after observing the $X_3 = b$, the probabilities for $Y_3 = 1$ and $Y_3 = 2$ remain tied at 50%. Thus, MEMM has essentially ignored the "i".

Conditional Random Fields (Lafferty *et al.*, 2001) were proposed to remedy the label bias problem. CRFs model the relationship between Y_t and Y_{t+1} as a Markov Random Field conditional on the sequence $X_{1:T}$. They represent the

graphical structure of Figure 2.3 as a set of potentials:

$$M_t(Y_{t-1}, Y_t|X_{1:T}) = \exp\left[\sum_{j=1}^{\tilde{p}_1} \beta_j^{(1)} f_j^{(1)}(Y_t, X_{1:T}) + \sum_{j=1}^{\tilde{p}_2} \beta_j^{(2)} f_j^{(2)}(Y_{t-1}, Y_t, X_{1:T})\right]. \tag{2.2.1}$$

As with MEMMs, any function of the sequence $X_{1:T} = \boldsymbol{X}$ can be incorporated as covariates. However, here there are $\tilde{p}_1 + \tilde{p}_2$ covariates where the first $\tilde{p}_1$ provide information about Y_t and $X_{1:T}$ and the last $\tilde{p}_2$ provide information about Y_{t-1}, Y_t, and $X_{1:T}$. This allows for the possibility of "long-distance features". Also, as with MEMMs, the $\beta_j^{(1)}$ and $\beta_j^{(2)}$ are the slopes on the effective covariates $f_j^{(1)}$ and $f_j^{(2)}$.

CRFs compute the conditional probability of $Y_{1:T}$ given $X_{1:T}$ as

$$\mathbb{P}(Y_{1:T}|X_{1:T}) = \frac{\prod_{t=1}^{T+1} M_t(Y_{t-1}, Y_t|X_{1:T})}{\left[\prod_{t=1}^{T+1} M_t(X_{1:T})\right]_{0,k+1}}$$

where, for computational reasons, Y_0 is set to 0 and Y_{T+1} is set to $k+1$ and the denominator is a normalizing constant equal to the $(0,k+1)$ entry of the matrix product of the potential matrices M_t. This formulation, where the entire conditional likelihood $\mathbb{P}(Y_{1:T}|X_{1:T})$ is maximized (as opposed to the MEMM specification which maximizes $\mathbb{P}(Y_{1:T}|X_{1:T})$ as a product of the $\mathbb{P}(Y_t|Y_{t-1}, X_{1:T})$) overcomes the label bias problem though it is computationally intensive.

Because of the label bias problem, we do not consider MEMMs as competitor models in this paper: CRFs have proven to be superior and therefore we use them. In the sequel, we will use two implementations of CRF. The first is MALLET

(McCallum, 2003) and it implements the linear model in Equation 2.2.1. The second is known as TreeCRF (Dietterich *et al.*, 2004) which trains the TreeCRF via the Friedman (2001) gradient tree boosting method.

Chapter 3

Probability Estimation

3.1 Introduction

Our focus in the previous chapters has been on *classification*. A related, more general problem is that of *probability estimation*. In the non-sequential case, this involves estimating the *conditional class probability distribution* $\mathbb{P}(Y = j|X = x)$ for $j = 1, \ldots, k$. For the sequential task, this probability estimation involves either estimating the *marginal conditional class probability distribution* $\mathbb{P}(Y_t = j|X_{1:T} = \boldsymbol{X})$ for $j = 1, \ldots, k$ or the more difficult task of estimating the *joint conditional class probability distribution* $\mathbb{P}(Y_{1:T} = \boldsymbol{y}|X_{1:T} = \boldsymbol{X})$ where $Y_{1:T}$ is the vector-valued random variable $(Y_1, ..., Y_T)^T$.

In this chapter, we will focus the discussion on the non-sequential learning task and in particular on the binary case. We do this because the literature on probability

estimation is quite small. The area is under-explored for several reasons. First, probability estimation is very difficult. We shall see that successful classification simply requires the ability to provide good estimates of one particular quantile (usually the median). On the contrary, successful probability estimation requires good estimates of all arbitrary quantiles. Second, one typically only knows the true conditional class probabilities in simulated data. Hence, it is very difficult to evaluate probability estimates on real world data. While so called proper scoring rules (Savage, 1971) are minimized at the true probabilities when one uses the labels $Y_i \in \{0, 1\}$ in place of them, they can only serve to evaluate models in a comparative sense.

Finally, another reason conditional class probability estimation is under-explored is that it is, in a sense, the Holy Grail of machine learning. Good probability estimates are a panacea: they solve all problems in all contexts. For example, if one has good estimates of the conditional class probabilities, one can easily form a good classifier (e.g., classify using the most likely class). Furthermore, good estimates of the conditional class probabilities allow one to minimize any loss function: $\hat{y} = \text{argmin}_i \sum_j \mathbb{P}(Y = j|X) \cdot c_{i,j}$ where $c_{i,j}$ gives the cost of assigning class i to an observation whose true class is j. In most applications, however, one is usually only concerned with doing well on some particular facet of the problem: one may seek to do well at estimating one of the k classes or identifying the observations with conditional class probability above some threshold q for example. Solving the more

general and difficult problem is therefore not typically required in practice.

Since the literature on this topic is sparse, we entirely omit discussion of the multi-class non-sequential problem and both the binary and multi-class sequential problems. We note, however, that the insights discussed here are relevant for and apply to these more difficult settings.

3.2 Quantile Estimation

Before discussing the general problem of estimating the entire conditional class probability distribution, we first narrow our focus even further to discussion of estimating the boundary or region for a particular quantile. That is, we are interested in identifying X such that $\mathbb{P}(Y = 1|X = x) > q$ where q is the quantile of interest. We will show that this subproblem is quite important in its own right and that methods exist which solve it reasonably well.

3.2.1 Unequal Costs

In the standard binary classification task (i.e., $k = 2$), classifiers are usually judged by misclassification error. Minimizing misclassification error is equivalent to minimizing the loss function which gives equal costs to false positives and false negatives; it is also equivalent to classifying at the $1/2$ quantile of the conditional class probability function $\mathbb{P}(Y = 1|x)$.

For many problems, equal costs are not correct and, since misclassification er-

ror assumes equal costs, it is an inappropriate loss function. For instance, in the classic courtroom setting, sending an innocent man to prison is considered worse than failing to convict a guilty man; likewise, in many medical applications, false negatives are more serious than false positives.

Without loss of generality, we can assume the cost of a false positive and the cost of false negative sum to one and that they equal c and $1 - c$ respectively. If $p(x) = \mathbb{P}(Y = 1|x)$ and $1 - p(x) = \mathbb{P}(Y = 0|x)$ are the conditional probabilities of a positive and negative respectively, then, the risk, or expected loss, of classifying a positive is $(1 - p(x))c$ and the risk of classifying a negative is $p(x)(1 - c)$. In order to minimize risk, we classify as a positive when $(1 - p(x))c < p(x)(1 - c)$ which is equivalent to $c < p(x)$. This shows that binary classification with unequal costs is equivalent to *quantile estimation*, estimating the region $p(x) > q = c$. Most classifiers, which implicitly assume equal costs, are therefore median classifiers since they estimate the region $p(x) > q = 1/2$.

A final point is that, besides arising from unequal misclassification costs, quantile classification can also be formulated as an end in itself. For instance, in an internet marketing campaign, one may only want to serve an ad on a particular website if the probability of a user clicking the ad is greater than some threshold probability q.

3.2.2 Imbalanced Base Rates

The problem of imbalanced base rates occurs when one applies a classifier trained on one dataset with one set of base rate probabilities to a dataset with a different set of base rate probabilities. For example, one might train a classifier on a population with 20% positives but apply it to a population with 50% positives. Below, we show that a change in the base rate is equivalent to changing the quantile at which to threshold the calculations. Hence, imbalanced base rates, quantile classification, and classifying with unequal costs of false positives and negatives are equivalent to one another (Elkan, 2001; Mease *et al.*, 2007). Let

$$\begin{aligned} p(x) &= \mathbb{P}(Y = 1|X = x) \\ \pi &= \mathbb{P}(Y = 1) \\ f_1(x) &= \mathbb{P}(X = x|Y = 1) \\ f_0(x) &= \mathbb{P}(X = x|Y = 0). \end{aligned}$$

By Bayes Theorem

$$p(x) = \frac{f_1(x)\pi}{f_1(x)\pi + f_0(x)(1-\pi)}.$$

Equivalently,

$$\frac{p(x)}{1-p(x)} = \frac{f_1(x)\pi}{f_0(x)(1-\pi)}$$

or

$$\frac{p(x)}{1-p(x)}/\frac{\pi}{1-\pi} = \frac{f_1(x)}{f_0(x)}. \tag{3.2.1}$$

Now, assume there is another population which is the same in all respects except that the base rates π and $1-\pi$ are different. Assume in this new population, the base rates are π^* and $1-\pi^*$. If we let $p^*(x) = \mathbb{P}(Y=1|X=x)$ be the conditional probability that $Y=1$ in this new population, Equation 3.2.1 implies that $p(x)$ and $p^*(x)$ can be related as follows:

$$\frac{p(x)}{1-p(x)}/\frac{\pi}{1-\pi} = \frac{f_1(x)}{f_0(x)} = \frac{p^*(x)}{1-p^*(x)}/\frac{\pi^*}{1-\pi^*}.$$

Hence,

$$\frac{p^*(x)}{1-p^*(x)} = \frac{p(x)}{1-p(x)}\frac{1-\pi}{\pi}\frac{\pi^*}{1-\pi^*}. \tag{3.2.2}$$

Thus, we can obtain a classifier on the new population by adjusting the old one for the new base rates. This has a profound implication: while it is obvious that an algorithm that produces good probability estimates will also produce good class estimates, Equation 3.2.2 suggests an algorithm that produces good class estimates will also produce good probability estimates if the base rate distribution is "tilted" in the proper way (in fact, this is the motivation Jittered Over/Under-Sampling-Boost (JOUS-Boost) technique of Mease *et al.* (2007)). That is, there may be an isomorphism of sorts between the space of good classifiers and the space of good probability estimators.

3.2.3 Machine Learning Approaches to Quantile Estimation

Binary classification with unequal costs or for populations with imbalanced base rates is common in the literature. A classic example of the latter is bankruptcy prediction where there are vast numbers of negatives but very few positives (Foster and Stine, 2004); one must correct for this discrepancy in order to make accurate predictions.

Algorithms such as AdaBoost (Freund and Schapire, 1996), which have proven exceptional at classification at the 1/2 quantile, have been modified to classify with unequal costs. Such modifications include Slipper (Cohen and Singer, 1999), AdaCost (Fan *et al.*, 1999), CSB1 and CSB2 (Ting, 2000), and RareBoost (Joshi *et al.*, 2001). All have shown some improvement over AdaBoost, but no method appears to dominate.

Another approach for dealing with the triply equivalent problem of unequal costs / quantile thresholding / imbalanced base rates that is popular in the computer science literature involves under-sampling and over-sampling (Chan and Stolfo, 1998; Elkan, 2001; Estabrooks *et al.*, 2004). One typically over-samples the rare class with replacement and/or under-samples the dominant class without replacement. Sampling with replacement carries with it the concomitant issue of ties in the sample (i.e., repeated datapoints). Tie-breaking is necessary for certain algorithms which are driven more by the set of unique datapoints than by number of tied ones (Mease *et al.*, 2007). An interesting approach for dealing with this issue is the Syn-

thetic Minority Over-Sampling TEchnique (SMOTE) of Chawla *et al.* (2002, 2003) which avoids ties in over-sampled classes by moving the sampled covariates towards neighbors of the same class.

3.3 Conditional Class Probability Estimation

3.3.1 Introduction

Conditional class probability estimation extends the problem of quantile classification from estimating at a particular quantile q to estimating at all arbitrary quantiles $q \in [0, 1]$. While machine learning methods have been adapted from estimating at $q = 1/2$ in order to estimate at a particular $q \in [0, 1]$, estimating at all quantiles is a significantly greater challenge.

Rather than proceeding from a single q to all q as has been done in the machine learning literature, the general approach in statistics has been to proceed in the opposite direction. First, estimate the entire conditional class probability function. Then, use this function to achieve classification at a particular quantile. For instance, model-based approaches such as logistic regression give an estimate of the conditional class probability function which can easily be transformed into an arbitrary quantile classifier by thresholding the probability function. Such approaches are indeed very successful but under very restrictive conditions: they require knowledge of the functional form of $\mathbb{P}(Y = 1|X = x)$ and also sufficient data for estimating

the parameters accurately.

When the functional form is unknown or data is scarce, conditional class probability estimation is *extremely difficult*. Whereas quantile classifiers only have to be accurate at one particular quantile, probability estimators must be accurate at *every* quantile. That is, probability estimators not only have to perform the task of a good quantile classifier, they must perform the tasks of *all* quantile classifiers and perform them well. In order to classify, both methods typically work by utilizing a score function and thresholding it (usually, arbitrarily at zero). Quantile classifiers, by focusing on one particular quantile, thus only need to be accurate up to the *sign* of the classifier to provide good performance on test sets; a conditional probability estimator, on the contrary, must be accurate at all thresholds and therefore the *absolute value* of the score function is also critical, not just the sign of it. Hence, probability estimators face a much more difficult task.

3.3.2 Machine Learning Methods and Probability Estimation

Many of the machine learning methods discussed in Chapter 2 can be used to form conditional class probability estimates as well as classifications. We briefly review some of the known results pertaining to these.

Not surprisingly, individual CART trees tend to give fairly poor conditional class probability estimates. This is a consequence of the fact that CART trees fit the

same conditional class probabilities for all points that fall within a given terminal node, thus ignoring any heterogeneity among them. This unrealistic property is not shared by the methods which combine trees such as boosting and Random forests and therefore those methods hold greater hope for providing successful probability estimates.

The forward, stage-wise additive view of boosting presented in Chapter 2 suggests that AdaBoost can be transformed into an estimator of the conditional class probability distribution via a link function (Friedman *et al.*, 2000a). It also led to the development of other algorithms like LogitBoost which use the same forward, stage-wise additive optimization but for other loss functions; these other algorithms are therefore also equipped with link functions to obtain conditional class probability estimates (Friedman *et al.*, 2000a).

Logistic regression also uses a link function and is known to provide good probability estimates (and therefore good classifications) when the functional form is known. Since AdaBoost provides good classifications even when the function form is unknown, it was hoped that the link function of Friedman *et al.* (2000a) would transform it into a good probability estimator in such cases.

Unfortunately, it has been shown by several studies that AdaBoost and LogitBoost provide poor estimates of the conditional class probability distribution (Mease *et al.*, 2007; Mease and Wyner, 2008; McShane, 2007). Typically, when AdaBoost has been run for enough iterations to produce good class estimates, its correspond-

ing probability estimates via the link function have diverged to near zero or one. Furthermore, the same is true of LogitBoost despite the fact that estimation of class probabilities via log-likelihood loss provided the motivation for this algorithm.

One of the reasons boosting is so successful at classification is that the "score function" (i.e., the weighted sum of base learners) tends to be very large in absolute value: this leads to overfit probability estimates that diverge to zero or one (and which are therefore quite poor) whereas it does not lead to overfit classifications (because, for classification, only the sign of the score function–not its absolute value–matters). Since providing probability estimates requires being a good quantile classifier for *all* quantiles (i.e., the absolute value of the score function does matter), AdaBoost tends to fail at probability estimation.

The apparent failure of boosting to estimate probabilities and the theoretical view of it as a forward, stagewise additive model have led to a number of refinements of the algorithm. Obviously one such refinement is LogitBoost (Friedman *et al.*, 2000a), but there are also suggestions for early stopping (Dettling and Buhlmann, 2003), shrinkage (Friedman *et al.*, 2000b), regularization methods (Bickel *et al.*, 2006; Jiang, 2004; Lugosi and Vayatis, 2004), and using shallower trees / weaker base learners (Friedman *et al.*, 2000a; Hastie *et al.*, 2001). But, given that boosting overfits probability estimates but *not* median estimates, it is questionable whether boosting's success is due to similarity with logistic regression as suggested in Friedman *et al.* (2000a). Thus, the practical suggestions derived from this view might

be misguided (Mease and Wyner, 2008).

If one wants to retain the forward, stagewise additive logistic regression viewpoint, it thus seems one must temper it by noting that overfit probability estimates may be required to attain optimal classifications. Furthermore, in the presence of unequal misclassification costs (or imbalanced base rates or classification at quantiles different from 1/2), this view may lead to poor performance: one must hope one stops the boosting algorithm early enough such that the score function has not diverged (and therefore produces bad probability estimates) but late enough that the algorithm has sufficiently learned the data structure (and therefore produces good class estimates).

A final point is that bagging, and in particular Random forests, tend to produce much more reasonable and sometimes even quite good probability estimates as is shown by Bostrom (2007) and Bostrom (2008) (particularly when calibrated) and by our own results presented in Chapter 6. It is thought that, since these techniques do not recursively re-weight the individual datapoints but instead rely on the bootstrap, they avoid the overfitting tendency of boosting methods. Much exploration is still needed, however.

3.4 Proper Scoring Rules

We have discussed some of the difficulties involved with quantile classification and probability estimation. In this section, we briefly discuss proper scoring rules (Sav-

age, 1971) which allow us to *evaluate* whether these probability estimates are indeed successful, either on an absolute or a relative basis and, in particular, for real world data. Since there is no single established method for evaluating probability estimates (Zadrozny and Elkan, 2001), it is essential we evaluate our estimates using proper scoring rules.

A proper scoring rule for the binary class problem is one that is *Fisher consistent*, that is one such that the $\text{argmin}_{\hat{p}(x)} \mathbb{E}_{Y \sim Bernoulli(p)} L(Y, \hat{p}(x)) = p$ for all $p \in [0, 1]$ and pointwise at all x. With proper scoring rules, the loss is minimized at $\hat{p} = p$ when one is required to use $I(Y = 1)$ in place of the true p because the latter is unknown (i.e., as is often the case when one uses real as opposed to simulated data).

Several popular examples of proper scoring rules are given by

$$\begin{aligned}
\text{Misclassification Loss:} \quad & \frac{1}{n} \sum_{i=1}^{n} I[p(x_i) > 1/2 \ \& \ \hat{p}(x_i) > 1/2] \\
\text{Squared Error Loss:} \quad & \frac{1}{n} \sum_{i=1}^{n} [p(x_i) - \hat{p}(x_i)]^2 \\
\text{Log Loss:} \quad & -\frac{1}{n} \sum_{i=1}^{n} [p(x_i) \log(\hat{p}(x_i)) - (1 - p(x_i)) \log(1 - \hat{p}(x_i))] \\
\text{Exponential Loss:} \quad & \frac{1}{n} \sum_{i=1}^{n} \left[p(x_i) \sqrt{\frac{1 - \hat{p}(x_i)}{\hat{p}(x_i)}} + (1 - p(x_i)) \sqrt{\frac{\hat{p}(x_i)}{1 - \hat{p}(x_i)}} \right]
\end{aligned}$$

where the $\{x_i\}_{i=1}^{n}$ form a hold-out sample. In all of these cases, one can substitute y_i for $p(x_i)$ and the loss is still minimized at the true $\hat{p}(x_i) = p(x_i)$. This is vital because, except in the case of simulated data, $p(x_i)$ is rarely known whereas y_i often is. Hence, when using a proper scoring rule, we are assured that our loss function

is minimized at the "right" place. Still, we can only show how various methods perform relative to one another using proper scoring rules. We have little sense how methods perform in an absolute sense without knowledge of $p(x)$ so even evaluation of model fits using proper scoring rules is quite limited.

The above loss functions are not the only possible proper scoring rules, however they do cover a range of popular ones (Buja *et al.*, 2005). The first loss function is very popular in computer science whereas the second and third are popular in statistics. The fourth is interesting because it is the proper scoring rule corresponding to the Friedman *et al.* (2000a) AdaBoost link function (Buja *et al.*, 2005). Two counter-examples which are often used but are not in fact proper scoring rules are absolute loss, $\frac{1}{n}\sum_{i=1}^{n}|p(x_i)-\hat{p}(x_i)|$, and power loss for powers α not equal to two, $\frac{1}{n}\sum_{i=1}^{n}[p(x_i)-\hat{p}(x_i)]^{\alpha}$. One should in general avoid such loss functions, except when the $p(x_i)$ are known, if one wants to obtain good estimates of the conditional class probability distribution.

3.5 Conclusion

Conditional class probability estimation is machine learning's Holy Grail: long sought-after, difficult to find, and priceless (because probability estimates can solve all problems). Many methods fail at the task of probability estimation even when they provide good classifications. Boosting methods in particular fall prey to this weakness. A principal difficulty concerns evaluating probability estimates on real

world data, where often only the class labels and not the underlying conditional probabilities are available. Proper scoring rules provide some help here, but it is of a limited nature.

Quantile estimation, a sub-problem of probability estimation, involves finding the region $\mathbb{P}(Y = 1|X = x) > q$ for some $q \in [0, 1]$. We have shown that this problem is equivalent to both classifying with unequal costs of misclassification and to classifying on a population with varying base rates. Various methods can be adapted with great success on this problem and the main strategies typically involve "tilting" the base rate distribution by over-sampling or under-sampling some of the classes. However, it has proven difficult to extend these strategies for classifying at one particular q to all q and it remains an important area for future research.

Chapter 4

Difficulties for Sequential Methods

4.1 Loss Functions and Probability Estimates

There are several difficulties for sequential learning which require mention. First and foremost is the notion of *probability estimation* and *loss function*. We have seen that standard classification methods generally focus on *misclassification loss*. However, in Chapter 2, we saw that by focusing on other loss functions such as exponential loss, binomial negative log likelihood, and squared error loss, classifications can be improved. Furthermore, in Chapter 3, we saw that different costs for false positives and false negatives require arbitrary quantile classification rather than the median classification implied by using misclassification loss on a binary problem.

Issues regarding probability estimation are even more difficult in the sequential case. For some problems, it may be sufficient to estimate the marginal distribu-

tion of the labels conditional on the *entire* covariate sequence, $\mathbb{P}(Y_t|X_{1:T})$ (this is the analogue to estimating the conditional class probability function, $\mathbb{P}(Y|X)$, in the standard case) and we have seen the forward-backwards algorithm can provide estimates of these marginal probabilities conditional on the model. However, the difficulties encountered in accurately estimating class probabilities for the non-sequential binary case are vastly compounded in the sequential multi-class case.

Even more daunting is the fact that some problems might require estimation of the joint distribution of the labels conditional on the *entire* covariate sequence, $\mathbb{P}(Y_{1:T}|X_{1:T})$, a quantity which has no analogue in the non-sequential case. Though this is generally difficult to compute, the Viterbi algorithm can be used to efficiently obtain $\text{argmax}_{\boldsymbol{y}} \mathbb{P}(Y_{1:T} = \boldsymbol{y}|X_{1:T} = \boldsymbol{X})$ conditional on the model.

These issues are important because only with good estimates of the full joint distribution $\mathbb{P}(Y_{1:T}|X_{1:T})$ can we minimize an *arbitrary* loss function. But, not only is obtaining good estimates of this distribution extremely difficult, it is also not even sufficient: equipped with a good estimate of it, we still might not be able to choose the $\boldsymbol{y}$ sequence which minimizes loss because computing the optimal $\boldsymbol{y}$ given the joint distribution (or an estimate of it) may be computationally too taxing.

There are losses, however, which depend only on either the marginal probabilities or classifying the entire sequence in which case the forwards-backwards $\gamma_t(i)$ probabilities or the Viterbi $\boldsymbol{y}^*$ sequence respectively suffice. For an example of the former, consider an arbitrary loss matrix $C = [c_{i,j}]_{i=1,\ldots,k;j=1,\ldots,k}$ where $c_{i,j}$ gives the

cost of assigning class i to an observation whose true class is j (usually $c_{i,i} = 0$). As in the standard case, the solution in the sequential case is to predict at each time t the class with minimum expected cost:

$$\hat{y}_t = \underset{i}{\operatorname{argmin}} \sum_j \mathbb{P}(Y_t = j | X_{1:T} = \boldsymbol{X}) \cdot c_{i,j}$$

In this case, the marginal probabilities suffice. One can also imagine situations where there is 0-1 loss on the entire sequence: either you correctly classify the entire sequence and have no loss or you make one or more mistakes and obtain a loss of one. In this case, the Viterbi sequence is optimal.

However, there are many cases which fall in between these two. For example, consider the detection of a rare class in a sequence and, for simplicity, assume there are only two classes. One could imagine generalizing the arbitrary binary loss matrix C with (normalized) costs c and $1 - c$ assigned to false positives and false negatives respectively for the entire sequence. But, such a loss may not in fact be realistic. Perhaps, if the rare class is detected at time s "close enough" to the true time t, zero or low cost is incurred while missed events (false negatives) incur some other cost and false positives incur a third cost.

Or consider the case of detecting credit card fraud where the goal is to predict the time t when the card was stolen. This is a sequential problem with $Y_1 = Y_2 = ... = Y_{t-1} = 0$ and $Y_t = Y_{t+1} = ... = Y_T = 1$, that is, it is a change-point detection problem. Now, after fitting a sequential classifier, there are many strategies to

estimate t: one can estimate t by some time period s where s might be the first time the $\mathbb{P}(Y_t = 1|X_{1:T}) > z$ for some threshold z_1, the last time $\mathbb{P}(Y_t = 1|X_{1:T}) < z_2$ for some threshold z_2, the first time at which the those probabilities remain above a certain threshold for some number of consecutive periods, or some other estimate. For estimates s where $s < t$, one might incur a loss c_{early} of an early alarm; for estimates s where $s > t$, one might incur a loss c_{late} or even $c_{late} \cdot (s - t)$ for late detection. More complicated loss functions are possible (for instance, ones that estimate how much business was lost due to an early alarm or the total cost of fraudulent purchases on the card after a late alarm or failure to detect). Regardless, the marginal probabilities and the Viterbi path will not suffice.

Finally, consider the example of hyphenation. Given a word, one seeks a sequence of zeroes and ones of the same length of the word where the ones denote letters after which it is permissible to hyphenate. Clearly, false positives are very costly since they cause confusion to readers. False negatives are not nearly as costly since the word will just be moved to the next line and a small amount of space will be wasted (or, if the word is polysyllabic and other hyphenation points are correctly identified, then the word will be hyphenated at a different place). An additional consideration is that hyphens in the middle of a long word are more useful than ones at the beginning or end. Hence $c_{i,j}$ is really $c_{i,j}(t)$ and is thus time-inhomogenous (i.e., depends on t).

For many problems, the marginal probabilities and most likely sequence given

by the forwards-backwards algorithm and Viterbi algorithm will be sufficient or at least useful, *provided they are estimated accurately*. But, for several others, these will not suffice. While, theoretically, a good estimate of $\mathbb{P}(Y_{1:T}|X_{1:T})$ can solve all problems for all loss functions, estimation of this quantity is *extremely difficult*. Furthermore, good estimates *may not suffice* in practice due to the computational difficulties associated with finding the optimal $\boldsymbol{y}$ given $\mathbb{P}(Y_{1:T}|X_{1:T})$ (or estimates thereof) and a loss function.

4.2 Variable Selection and "Long-Distance Features"

In the traditional regression setting, we are often faced with the task of variable selection, that is selecting some subset of the p covariates in order to predict Y. There is a vast literature on this topic of three broad flavors: penalized likelihood, regularization, and Bayesian (Bayes, 1764). Penalized likelihood methods including the Akaike Information Criterion (Akaike, 1974), Mallows' C_p (Mallows, 1973), the Bayesian Information Criterion (Schwarz, 1978), Minimum Description Length (Rissanen, 1978), the Risk Inflation Criterion (Foster and George, 1994), hard-thresholding (Donoho and Johnstone, 1994), and the Empirical Bayes Information Criterion (George and Foster, 2000). These correspond to thresholding the t-statistics in an orthogonal regression at $\sqrt{2}$, $\sqrt{2}$, $\sqrt{\log n}$, $\sqrt{\log n}$, $\sqrt{2\log p}$, $\sqrt{2\log p}$,

and $\sqrt{2 \log q/p}$ respectively (Foster and Stine, 1997) (q is the number of variable selected so far). There are also regularization methods which put penalties on the size of the estimated parameters; such methods include L_2 penalization via ridge regression (Hoerl, 1962), L_1 penalization via the Lasso (Tibshirani, 1996), combined L_1 and L_2 penalization via the elastic net (Zou and Hastie, 2005), L_1 penalization via the Dantzig selector (Candes and Tao, 2007), and many others. Finally, there are also Bayesian methods such Bayes factors, the Deviance Information Criterion (Spiegelhalter *et al.*, 2002), and the mixture model approach of George and McCulloch (1997).

Clearly, variable selection is an important and well-explored problem in the non-sequential literature and it is particularly salient for more model-based approaches. It is even *more important* in the sequential case. This is because, as is evident for sliding window and recurrent sliding window approaches as well as MEMMs and CRFs (and as will be shown for the variant of HMMs we introduce), the covariates used at any time t can depend on the entire sequence of covariates $X_{1:T}$. Hence, there are $p \cdot T$ potential covariates rather than the usual p (plus, as in the non-sequential case, any functions or transformations of those covariates).

In the sequential case, one can in theory pursue strategies which are similar to the non-sequential case. For instance, one can generate large numbers of features from $X_{1:T} = \boldsymbol{X}$ to be used at each time t and then run a forward or backward stepwise procedure, selecting the model at each stage which performs best according to a

criterion such as AIC. Alternatively, one can enter all of the features to the model and use L_1 penalization on the parameters. While both of these are possible, the computational cost involved in fitting sequential models makes them impractical.

Another strategy involves pre-screening covariates. One can compute measures of relevance between a given covariate $X^j_{1:T}$ and the class labels $Y_{1:T}$ and then remove covariates with low relevance. Such measures might include the mutual information of $X^j_{1:T}$ and $Y_{1:T}$ or the classification error of the model when using a given covariate and no others. This is similar to what is done when growing CART trees. Unfortunately, this method ignores interactions between features and therefore can miss important ones.

A final method consists of fitting (or possibly over-fitting) a simple model to the dataset and using that model to identify important covariates before proceeding to a more complex model. In the non-sequential setting, this might involve fitting a large (potentially over-grown or under-pruned) CART tree to the data; then, the subset of the variables selected by the CART tree could be used in more complicated methods such as AdaBoost or stepwise logistic regression. In the sequential setting, similar approaches are possible.

In practice, researchers often apply a fixed window-width, say of size w, and augment the covariates in the manner of the sliding window approach. While this may be fine for many problems, it is unsuitable in other cases and has several important drawbacks. First, it is unlikely that all $w \cdot p$ covariates used at time t

are helpful for predicting Y_t. Second, and perhaps more important, there may be "long-distance features" which are important (e.g., the features $X_{1:T}$ "begins with a number", "ends with a questions mark", and "is shorter than thirty" used in McCallum *et al.* (2000)).

4.3 Computational Complexity

A final consideration, which has come up several times above, is computational complexity and the runtimes of these algorithms. Many algorithms such as CRFs are difficult to fit, thus compounding issues of variable selection. Furthermore, while the forward-backward algorithm and the Viterbi algorithm are very efficient, they can still be slow to apply for complicated model structures. Furthermore, as mentioned above, they do not give a sequence $\boldsymbol{y}$ which minimizes arbitrary loss functions: only when the loss depends on the individual y_t (forward-backward) or requires correct classification of the entire sequence (Viterbi) are they optimal.

Finally, even if one does settle on a model, one fits it, and it provides excellent probability estimates of $\mathbb{P}(Y_{1:T}|X_{1:T})$, one still may be unable to find the sequence $\boldsymbol{y}$ which minimizes the expected loss for arbitrary loss functions. That is, computing the optimal $\boldsymbol{y}$ given the joint distribution (or an estimate of it) may be computationally too taxing.

Chapter 5

PrAGMaTiSt: Prediction and Analysis for Generalized Markov Time Series of States

5.1 Introduction

In Chapter 2, we presented a number of limitations of the standard Markov models and Hidden Markov Models. First, estimation of k multivariate probabilities $\boldsymbol{\mu}$ is a difficult task, particularly when the number of covariates p is large or in the presence of rare classes. Second, long term dependencies are difficult to model in an first order Markov setting because (i) any relationship between Y_t and Y_{t+k} must be "communicated" via $Y_{t+1}, ..., Y_{t+k-1}$ and because (ii) introducing a sliding

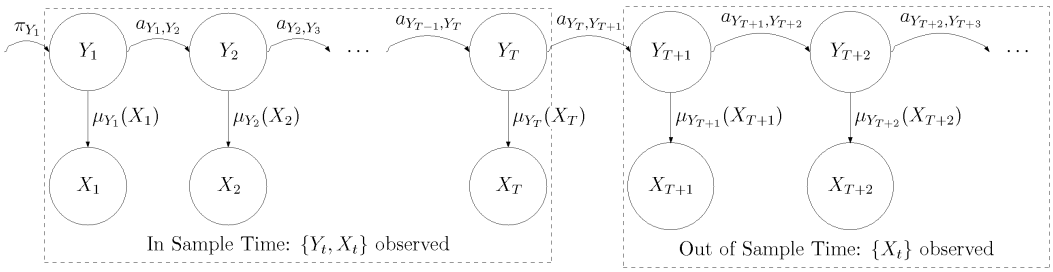

Figure 5.1: A First Order Markov Model. In the first part, we observe both Y_t and X_t. In the second part, we observe X_t and must predict the Y_t.

window of covariates would involve replacing the modeling of the conditional covariate distribution $\mathbb{P}(X_t|Y_t)$ with modeling of the more complicated distribution $\mathbb{P}(X_t|Y_{t-d}, ..., Y_t, ..., Y_{t+d})$.

In this chapter, we attempt to mitigate these problems whilst remaining in the model-based Markov model setting. We solve the first problem, that of estimating $\boldsymbol{\mu}$, by training the model in a *discriminative* fashion. This will also allow us to partially solve the second problem as it will allow the use of a sliding window of covariates. A second approach to solving the second problem is introducing longer-term dependence structures directly into the Y_t themselves.

5.2 Discriminative Markov Models

The structure of a supervised Markov model is given in Figure 5.1. To refresh, $\pi = (\pi_1, ..., \pi_k)^T$ is the vector of initial probabilities, $\mathbb{P}(Y_1 = i)$; $\boldsymbol{A} = [a_{i,j}]_{i=1,...,k;j=1,...,k}$ is the transition probability matrix whose entries are $a_{i,j} = \mathbb{P}(Y_{t+1} = j|Y_t = i)$; and $\boldsymbol{\mu} = (\mu_1(x), ..., \mu_k(x))^T$ is vector of probability measures on the covariate space such

that each μ_i gives the conditional covariate probability $\mu_i(x) = \mathbb{P}(X_t = x|Y_t = i)$. We let $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu})$ denote the collection of these distributions.

It is easy to see that the likelihood for this model is given by

$$\begin{aligned} L(Y_{1:T}, X_{1:T}|\boldsymbol{\Theta}) &= \mathbb{P}(Y_1)\mathbb{P}(X_1|Y_1) \cdot \mathbb{P}(Y_2|Y_1)\mathbb{P}(X_2|Y_2) \cdots \mathbb{P}(Y_T|Y_{T-1})\mathbb{P}(X_T|Y_T) \\ &= \pi_{Y_1}\mu_{Y_1}(X_1) \cdot a_{Y_1,Y_2}\mu_{Y_2}(X_2) \cdots a_{Y_{T-1},Y_T}\mu_{Y_T}(X_T) \\ &= \pi_{Y_1}\left[\prod_{t=2}^{T} a_{Y_{t-1},Y_t}\right]\left[\prod_{t=1}^{T} \mu_{Y_t}(X_t)\right]. \end{aligned}$$

The likelihood factorizes nicely. It is clear that we can estimate the transition distribution, $\mathbb{P}(Y_t|Y_{t-1}) = a_{Y_{t-1},Y_t}$, by the empirical rate of transition from state i to state j, that these are the maximum likelihood estimates, and that they are therefore asymptotically consistent (of course, more sophisticated estimation strategies, for example shrinking to a common distribution, are possible). The MLE for the initialization $\mathbb{P}(Y_1) = \pi_{Y_1}$ would be to place all of the mass on the state which actually did come first but this is clearly a poor estimate and therefore one usually uses the empirical frequency of each of the k classes. It is interesting to note that there are two cases for out of sample prediction. The first case is illustrated in Figure 5.1 and occurs when the out of sample sequence continues from the in sample sequence: in this case, since Y_T is known, the initialization distribution for Y_{T+1} is the appropriate row of transition probability matrix $\boldsymbol{A}$. The second case occurs when the out of sample sequence is an entirely new one in which case an estimate

of $\boldsymbol{\pi}$ is required.

The difficulty in estimating $\boldsymbol{\pi}$ can be resolved if we have multiple training sequences, $\{(\boldsymbol{y}_i, \boldsymbol{X}_i)\}_{i=1}^N$ in which case the likelihood above becomes

$$\begin{aligned} L(\{(Y_{i,1:T}, X_{i,1:T})\}_{i=1}^N|\boldsymbol{\Theta}) &= \prod_{i=1}^N \left[\pi_{Y_{i,1}} \left[\prod_{t=2}^{T_i} a_{Y_{i,t-1},Y_{i,t}}\right] \left[\prod_{t=1}^{T_i} \mu_{Y_{i,t}}(X_{i,t})\right]\right] \\ &= \left[\prod_{i=1}^N \pi_{Y_{i,1}}\right] \left[\prod_{i=1}^N \prod_{t=2}^{T_i} a_{Y_{i,t-1},Y_{i,t}}\right] \left[\prod_{i=1}^N \prod_{t=1}^{T_i} \mu_{Y_{i,t}}(X_{i,t})\right]. \end{aligned}$$

Again, the likelihood factorizes and we now have multiple observations with which to estimate $\boldsymbol{\pi}$. In the sequel, we assume a single training sequence for simplicity though it is conceptually trivial to extend to the case of multiple training sequences as above.

The factorization of the likelihood shown above demonstrates that $\boldsymbol{\mu}$ can be estimated by estimating each of its component parts $\mu_1, ..., \mu_k$ individually conditional on the set of (Y_t, X_t) for which $Y_t = i$. As noted, a principal difficulty for Markov models is the estimation of several multivariate distributions conditional on categorical Y_t. However, the inverse problem of classifying categorical Y_t based on a high-dimensional X_t is the ideal situation for the classification methods we encountered in Chapter 2. We can adapt these classification methods to address

our model estimation by using Bayes theorem

$$\begin{aligned}\mu_i(X_t) &= \mathbb{P}(X_t = x_t | Y_t = i) \\ &= \frac{\overbrace{\mathbb{P}(Y_t = i \,|\, X_t = x_t)}^{Classification\ Methods} \cdot \overbrace{\mathbb{P}(X_t = x_t)}^{Constant}}{\underbrace{\mathbb{P}(Y_t = i)}_{Marginal\ Probabilities}}.\end{aligned}$$

With this in mind, we see that we can rewrite the likelihood as

$$\begin{aligned}
L(Y_{1:T}, X_{1:T}|\boldsymbol{\Theta}) &= \mathbb{P}(Y_1)\mathbb{P}(X_1|Y_1) \cdot \mathbb{P}(Y_2|Y_1)\mathbb{P}(X_2|Y_2) \cdots \mathbb{P}(Y_T|Y_{T-1})\mathbb{P}(X_T|Y_T) \\
&= \mathbb{P}(Y_1) \left[\prod_{t=2}^{T} \mathbb{P}(Y_t|Y_{t-1})\right] \left[\prod_{t=1}^{T} \mathbb{P}(X_t|Y_t)\right] \\
&= \mathbb{P}(Y_1) \left[\prod_{t=2}^{T} \mathbb{P}(Y_t|Y_{t-1})\right] \left[\prod_{t=1}^{T} \frac{\mathbb{P}(Y_t|X_t) \cdot \mathbb{P}(X_t)}{\mathbb{P}(Y_t)}\right] \\
&= \pi_{Y_1} \left[\prod_{t=2}^{T} a_{Y_{t-1},Y_t}\right] \left[\prod_{t=1}^{T} \frac{f_{Y_t}(X_t) \cdot \mathbb{P}(X_t)}{p_{Y_t}}\right] \\
&\propto \pi_{Y_1} \left[\prod_{t=2}^{T} a_{Y_{t-1},Y_t}\right] \left[\prod_{t=1}^{T} \frac{1}{p_{Y_t}}\right] \left[\prod_{t=1}^{T} f_{Y_t}(X_t)\right] \\
&= L(Y_{1:T}, X_{1:T}|\tilde{\boldsymbol{\Theta}})
\end{aligned}$$

where $f_i(X_t) = \mathbb{P}(Y_t = i|X_t)$ is the conditional class probability distribution and $p_i = \mathbb{P}(Y_t = i)$ are the marginal probabilities of the Y_t. Hence, we have transformed the difficult problem of estimating k p-variate probability distributions $\boldsymbol{\mu}$ into the easier problem of estimating a k-dimensional probability vector $\boldsymbol{f} = (f_1(x), ..., f_k(x))^T$ and the marginal probabilities $\boldsymbol{p} = (p_1, ..., p_k)^T$. There are clearly many ways to estimate $\boldsymbol{f}$. We typically estimate $\boldsymbol{p}$ by either (i) the empirical fre-

quency in each of the k states or (ii) the stationary distribution corresponding to the estimate of $\boldsymbol{A}$.

Thus, rather than training the Markov model *generatively*, we are training it *discriminatively* (Smyth, 1994). Consequently, we substitute estimation of $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu})$ for estimation of $\tilde{\boldsymbol{\Theta}} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{f}, \boldsymbol{p})$.

There are two principal advantages of training the Markov model in a discriminative fashion. First, as mentioned above, it is easier to estimate a k-dimensional probability vector than k p-variate probability measures. Second, it is now easy to accommodate "long distance features": sliding windows as well as any other feature of the sequence $X_{1:T} = \boldsymbol{X}$ can easily be incorporated into $X_t = x_t$ when the model is trained in a discriminative fashion. A further advantage is that, under certain assumptions, we can show our estimates are maximum likelihood estimates. While it is possible for estimates of the generative Markov models to be MLEs, this requires restrictive (and likely untestable) assumptions on the μ_i. However, for the discriminative version, we need only make assumptions about the f_i. For instance, if we assume they are linear functions of the covariates as in multinomial logistic regression, our likelihood becomes

$$\begin{aligned} L(Y_{1:T}, X_{1:T}|\tilde{\boldsymbol{\Theta}}) &\propto \pi_{Y_1}\left[\prod_{t=2}^{T} a_{Y_{t-1},Y_t}\right]\left[\prod_{t=1}^{T}\frac{\exp(X_t\cdot\beta_{Y_t})/\sum_{j=1}^{k}\exp(X_t\cdot\beta_j)}{p_{Y_t}}\right] \\ &= \pi_{Y_1}\left[\prod_{t=2}^{T} a_{Y_{t-1},Y_t}\right]\left[\prod_{t=1}^{T}\frac{1}{p_{Y_t}}\right]\left[\prod_{t=1}^{T}\frac{\exp(X_t\cdot\beta_{Y_t})}{\sum_{j=1}^{k}\exp(X_t\cdot\beta_j)}\right] \end{aligned}$$

Algorithm 5.1 The Discriminative Forward Algorithm.

Begin with estimates $\widehat{\tilde{\boldsymbol{\Theta}}} = (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{f}}, \hat{\boldsymbol{p}})$ where $\hat{\boldsymbol{\pi}} = \hat{\boldsymbol{p}}$ or $\hat{\boldsymbol{\pi}}$ is the proper row of $\hat{\boldsymbol{A}}$.
Define $\alpha_t(i) = \mathbb{P}(X_1 = x_1, ..., X_t = x_t, Y_t = i | \widehat{\tilde{\boldsymbol{\Theta}}})$.

1. Initialization: $\alpha_1(i) = \hat{\pi}_i \frac{\hat{f}_i(x_1)}{\hat{p}_i},\ i = 1, ..., k.$
2. Induction: $\alpha_{t+1}(j) = \left[\sum_{i=1}^{k} \alpha_t(i)\hat{a}_{i,j}\right] \frac{\hat{f}_j(x_{t+1})}{\hat{p}_j},\ t = 1, ..., T-1;\ i = 1, ..., k.$
3. Termination: $\mathbb{P}(X_{1:T} = \boldsymbol{X} | \widehat{\tilde{\boldsymbol{\Theta}}}) = \sum_{i=1}^{k} \alpha_T(i).$

Algorithm 5.2 The Discriminative Backward Algorithm.

Begin with estimates $\widehat{\tilde{\boldsymbol{\Theta}}} = (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{f}}, \hat{\boldsymbol{p}})$ where $\hat{\boldsymbol{\pi}} = \hat{\boldsymbol{p}}$ or $\hat{\boldsymbol{\pi}}$ is the proper row of $\hat{\boldsymbol{A}}$.
Define $\beta_t(i) = \mathbb{P}(X_{t+1} = x_{t+1}, ..., X_t = x_T | Y_t = i, \widehat{\tilde{\boldsymbol{\Theta}}})$.

1. Initialization: $\beta_T(i) = 1,\ i = 1, ..., k.$
2. Induction: $\beta_t(i) = \sum_{j=1}^{k} \hat{a}_{i,j}\beta_{t+1}(j)\frac{\hat{f}_j(x_{t+1})}{\hat{p}_j},\ t = T-1, ..., 1;\ i = 1, ..., k.$

where, as usual, $\beta_k = 0$ by definition. Due to the way the likelihood factors, it is clear our estimation strategy will maximize the three bracketed terms (and, as above, it is not desirable to maximize the first term π_{Y_1}.). In general, any procedure which provides an MLE for the f_i will also provide an MLE for $\tilde{\boldsymbol{\Theta}}$ when used in conjunction with the strategy outlined above. Hence, since all of our parameters can be estimated as MLEs, all the guarantees that apply to MLEs apply to our model estimates.

Now that we have shown how to transform the generative Markov modeling problem into a discriminative one and how to estimate parameters in an MLE manner, we must modify the forward-backward and Viterbi algorithms to apply to our estimates of $\widehat{\tilde{\boldsymbol{\Theta}}} = (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{f}}, \hat{\boldsymbol{p}})$ rather than estimates of $\boldsymbol{\Theta}$. These modified algorithms are given in Algorithms 5.1, 5.2, and 5.3.

As before, we can use the forward-backward algorithm to find the marginal

Algorithm 5.3 The Discriminative Viterbi Algorithm.

Begin with estimates $\widehat{\widehat{\boldsymbol{\Theta}}} = (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{f}}, \hat{\boldsymbol{p}})$ where $\hat{\boldsymbol{\pi}} = \hat{\boldsymbol{p}}$ or $\hat{\boldsymbol{\pi}}$ is the proper row of $\hat{\boldsymbol{A}}$. Define $\delta_t(i) = \max_{y_1,...,y_{t-1}} \mathbb{P}(Y_1 = y_1, Y_2 = y_2, ..., Y_t = i, X_1 = x_1, X_2 = x_2, ..., X_t = x_t | \widehat{\widehat{\boldsymbol{\Theta}}})$.

1. Initialization:

$$\begin{aligned} \delta_1(i) &= \hat{\pi}_i \frac{\hat{f}_i(x_1)}{\hat{p}_i} \quad i = 1, ..., k \\ \psi_1(i) &= 0 \quad i = 1, ..., k. \end{aligned}$$

2. Recursion:

$$\begin{aligned} \delta_t(j) &= \max_{i=1,...,k}[\delta_{t-1}(i)\hat{a}_{i,j}] \frac{\hat{f}_j(x_t)}{\hat{p}_j} \quad t = 2, ..., T;\ j = 1, ..., k \\ \psi_t(j) &= \text{argmax}_{i=1,...,k}[\delta_{t-1}(i)\hat{a}_{i,j}] \quad t = 2, ..., T;\ j = 1, ..., k. \end{aligned}$$

3. Termination:

$$\begin{aligned} P^* &= \max_{i=1,...,k} \delta_T(i) \\ y_T^* &= \text{argmax}_{i=1,...,k} \delta_T(i). \end{aligned}$$

4. Path (state sequence) backtracking: $y_t^* = \psi_{t+1}(y_{t+1}^*),\ t = T-1, ..., 1$.

conditional probability $\gamma_t(i) = \mathbb{P}(Y_t = i | \widehat{\widehat{\boldsymbol{\Theta}}})$:

$$\hat{\gamma}_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\mathbb{P}(X_{1:T} = \boldsymbol{X} | \widehat{\widehat{\boldsymbol{\Theta}}})} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^k \alpha_t(j)\beta_t(j)}.$$

Also before, our best estimate of Y_t is $\hat{y}_t = \text{argmax}_{i=1,...,k} \hat{\gamma}_t(i)$.

Training a Markov model in a discriminative fashion provides many benefits such as (i) the ability to include long-distance, sliding window covariates in X_t, (ii) avoiding the estimation of k p-variate probability measures, (iii) the ability to convert any standard classification methodology into a sequential one, and (iv) the vast computation savings associated with doing so (as compared to using methods such as CRFs). However, there is still one major problem: any relationship between Y_t and Y_{t+k} still must be communicated via $Y_{t+1}, Y_{t+2}, ..., Y_{t+k-1}$ (except insofar as

they are predictable from long-distance and sliding window covariates introduced into X_t). While this disadvantage is common to MEMMs and CRFs as well as Markov models, it is one we would like to avoid. Methods for extending the discriminative Markov approach to accommodate this feature are introduced in the next section.

However, before proceeding, we note the following. Whether we use the generative parameters $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu})$ or the discriminative parameters $\tilde{\boldsymbol{\Theta}} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{f}, \boldsymbol{p})$, there are really only two parameters to estimate: $\boldsymbol{A}$ and either $\boldsymbol{\mu}$ or $\boldsymbol{f}$. It should be clear that $\boldsymbol{\pi}$ is not really an additional parameter which needs to be estimated because, if the out of sample sequence continues from the in sample sequence as in Figure 5.1, then the estimate of $\boldsymbol{\pi}$ is the row of the estimate of $\boldsymbol{A}$ corresponding to Y_T; if it does not, then the estimate of $\boldsymbol{\pi}$ is the estimate of $\boldsymbol{p}$. Furthermore, especially for the generalizations of the model considered below, $\boldsymbol{p}$ is typically estimated as the stationary distribution of $\boldsymbol{A}$. Thus, while there are many parameters, the task may not be as daunting as it first seems.

5.3 Generalized Time Series Structures

In theory, it is straightforward to incorporate higher order dependencies among the Y_t into a probability model. The difficulty is in computation. In Chapter 2, we saw that the forward-backward algorithm gave us a relatively efficient recipe for computing $\gamma_t(i) = \mathbb{P}(Y_t = i | X_{1:T} = \boldsymbol{X}, \boldsymbol{\Theta})$ and the Viterbi algorithm for a

most likely sample path $\boldsymbol{y}^*$. These algorithms, however, are only applicable to first order Markov chains. Hence, it might seem that, for computational reasons, we are limited to forcing dependencies between Y_t and Y_{t+k} to be communicated via $Y_{t+1}, Y_{t+2}, ..., Y_{t+k-1}$.

It turns out, however, that the first order Markov structure can be made very rich by considering various augmentations and transformations of the state space $S = \{1, ..., k\}$. Hence, we can incorporate very general patterns of dependence among the Y_t into our model *and* retain use of the efficient forward-backward and Viterbi algorithms by embedding these more complex structures into a first order Markov chain structure. Below, we will consider several generalizations.

5.3.1 Higher Order Markov Models

We first generalize from a first order Markov chain to an m^{th} order Markov chain. In this case, our initialization probability distribution $\boldsymbol{\pi}$ and our covariate emission distributions $\boldsymbol{\mu}$ (or, in the discriminative case, the conditional class probability function $\boldsymbol{f}$ and the marginal probabilities $\boldsymbol{p}$) remain conceptually the same. The only thing that is different is the transition probability matrix, $\boldsymbol{A}$.

For m^{th} order Markov chains, it is no longer the case that $\mathbb{P}(Y_t|Y_{t-1}, ..., Y_1) = \mathbb{P}(Y_t|Y_{t-1})$ (and therefore that the distribution can be represented as a k x k transition probability matrix). Rather, we have $\mathbb{P}(Y_t|Y_{t-1}, ..., Y_1) = \mathbb{P}(Y_t|Y_{t-1}, ..., Y_{t-m})$ for $t > m$.

If we wish to use forward-backward and Viterbi algorithms, we must convert the m^{th} order Markov chain into a first order Markov chain. Fortunately, this is a relatively straightforward affair. We move to an augmented state space $S' = \prod_{i=1}^{m} S$ (i.e., the Cartesian product of the state space S m times with itself) with $Y'_t = (Y_{t-m}, ..., Y_t)$. Our transition probability distribution $\mathbb{P}(Y_t|Y_{t-1}, ..., Y_{t-m})$, $t > m$ can now be represented by a k^m x k^m matrix, $\boldsymbol{A'}$ (which, as we will see, is relatively sparse) and initialization distribution $\boldsymbol{\pi'} = \mathbb{P}(Y_1, ..., Y_m) = \mathbb{P}(Y'_m)$ by a vector of length k^m. Finally, we now have k^m covariate emission distributions $\boldsymbol{\mu'}$ corresponding to each class in the augmented state space (or, in the discriminative case, we have a conditional class probability function $\boldsymbol{f'}$ which returns a vector of length k^m and the marginal probability vector $\boldsymbol{p'}$ also of length k^m).

The likelihood is now given by

$$\begin{aligned} L(Y_{1:T}, X_{1:T}|\boldsymbol{\Theta'}) &= \mathbb{P}(Y_1, ..., Y_m, X_1, ..., X_m) \cdot \\ &\quad \left[\prod_{t=m+1}^{T} \mathbb{P}(Y_t|Y_{t-1}, ..., Y_{t-m})\right]\left[\prod_{t=m+1}^{T} \mathbb{P}(X_t|Y_t, ..., Y_{t-m+1})\right] \\ &= \mathbb{P}(Y_{1:m}, X_{1:m})\left[\prod_{t=m+1}^{T} \mathbb{P}(Y'_t|Y'_{t-1})\right]\left[\prod_{t=m+1}^{T} \mathbb{P}(X_t|Y'_t)\right] \\ &= \mathbb{P}(Y_{1:m}, X_{1:m})\left[\prod_{t=m+1}^{T} a'_{Y'_t|Y'_{t-1}}\right]\left[\prod_{t=m+1}^{T} \mu'_{Y'_t}(X_t)\right]. \end{aligned}$$

where $\mathbb{P}(Y_{1:m}, X_{1:m})$ is the asymptotically-irrelevant joint distribution of $(Y_{1:m}, X_{1:m})$, the first m states and covariate vectors. Typically, in practice, we simply dis-

card the first m observations for which we cannot create the concatenated value $Y'_t = (Y_{t-m}, ..., Y_t)$.

We can also easily write this in the discriminative fashion as

$$\begin{aligned}
L(Y_{1:T}, X_{1:T}|\boldsymbol{\Theta}') &= \mathbb{P}(Y_{1:m}, X_{1:m}) \left[\prod_{t=m+1}^{T} \mathbb{P}(Y'_t|Y'_{t-1})\right] \left[\prod_{t=m+1}^{T} \mathbb{P}(X_t|Y'_t)\right] \\
&= \mathbb{P}(Y_{1:m}, X_{1:m}) \left[\prod_{t=m+1}^{T} \mathbb{P}(Y'_t|Y'_{t-1})\right] \left[\prod_{t=1}^{T} \frac{\mathbb{P}(Y'_t|X_t) \cdot \mathbb{P}(X_t)}{\mathbb{P}(Y'_t)}\right] \\
&\propto \mathbb{P}(Y_{1:m}, X_{1:m}) \cdot \\
&\quad \left[\prod_{t=m+1}^{T} \mathbb{P}(Y'_t|Y'_{t-1})\right] \cdot \left[\prod_{t=m+1}^{T} \frac{1}{\mathbb{P}(Y'_t)}\right] \left[\prod_{t=m+1}^{T} \mathbb{P}(Y'_t|X_t)\right] \\
&= \mathbb{P}(Y_{1:m}, X_{1:m}) \cdot \\
&\quad \left[\prod_{t=m+1}^{T} a'_{Y'_t|Y'_{t-1}}\right] \cdot \left[\prod_{t=m+1}^{T} \frac{1}{p'_{Y'_t}}\right] \left[\prod_{t=m+1}^{T} f'_{Y'_t}(X_t)\right] \\
&= L(Y_{1:T}, X_{1:T}|\tilde{\boldsymbol{\Theta}}').
\end{aligned}$$

In both the generative and discriminative cases, just as our estimates in the first order Markov model were MLEs, so too are our estimates for the m^{th} order Markov model since we have effectively transformed or embedded the m^{th} order Markov model into a first order Markov model.

As in the first order case, we can estimate $\boldsymbol{A}'$ by the empirical frequency of transitions amongst the Y'_t. Similarly, $\boldsymbol{p}'$ can be estimated by the stationary distribution of $\boldsymbol{A}'$ and $\boldsymbol{\pi}'$ by either the appropriate row of $\boldsymbol{A}'$ or by $\boldsymbol{p}'$ as appropriate. Finally, as in the first order case, we can pursue whatever strategy we like to estimate $\boldsymbol{\mu}'$ or $\boldsymbol{f}'$.

To see how this structure embeds the m^{th} order Markov model into a first order

Markov model, we give an example. Let $m = 2$ and let the original state space be given by $S = \{a, b, c\}$. Then, the augmented state space has nine states and is given by $S' = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$. We now have a nine by nine transition probability matrix. However, many of these entries are zeroes because certain state transitions are impossible. For instance, if we are in the state ba, we can only transition to aa, ab, ac due to the fact that $y'_t = (y_{t-1}, y_t)$. Hence, there are substantial restrictions on the transition distribution.

In a sense, this is all a device: the "real" random variables are still the Y_t (rather than the Y'_t) and the "real" state space is S (rather than S'). Generalizing to an m^{th} Markov model was simply a way of directly inducing higher order dependence into the Y_t series, a dependence we do not necessarily want to force on the covariates. Realization of this fact yields some further restrictions on the model parameters which help with estimation. Though they are not strictly required, they make sense in practice.

The further restrictions are put directly on the $\mu'_{Y'_t}$ in the generative case or the $f'_{Y'_t}(X_t) = \mathbb{P}(Y'_t|X_t)$. For instance, typically we would set $\mu'_{Y'_t} = \mu_{Y_t}$ (i.e., estimate the conditional covariate distribution as in the first order case). For example, using the toy model above, if $Y'_t = ba$ then $Y_t = a$. Rather than assuming a separate covariate emission distribution for ba and, say ca, we simply assume a covariate emission distribution for a (as well as the other two states in S).

For the discriminative case, we typically train the classifier on the random

variable Y_t rather than Y_t' to get $f_{Y_t}(X_t) = \mathbb{P}(Y_t|X_t)$ and then set $f'_{Y_t'}(X_t) = f_{Y_t}(X_t)\frac{\mathbb{P}(Y_t')}{\sum_{Y_t'|Y_t=i}\mathbb{P}(Y_t')}$ where the Y_t corresponding to Y_t' equals i. That is, we must normalize the probability estimates from our classifier trained on S to reflect the marginal probabilities of S'. For instance, using our example above, if $Y_t' = ba$ then $Y_t = a$ and we would set the probability of ba equal to the k-class classifier's probability of a normalized by the ratio of (i) the probability of $Y_t' = ba$ and (ii) the sum of the probabilities of all Y_t' such that $Y_t = a$ (in this case, the sum over the probabilities of aa, ba, and ca).

Using this restriction, the model follows exactly the same structure as Figure 5.1 with one exception: the transition probabilities are a function of the $m+1$ random variables $(Y_{t-m}, ..., Y_t)$ in the m^{th} order case rather than the two random variables (Y_{t-1}, Y_t) as in the first order case.

It is worth noting that these restrictions can be thought of as either (i) restrictions on the model itself or as (ii) restrictions on our estimates. We may be willing to assume the former when the covariates really do seem to be "emitted" based on the contemporaneous state of the system. For instance, perhaps a mouse in NREM will give off the same velocity no matter whether he was previously in REM, NREM, or WAKE. The second case might apply when we feel we simply do not have enough data to estimate the parameters for a k^m class classifier and require a natural way to reduce the size of the parameter space given the fact that there are only k "true" classes.

In the sequel, we will always make this simplifying assumption, typically because we believe the restrictions apply to the model itself. However, it is really a simplification without loss of generality because the unrestricted model really is a first order Markov model on the state space S'. Hence, everything we covered in the previous section holds. Furthermore, we will simply work with the discriminative case going forward. This again is without loss of generality since the two approaches are equivalent via Bayes' Theorem and it is always easy to recover the generative version by swapping the likelihoods as shown in this section and the previous. Furthermore, the code that estimates these models (and subsequent ones) also makes this assumption: the user must only supply an estimate of the augmented transition probability matrix and the k-classifier conditional class probability estimates (and, optionally, the last m states if the test data continues in sequence from the training data so the code knows to set the initialization distribution to equal to the appropriate row of the transition probability matrix; the marginal probabilities are calculated from the stationary distribution of the transition probability matrix).

Though the m^{th} order Markov approach is rather elegant, it has a principal and debilitating difficulty: for m large, it is difficult to estimate $\boldsymbol{A'}$ even with the sparsity required by the fact that $S' = \prod_{i=1}^{m} S$. Therefore, it is also difficult to estimate $\boldsymbol{p'}$, the stationary distribution of $\boldsymbol{A'}$. Consequently, in the (restricted) discriminative case (the case with which we are most concerned), even if we have good estimates of $f_{Y_t}(X_t)$, our estimates of $f'_{Y'_t}(X_t)$ may suffer quite substantially

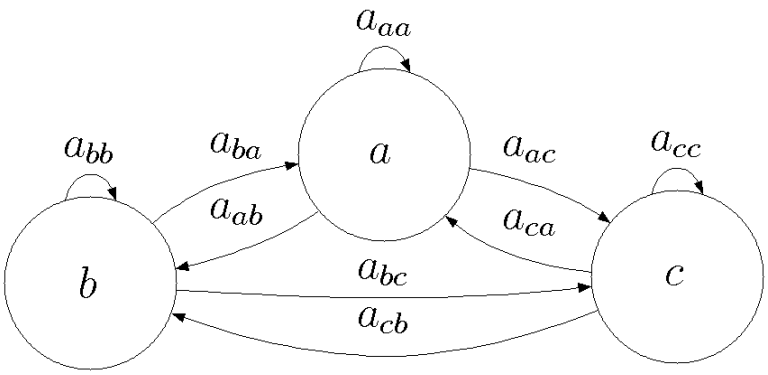

Figure 5.2: The transition diagram corresponding to a first order Markov chain with state space $S = \{a, b, c\}$ and transition probabilities given by a_{ij}.

due to their dependence on $\mathbb{P}(Y_t'|Y_t = i)$ (which clearly depends on $\boldsymbol{p}' = \mathbb{P}(Y_t')$). Hence, it is usually only feasible with small m and therefore is of limited use in inducing long-term dependence structures directly into the Y_t. In the next section, we consider methods which allow for very long-term dependence structures in the Y_t that do not require the estimation of as many parameters as the Markov chains considered here when m is large.

5.3.2 Generalized Markov Models

In Figure 5.2 we present the state space transition diagram for a first order Markov chain. There are three states $S = \{a, b, c\}$ and therefore a 3 x 3 transition probability matrix given by the a_{ij} in the diagram. Such a model implies the holding times in each state i are geometrically distributed with parameter $(1 - a_{ii})$. That is, the probability of staying in state i for τ epochs is $\mathbb{P}_i(\tau) = a_{ii}^{\tau-1}(1 - a_{ii})$.

In many real data applications, the geometric distribution is not plausible. For

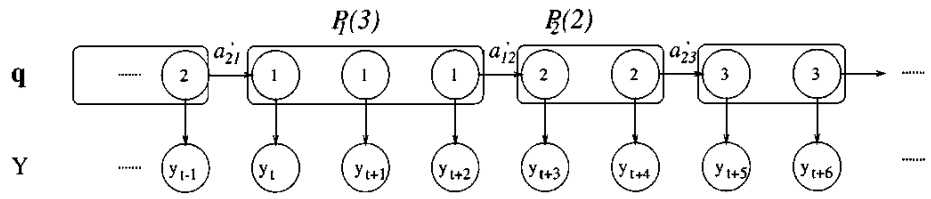

Figure 5.3: A Generalized Markov Model with states given by $\mathbf{q} \in \{1, 2, 3\}$ and covariates y_t. The amount of time spent in state i depends on the duration distribution $P_i(\tau)$. Source: Djuric and Chun (2002).

Algorithm 5.4 Generating Data From a Generalized Markov Model

1. Draw Y_1 from the initialization distribution, $\mathbb{P}(Y_1)$.
2. Draw duration τ_1 from duration distribution, $\mathbb{P}_{Y_1}(\tau)$ and set $Y_2, ..., Y_{\tau_1}$ equal to Y_1.
3. Draw $X_1, ..., X_{\tau_1}$ from the covariate emission distribution, $\mathbb{P}(X_1, ..., X_{\tau_1}|Y_1)$ which we assume is equal to $\prod_{t=1}^{\tau_1} \mathbb{P}(X_t|Y_t)$.
4. Transition to state Y_{τ_1+1} according to transition probability distribution $\mathbb{P}(Y_{t+1}|Y_t)$ which has zero probability of a self-transition.
5. Repeat steps 2-4 *mutatis mutandis* until time T is reached.

instance, in credit card fraud detection, where the Y_t are equal to zero before the card is stolen and one afterwards, the geometric distribution is clearly inappropriate. In sleep, the memoryless property of the geometric distribution also makes it an absurd choice at least in theory.

Thus, one might ask whether one can build more general holding time distributions into the model and whether this can be embedded into the first order Markov framework so that the forward-backward and Viterbi algorithms can be used. Such a model is called a **Generalized Markov Model** (GMM), and it is visualized in Figure 5.3. As we will show shortly, the answer to both questions is yes.

The GMM is parameterized by $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\delta})$ in the generative case or $\tilde{\boldsymbol{\Theta}} =$

$(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{f}, \boldsymbol{p}, \boldsymbol{\delta})$ in the discriminative case. $\boldsymbol{\pi}$, $\boldsymbol{A}$, $\boldsymbol{\mu}$, $\boldsymbol{f}$, and $\boldsymbol{p}$ are exactly as before, now with one exception: $\boldsymbol{A}$ must have zeroes on its diagonal (i.e., no self-transitions because self-transitions are governed by $\boldsymbol{\delta}$ in this model). The new parameter is $\boldsymbol{\delta} = (\delta_1(\tau), ..., \delta_k(\tau))^T$, a vector of probability distributions ("duration distributions") where each element $\delta_i(\tau) = \mathbb{P}_i(\tau)$ gives the probability of remaining in state i for length $\tau = 1, 2, ..., M_i < \infty$ where M_i is the maximal consecutive time that can be spent in state i (we let $M = max_{i=1,...k} M_i$). Data can be generated from this model according to Algorithm 5.4.

Before proceeding to the likelihood and estimation strategies, we discuss how to embed this model into a first order Markov structure. For any sequence $\{Y_t\}_{t=-\infty}^{\infty}$ let us assume we observe $\{Y_t\}_{t=1}^{T}$ and define the sequence of variables $\{Z_t\}_{t=1}^{T}$ as follows. First, we assume that Y_1 is the first episode of its state i and Y_T is the last episode of its state j (i.e., we assume $Y_0 \neq Y_1$ and $Y_T \neq Y_{T+1}$). Then, let $Z_t = \text{argmax}_\tau \{Y_t = Y_{t+1} = ... = Y_{t+\tau-1} \neq Y_{t+\tau}\}$. That is, Z_t gives how much longer the sequence remains in the current state. Finally, we let $Y_t' = (Y_t, Z_t)$. Hence, we have moved from state space $S = \{1, ..., k\}$ to

$$\begin{aligned} S' \;=\; & \{(1,1), (1,2), ..., (1, M_1), \\ & (2,1), (2,2), ..., (2, M_2), \\ & ..., \\ & (k,1), (k,2), ..., (k, M_k)\} \end{aligned}$$

which we sometimes for simplicity define as

$$\begin{aligned} S' &= \{(1,1),(1,2),...,(1,M), \\ &\quad (2,1),(2,2),...,(2,M), \\ &\quad ..., \\ &\quad (k,1),(k,2),...,(k,M)\}. \end{aligned}$$

As an example, the sequence $a, a, b, b, b, b, c, c, a$ would be transformed into

$$(a,2),(a,1),(b,4),(b,3),(b,2),(b,1),(c,2),(c,1),(a,1).$$

Now, we can form the augmented transition probability matrix $\boldsymbol{A}'$ as follows. Each transition from (i,τ) to $(i,\tau-1)$ has probability one for $\tau > 1$ and all other transitions have probability zero. For $\tau = 1$ (that is, $Y'_t = (i,1)$) there are two cases: (i) transitions to (i,n) have probability zero for all $n = 1,...,M_i$ and (ii) transitions to (j,n) have probability $a_{i,j}\delta_j(n)$ for $i \neq j$ and $n = 1,...,M_j$. This is perhaps better explained via an example which we give in Figure 5.4.

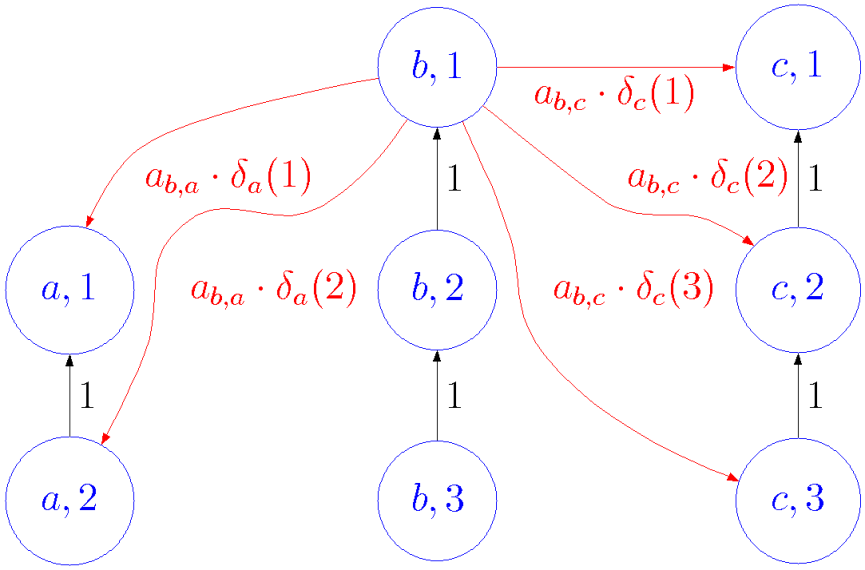

Figure 5.4: The transition diagram corresponding to a generalized Markov chain with state space $S = \{a, b, c\}$, maximal durations $M_a = 2$ and $M_b = M_c = 2$, transition probabilities given by $\boldsymbol{A}$ whose entries are a_{ij}, and duration distributions given by $\boldsymbol{\delta} = (\delta_a, \delta_b, \delta_c)^T$. The augmented state space is $S' = \{(a,2),(a,1),(b,3),(b,2),(b,1),(c,3),(c,2),(c,1)\}$ and is indicated in blue. The augmented transition probabilities $\boldsymbol{A}'$ which are equal to one are given in black. Those that are equal to the products of the $a_{i,j}$ and $\delta_i(\tau)$ and which originate from $(b,1)$ are given in red. Similar transitions from $(a,1)$ to each (b,n) and (c,n) and from $(c,1)$ to each (a,n) and (b,n) are omitted for aesthetic reasons but are formed in an way analogous to that shown in the picture and described in the main text.

Now, similar to before, the likelihood is given by

$$\begin{aligned}
L(Y_{1:T}, X_{1:T}|\tilde{\boldsymbol{\Theta}}') &= \mathbb{P}(Y_1)\mathbb{P}_{Y_1}(Z_1) \cdot \\
&\quad \left[\prod_{t|Y_t \neq Y_{t-1}} \mathbb{P}(Y_t|Y_{t-1})\mathbb{P}_{Y_t}(Z_t)\right] \left[\prod_{t=1}^{T} \frac{\mathbb{P}(Y_t|X_t) \cdot \mathbb{P}(X_t)}{\mathbb{P}(Y_t)}\right] \\
&= \mathbb{P}(Y_1)\mathbb{P}_{Y_1}(Z_1) \cdot \\
&\quad \left[\prod_{t|Y_t \neq Y_{t-1}} \mathbb{P}(Y_t|Y_{t-1})\mathbb{P}_{Y_t}(Z_t)\right] \left[\prod_{t=1}^{T} \frac{\mathbb{P}(Y'_t|X_t) \cdot \mathbb{P}(X_t)}{\mathbb{P}(Y'_t)}\right] \\
&\propto \mathbb{P}(Y_1) \left[\prod_{t=2}^{T} \mathbb{P}(Y'_t|Y'_{t-1})\right] \left[\prod_{t=1}^{T} \frac{1}{\mathbb{P}(Y'_t)}\right] \left[\prod_{t=1}^{T} \mathbb{P}(Y'_t|X_t)\right] \\
&= \pi_{Y_1} \left[\prod_{t=2}^{T} a'_{Y'_t|Y'_{t-1}}\right] \left[\prod_{t=1}^{T} \frac{1}{p'_{Y'_t}}\right] \left[\prod_{t=1}^{T} f_{Y'_t}(X_t)\right].
\end{aligned}$$

Now, the first bracketed term is the augmented transition probability matrix. An MLE estimate for this is given by the empirical frequencies of the Y'_t transitions. However, this is typically not very efficient. Rather, it is better (i) to estimate the non-augmented transition probability matrix with zero diagonal entries by empirical frequencies and (ii) to parameterize the duration distributions (for example, as negative binomial distributions) and then estimate parameters based on the observed durations spent in each state. This, of course, will also be MLE provided the assumed parameterizations are correct. The second bracketed term is the marginal probabilities of the augmented state which can be estimated as the stationary distribution of the estimate of the augmented transition probability matrix.

Finally, the last bracketed term is the conditional class probability estimates

from our classifier. This requires special explanation because, in general, the move from the first line of the above equation to the second requires

$$\prod_{t=1}^{T} \frac{\mathbb{P}(Y_t|X_t) \cdot \mathbb{P}(X_t)}{\mathbb{P}(Y_t)} = \prod_{t=1}^{T} \frac{\mathbb{P}(Y'_t|X_t) \cdot \mathbb{P}(X_t)}{\mathbb{P}(Y'_t)} \tag{5.3.1}$$

which is not in general true. In our case, it is true because

$$\frac{\mathbb{P}(Y_t|X_t)}{\mathbb{P}(Y_t)} = \frac{\mathbb{P}(Y'_t|X_t)}{\mathbb{P}(Y'_t)} \quad \forall t. \tag{5.3.2}$$

To see why this is the case, we should consider that $f_{Y'_t}(X_t) = \mathbb{P}(Y'_t|X_t)$, as written, returns the conditional probability of each class in the state space S' given X_t. However, since the underlying space is S of size k rather than S' of size $\sum_{i=1}^{k} M_i$, we estimate our classifier f as a k-classifier on the space S and adjust it to accommodate the augmented state space S'. As discussed in the previous section, this is reasonable both because the true state space is S and because it makes estimation easier.

The adjustment is the same as the one we presented for m^{th} order Markov models. In particular, we let $f_i(X_t) = \mathbb{P}(Y_t = i|X_t)$ which we estimate by fitting a classifier to the original labels. Then, $f'_i(X_t) = \mathbb{P}(Y'_t = i|X_t)$ is simply a normalized version: $f'_{Y'_t}(X_t) = f_{Y_t}(X_t)\frac{\mathbb{P}(Y'_t)}{\sum_{Y'_t|Y_t=i}\mathbb{P}(Y'_t)}$ where the Y_t corresponding to Y'_t equals i.

Hence

$$\begin{aligned}\frac{\mathbb{P}(Y_t'|X_t)}{\mathbb{P}(Y_t')} &= \frac{\mathbb{P}(Y_t|X_t)\frac{\mathbb{P}(Y_t')}{\sum_{Y_t'|Y_t=i}\mathbb{P}(Y_t')}}{\mathbb{P}(Y_t')} \\ &= \frac{\mathbb{P}(Y_t|X_t)}{\sum_{Y_t'|Y_t=i}\mathbb{P}(Y_t')} \\ &= \frac{\mathbb{P}(Y_t|X_t)}{\mathbb{P}(Y_t)}\end{aligned}$$

As in the first order Markov case, if we believe our classifier is a maximum likelihood estimator for the true conditional class probability function and we believe the emissions depend only on the contemporaneous state Y_t, then this procedure returns a maximum likelihood estimate for $\boldsymbol{f}'$ and hence our parameter estimates $\tilde{\boldsymbol{\Theta}}'$ are maximum likelihood.

In sum, the GMM is parameterized by $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\delta})$ or $\tilde{\boldsymbol{\Theta}} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{f}, \boldsymbol{p}, \boldsymbol{\delta})$. In order to fit the model, one typically estimates the following three parameters. First, one estimates the "standard" k x k transition probability matrix $\boldsymbol{A}$ by using the empirical frequencies of the Y_t transitions, setting the self-transition probabilities to zero and normalizing the rows to sum to one. Second, one parameterizes the duration distributions $\boldsymbol{\delta}$ and then estimates them using the observed durations for each class. Third, one fits a classifier to the Y_t to obtain an estimated k-vector of probabilities for each $X_t = x_t$.

Using these, we can obtain the full parameter estimates. The estimate of $\boldsymbol{A}$ and $\boldsymbol{\delta}$ combine to form an estimate of $\boldsymbol{A}'$. This can be used to obtain an estimate of $\boldsymbol{p}'$;

the estimate of $\boldsymbol{\pi}'$ is either the estimate of $\boldsymbol{p}'$ or a row of the estimated $\boldsymbol{A}'$. Finally, the full augmented conditional class probability estimate $\boldsymbol{f}'$ can be obtained via the procedure discussed above by combining the estimated k-class classifier and the estimate of $\boldsymbol{p}'$.

5.3.3 Transition Dependent GMMs

In this section, we consider a small generalization of the GMM termed the **Transition Dependent Generalized Markov Model** (TDGMM). This model is almost equivalent to a GMM and is parameterized by the same list of parameters: $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\delta})$ in the generative case or $\tilde{\boldsymbol{\Theta}} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{f}, \boldsymbol{p}, \boldsymbol{\delta})$ in the discriminative case. However, now, $\boldsymbol{\delta}$ is no longer a vector of k duration distributions of the form $\delta_i(\tau)$,$i = 1, ..., k$; rather, it is a vector of $k \cdot (k-1)$ duration distribution $\delta_{i,j}(\tau)$ $i = 1, ..., k$, $j = 1, ..., k$, $i \neq j$ where each $\delta_{i,j}(\tau) = \mathbb{P}_{i,j}(\tau)$ gives the probability of arriving in and remaining for τ consecutive periods in state j having come from state i, $\tau = 1, 2, ..., M_{i,j} < \infty$ (where again $M_{i,j}$ is the maximum amount of time spent in state j when arriving from state i and $M = \max_{i,j}(M_{i,j})$).

That is, in this case, the amount of time spent in state j depends not just on the state j itself but also the state i from which state j was reached. This is incredibly useful in practice because often state duration distributions do indeed depend on the previous state. The visualization for this model (as well as the data-generating algorithm) are extremely similar to those in Figure 5.3 and Algorithm 5.4 using $\delta_{i,j}$

rather than δ_i.

We now explain how to embed this model into a first order Markov structure. For any sequence $..., Y_1, Y_2, ..., Y_T, ...$ let us define the sequence of variables $Z_1, ..., Z_T$ as follows. First, we assume that Y_1 is the first episode of its state i and Y_T is the last episode of its state j (i.e., we assume $Y_0 \neq Y_1$ and $Y_T \neq Y_{T+1}$). For simplicity, we also assume Y_0 is known.

As for GMMs, let $Z_t = \text{argmax}_\tau\{Y_t = Y_{t+1} = ... = Y_{t+\tau-1} \neq Y_{t+\tau}\}$ (i.e., the length of time the sequence will remain in the current state). Now, let $U_t = Y_{\sigma_t}$ where $\sigma_t = \text{argmin}_s(Y_s \neq Y_{s+1} = Y_{s+2} = ... = Y_t)$ (i.e., it is the last state the sequence was in before it got to the one it is currently in). Finally, we let $Y_t' = (U_t, Y_t, Z_t)$ (this triplet represents the state the sequence came from, the state it is currently in, and how much longer it will remain in the current state). Hence, we

have moved from state space $S = \{1, ..., k\}$ to

$$
\begin{aligned}
S' \;=\; & \{(2,1,1),(2,1,2),...,(2,1,M_{2,1}),\\
& (3,1,1),(3,1,2),...,(3,1,M_{3,1}),\\
& ...,\\
& (k,1,1),(k,1,2),...,(k,1,M_{k,1}),\\
& (1,2,1),(1,2,2),...,(1,2,M_{1,2}),\\
& (3,2,1),(3,2,2),...,(3,2,M_{3,2}),\\
& ...,\\
& (k,2,1),(k,2,2),...,(k,2,M_{k,1}),\\
& ...,\\
& (1,k,1),(1,k,2),,...,(1,k,M_{1,k}),\\
& (2,k,1),(2,k,2),,...,(2,k,M_{2,k}),\\
& ...,\\
& (k-1,k,1),(k-1,k,2),...,(k-1,k,M_{k-1,k})\}
\end{aligned}
$$

which for simplicity we sometimes write as

$$
\begin{aligned}
S' \;=\; \{&(2,1,1),(2,1,2),...,(2,1,M),\\
&(3,1,1),(3,1,2),...,(3,1,M),\\
&...,\\
&(k,1,1),(k,1,2),...,(k,1,M),\\
&(1,2,1),(1,2,2),...,(1,2,M),\\
&(3,2,1),(3,2,2),...,(3,2,M),\\
&...,\\
&(k,2,1),(k,2,2),...,(k,2,M),\\
&...,\\
&(1,k,1),(1,k,2),,...,(1,k,M),\\
&(2,k,1),(2,k,2),,...,(2,k,M),\\
&...,\\
&(k-1,k,1),(k-1,k,2),...,(k-1,k,M)\}
\end{aligned}
$$

as we did for GMMs. For example, consider the sequence $a, a, b, b, b, b, c, c, a$ and suppose it came from b; it gets transformed into

$$(b,a,2),(b,a,1),(a,b,4),(a,b,3),(a,b,2),(a,b,1),(b,c,2),(b,c,1),(c,a,1).$$

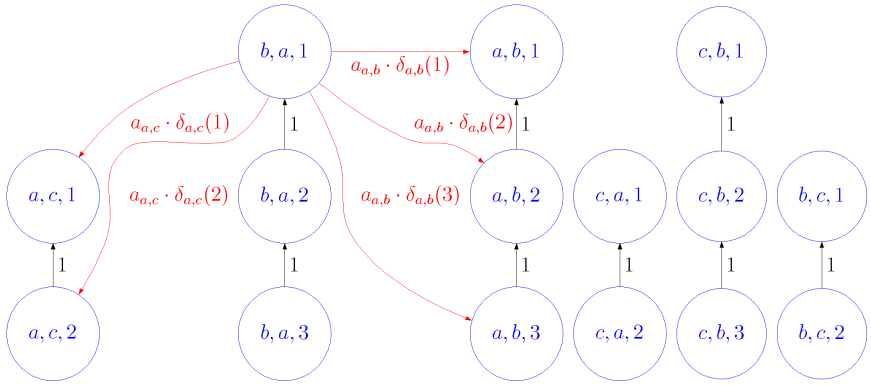

Figure 5.5: The transition diagram corresponding to a generalized Markov chain with state space $S = \{a, b, c\}$, maximal durations $M_{b,a} = M_{a,b} = M_{c,b} = 3$, $M_{c,a} = M_{a,c} = M_{b,c} = 2$, transition probabilities given by $\boldsymbol{A}$ whose entries are a_{ij}, and duration distributions given by $\boldsymbol{\delta} = (\delta_{b,a}, \delta_{c,a}, \delta_{a,b}, \delta_{c,b}, \delta_{a,c}, \delta_{b,c})^T$. The augmented state space is $S' = \{(b, a, 3), (b, a, 2), (b, a, 1), (c, a, 2), (c, a, 1), (a, b, 3), (a, b, 2), (a, b, 1), (c, b, 3), (c, b, 2), (c, b, 1), (a, c, 2), (a, c, 1), (b, c, 2), (b, c, 1)\}$ and is indicated in blue. The augmented transition probabilities $\boldsymbol{A'}$ which are equal to one are given in black. Those that are equal to the products of the $a_{i,j}$ and $\delta_{i,j}(\tau)$ and which originate from $(b, a, 1)$ are given in red. Similar transitions from $(c, a, 1)$ to each (a, b, n) and (a, c, n); from $(a, b, 1)$ to each (b, a, n) and (b, c, n); from $(c, b, 1)$ to each (b, a, n) and (b, c, n); from $(a, c, 1)$ to each (c, a, n) and (c, b, n); and from $(b, c, 1)$ to each (c, a, n) and (c, b, n) are omitted for aesthetic reasons but are formed in an way analogous to that shown in the picture and described in the main text.

Now, we can form the augmented transition probability matrix $\boldsymbol{A'}$ as follows. Each transition from (i, j, τ) to $(i, j, \tau - 1)$ has probability one for $\tau > 1$ and all other transitions have probability zero. For $\tau = 1$ (that is, $Y'_t = (i, j, 1)$) there are three cases: (i) transitions to (i, j, n) have probability zero for all $n = 1, ..., M_{i,j}$; (ii) transitions to (i', j', n) have probability zero for all $n = 1, ..., M_{i',j'}$ and all $i' \neq j$; and (iii) transitions to (i', j', n) have probability $a_{i',j'}\delta_{i',j'}(n)$ for $i' = j$, $j' \neq j$, and $n = 1, ..., M_{i',j'}$. This is perhaps better explained via an example which we give in Figure 5.5.

The likelihood is practically identical to the GMM case and is therefore omitted. The estimation of either $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\delta})$ or $\tilde{\boldsymbol{\Theta}} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{f}, \boldsymbol{p}, \boldsymbol{\delta})$ is also quite similar to the GMM case and the same three parameters. First, one estimates the "standard" k x k transition probability matrix $\boldsymbol{A}$ by using the empirical frequencies of the Y_t transitions, setting the self-transition probabilities to zero and normalizing the row sums to one. Second, one parameterizes the duration distributions $\boldsymbol{\delta}$ and then estimates them using the observed durations for each class conditional on the previous class. Third, one fits a classifier to the Y_t to obtain an estimated k-vector of probabilities for each $X_t = x_t$.

Using these, we can obtain the full parameter estimates. The estimate of $\boldsymbol{A}$ and $\boldsymbol{\delta}$ combine to form an estimate of $\boldsymbol{A}'$. This can be used to obtain an estimate of $\boldsymbol{p}'$; the estimate of $\boldsymbol{\pi}'$ is either the estimate of $\boldsymbol{p}'$ or a row of the estimated $\boldsymbol{A}'$. Finally, the full augmented conditional class probability estimate $\boldsymbol{f}'$ can be obtained via the procedure discussed above by combining the estimate k-class classifier and the estimate of $\boldsymbol{p}'$.

5.3.4 GMMs and TDGMMs With Infinite State Durations

A disadvantage of the proposed methodology is that, thus far, we have required each duration distribution to have finite support. This has been necessary in order to embed the generalized models into the first order Markov structure so that the forward-backward and Viterbi algorithms can be used.

In this brief section, we relax that assumption. Our duration distributions now can have support on the positive integers provided that, in the tail, the distribution is geometric. Formally, let $\delta(\tau)$ be a duration distribution. Then we can write δ as

$$\delta(\tau|\theta, q, s, M) = qf(\tau|\theta)I(\tau \leq M) + (1-q)g(\tau|s)I(\tau > M). \tag{5.3.3}$$

Here, f is any probability density on $1, ..., M$ parameterized by θ. The probability of being in the tail is given by $1-q \in [0,1]$ and g is a shifted geometric distribution supported on $M+1, M+2, ...$ with $g(M+n|s) = s^{n-1} \cdot (1-s)$ for $n = 1, 2, ...$

We provide an in-depth explanation for the GMM with infinite duration (which we term the GMM$^+$) noting that it applies analogously to the TDGMM (termed the TDGMM$^+$). The augmented state space is just like that of the GMM except with one augmented state for each original state which corresponds to the tail. Namely, for $S = \{1, ..., k\}$ we get

$$\begin{aligned} S' \quad = \quad & \{(1,1), (1,2), ..., (1, M_1), (1, M_1^+ = M_1 + 1), \\ & (2,1), (2,2), ..., (2, M_2), (2, M_2^+ = M_2 + 1), \\ & ..., \\ & (k,1), (k,2), ..., (k, M_k)(k, M_k^+ = M_k + 1), \} \end{aligned}$$

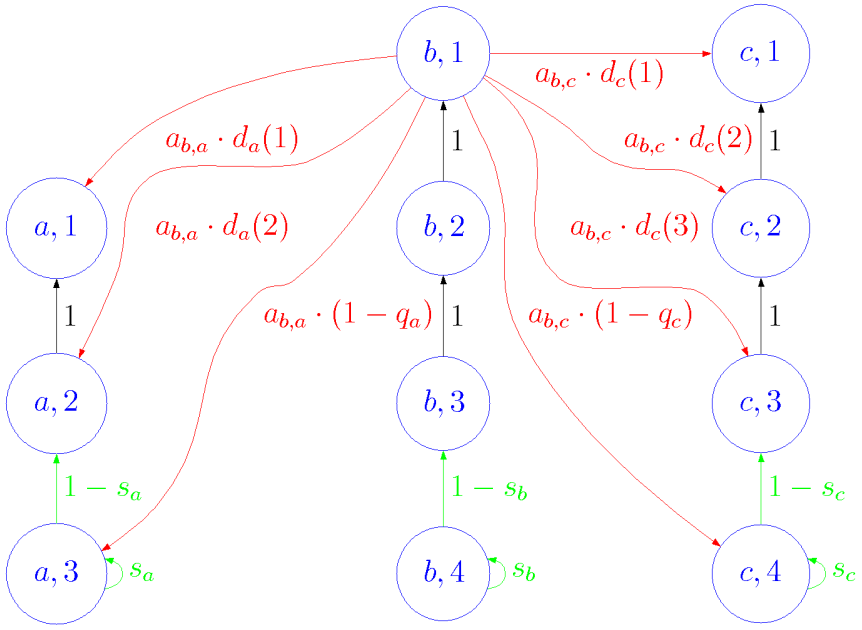

Figure 5.6: The transition diagram corresponding to a generalized Markov chain with state space $S = \{a, b, c\}$, tail definition $M_a = 2$ and $M_b = M_c = 3$, transition probabilities given by $\boldsymbol{A}$ whose entries are a_{ij}, duration distributions given by $\boldsymbol{d} = (d_a, d_b, d_c)^T$, tail self-transition probabilities given by s_i, and tail mass given by $\boldsymbol{q} = (q_a, q_b, q_c)^T$ where $q_i = \sum_{\tau=1}^{M_i} d_i(\tau)$. The augmented state space is $S' = \{(a,3), (a,2), (a,1), (b,4), (b,3), (b,2), (b,1), (c,4), (c,3), (c,2), (c,1)\}$ and is indicated in blue. The augmented transition probabilities $\boldsymbol{A}'$ which are equal to one are given in black. Those that are equal to the products of the $a_{i,j}$ and $\delta_i(\tau)$ and which originate from $(b,1)$ are given in red. Similar transitions from $(a,1)$ to each (b,n) and (c,n) and from $(c,1)$ to each (a,n) and (b,n) are omitted for aesthetic reasons but are formed in an way analogous to that shown in the picture and described in the main text. Most importantly, the "tail self-transition probabilities" which allow for an infinite duration distribution as well as their corresponding "out-of-tail transition probabilities" are given in green. This figure is the GMM$^+$ analogue of the GMM given in Figure 5.4.

which we sometimes for simplicity define as

$$\begin{aligned} S' &= \{(1,1),(1,2),...,(1,M),(1,M^+ = M+1), \\ &\quad (2,1),(2,2),...,(2,M),(2,M^+ = M+1), \\ &\quad ..., \\ &\quad (k,1),(k,2),...,(k,M)(k,M^+ = M+1),\}. \end{aligned}$$

Since each state has its own δ, we have a collection $\boldsymbol{\delta} = (\delta_1, ..., \delta_k)^T$ with

$$\delta_i(\tau|\theta_i, q_i, s_i, M_i) = q_i f_i(\tau|\theta_i) I(\tau \leq M_i) + (1-q_i) g(\tau|s_i) I(\tau > M_i)$$

for $i = 1, ..., k$. And, for convenience, we define $\boldsymbol{d} = (d_1(\tau), ..., d_k(\tau))^T$ where $d_i(\tau)$ is a vector of size M_i giving $q_i f_i(\tau|\theta_i)$ for $\tau = 1, ..., M_i$.

The augmented transition probability matrix $\boldsymbol{A'}$ is formed in a way very similar to the GMM. Each transition from (i, τ) to $(i, \tau - 1)$ has probability one for $\tau = 2, ..., M_i$ and all other transitions have probability zero. For $\tau = 1$ (that is, $Y'_t = (i, 1)$) there are two cases: (i) transitions to (i, n) have probability zero for all $n = 1, ..., M_i$ and (ii) transitions to (j, n) have probability $a_{i,j}\delta_j(n) = q_i f_i(\tau|\theta_i) = d_i(\tau)$ for $i \neq j$ and $n = 1, ..., M_j$. For $(i, M_i^+ = M_i + 1)$ there is a self-transition with probability s_i and a transition to (i, M_i) with probability $(1 - s_i)$. An example is illustrated in Figure 5.6.

The augmented matrix $\boldsymbol{A'}$ yields $\boldsymbol{\pi'}$ and $\boldsymbol{p}$ as above. Furthermore, the likelihood

is equivalent to the GMM case. However, the parameter estimation strategy is usually a bit different.

The GMM$^+$ model is parameterized by $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\delta})$ or $\tilde{\boldsymbol{\Theta}} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{f}, \boldsymbol{p}, \boldsymbol{\delta})$ where each δ_i is as above. We usually use the equivalent parameterization $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{d}, \boldsymbol{s}, \boldsymbol{q})$ or $\tilde{\boldsymbol{\Theta}} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{f}, \boldsymbol{p}, \boldsymbol{d}, \boldsymbol{s}, \boldsymbol{q})$ instead. In this parameterization, $\boldsymbol{d} = (d_1(\tau), ..., d_k(\tau))^T$ and each $d_i(\tau)$ is a vector of size M_i giving $q_i f_i(\tau|\theta_i)$ for $\tau = 1, ..., M_i$. Similarly, $\boldsymbol{s} = (s_1, ..., s_k)^T$ where each s_i is the self-transition probability for the tail state (i, M_i^+). Finally, $\boldsymbol{q} = (q_1, ..., q_k)^T$ is the total mass in the non-tail portion of each state given by $\sum_{\tau=1}^{M_i} \delta_i(\tau) = \sum_{\tau=1}^{M_i} d_i(\tau)$.

In order to fit the model, one typically estimates the following three parameters. First, one estimates the "standard" k x k transition probability matrix $\boldsymbol{A}$ by using the empirical frequencies of the Y_t transitions, setting the self-transition probabilities to zero. Second, one parameterizes the duration distributions $\boldsymbol{\delta}$ as in Equation 5.3.3 and then estimates them using the observed durations for each class, saving the d_i and s_i components from each δ_i to yield $\boldsymbol{d}$, $\boldsymbol{s}$, and $\boldsymbol{q}$. Third, one fits a classifier to the Y_t to obtain an estimated k-vector of probabilities for each $X_t = x_t$.

Using these, we can obtain the full parameter estimates. The estimates of $\boldsymbol{A}$, $\boldsymbol{d}$, $\boldsymbol{s}$, and $\boldsymbol{q}$ combine to form an estimate of $\boldsymbol{A}'$. This can be used to obtain an estimate of $\boldsymbol{p}'$; the estimate of $\boldsymbol{\pi}'$ is either the estimate of $\boldsymbol{p}'$ or a row of the estimated $\boldsymbol{A}'$. Finally, the full augmented conditional class probability estimate $\boldsymbol{f}'$ can be obtained via the procedure discussed above by combining the estimate k-class classifier and

the estimate of $\boldsymbol{p}'$.

A very similar strategy can be applied to form a TDGMM$^+$.

5.3.5 Other Approches to Extending Markov Models

We are not the first to generalize Markov models to accommodate duration distributions other than the geometric. For example, Ferguson (1980) allowed for non-parametric probability mass functions for each duration and Levinson (1981) allowed for the continuous durations given by normal and gamma distributions (truncated to have finite minima and maxima). Such variants on the Markov model, which we have termed here a GMM, are also referred to as explicit-duration Markov models, variable duration Markov models, Markov models with explicit duration, and semi-Markov models.

As should be evident, these are all more or less variants of the same broad GMM approach. To our knowledge, no one however has combined these strategies with the discriminative approach, extended it to a TDGMM, allowed for infinite durations (via GMM$^+$ and TDGMM$^+$ geometric tails), or embedded the more general model in first order Markov chain as above allowing for use of the forward-backward and Viterbi algorithms.

Another approach, which appears quite different is that of the non-stationary Markov model (NSMM) (Sin and Kim, 1995; Vaseghi, 1995; Brillinger *et al.*, 2000; Djuric and Chun, 2002)[1]. The NSMM is parameterized in exactly the same way

[1]Djuric and Chun (2002) is particularly interesting because it estimates the model in a Bayesian

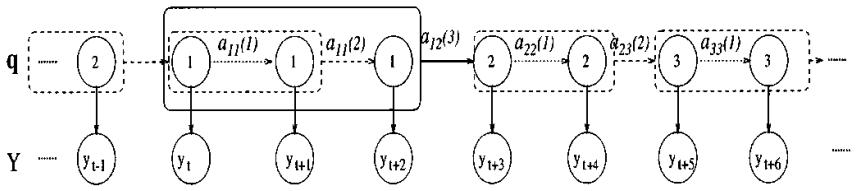

Figure 5.7: A Non-Stationary Markov Model with states given by $\mathbf{q} \in \{1, 2, 3\}$ and covariates y_t. At each time, the transition probabilities are determined by which state the process is currently in and how long it has been there. Source: Djuric and Chun (2002).

Algorithm 5.5 Generating Data From a Non-Stationary Markov Model

1. Draw Y_1 from the initialization distribution, $\mathbb{P}(X_1)$.
2. Record the duration of the current state (i.e., d_t=1).
3. Draw X_1 from the covariate emission distribution, $\mathbb{P}(X_1|Y_1)$.
4. For $t = 2, ..., T$:
   (a) Draw Y_t from the transition probability distribution, $\mathbb{P}_{d_{t-1}}(Y_t|Y_{t-1} = ... = Y_{t-d_{t-1}})$.
   (b) Record the duration of the current state as d_t.
   (c) Draw X_t from the covariate emission distribution, $\mathbb{P}(X_t|Y_t)$.

as the standard Markov model with one exception: instead of a k x k transition probability matrix $\boldsymbol{A}$, there are now a sequence of them $\{\boldsymbol{A}^\tau\}_{\tau=1,...,M\leq\infty}$ where the entries $a^\tau_{i,j} = \mathbb{P}_\tau(Y_t = j|Y_{t-1} = Y_{t-2} = ... = Y_{t-\tau} = i)$ give the probability of transitioning from state i to state j given that the sequence has spent τ consecutive periods in state i. We can generate data from this model using Algorithm 5.5 and it is illustrated in Figure 5.7.

While it should be clear that diagonal entries of the transition probability matrix sequence $\{\boldsymbol{A}^\tau\}_\tau$ induce duration distributions on each of the states, what may be surprising is that the NSMM is actually *equivalent* to a GMM (Djuric and Chun,

fashion using Markov chain Monte Carlo (MCMC) methods including the Gibbs (Geman and Geman, 1984) and Metropolis-Hastings (Metropolis *et al.*, 1953; Hastings, 1970) samplers.

2002) and that the former can sometimes be more tractably implemented. Hence, many of the other approaches explored in the literature are special cases of or are equivalent to the GMM approach laid out here (and therefore can also be embedded in a first order Markov model).

It should be clear that one could also represent a TDGMM as a NSMM if we make the transition matrix more general, $\{\boldsymbol{A}^{i,\tau}\}_{i=1,\ldots,k,\tau=1,\ldots,M\leq\infty}$ where τ is as above and i is the unique state the sequence was previously in before arriving at the current state j.

5.3.6 Variable Length Markov Models

The final extension we consider is the Variable Length Markov Model (VLMM) (Buhlmann and Wyner, 1999). These are perhaps best described by trees as is done in Figure 5.8. Essentially, a VLMM is like an m^{th} order Markov model where m depends on the most recent sequence one has observed. Using the example of Figure 5.8, if the Y_{t-1} and Y_{t-2} were both a, then the process resembles a third order Markov chain because the distribution of Y_t given $(Y_{t-1}, Y_{t-2}, Y_{t-3})$ equals (a, a, a), (a, a, b), and (a, a, c) differ from one another (i.e., $\alpha_1 \neq \alpha_2 \neq \alpha_3$). If ($Y_{t-1}$,$Y_{t-2}$) were ($a$,$b$) or ($a$,$c$), the process resembles a second order Markov chain. When $Y_{t-1} = b$, it is like a first order Markov chain and when $Y_{t-1} = c$ it is like a second order Markov chain.

In fact, a VLMM is a special case of the m^{th} order Markov model with restrictions

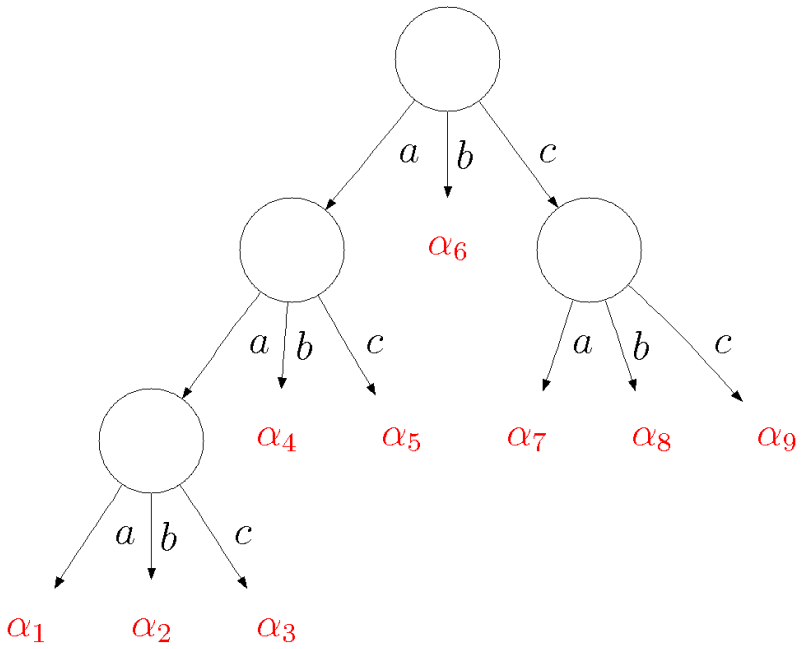

Figure 5.8: A Variable Length Markov Model on the state space $S = \{a, b, c\}$. Each terminal node α_j is a vector of length 3 giving the transition probabilities to each of the three states in S. One reads from the top down such that $\alpha_3 = \mathbb{P}(Y_t | Y_{t-1} = a, Y_{t-2} = a, Y_{t-3} = c)$. In general, each of the α_j will be unique though this does not necessarily have to be the case and estimation strategies which shrink them to a common distribution can be effective.

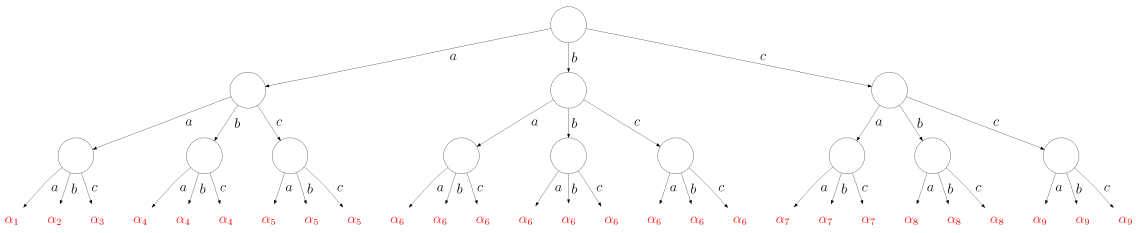

Figure 5.9: The Variable Length Markov Model from Figure 5.8 represented as a Third Order Markov Model.

on the distribution of $\mathbb{P}(Y_t|Y_{t-1}, ..., Y_{t-m})$. To see this, let m be the maximum depth of the tree corresponding to VLMM (in the case of Figure 5.8, $m = 3$). Now, consider a tree fully grown to depth m such that it has k^m terminal nodes. For each terminal node in the new tree, give it the terminal distribution α if an equivalent terminal node exists in the original tree and has terminal distribution α. If there is no such corresponding node in the original tree, use the α corresponding to the closest parent, grandparent, etc. which does exist. For instance, in the example of Figure 5.8, in terminal node (a, a, c) of the fully grown new tree would have terminal distribution α_3 since this node exists in the original tree; on the other hand, node (a, b, a) has no equivalent in the original tree but has closest parent (a, b) and therefore would receive terminal distribution α_4. The fully grown tree which gives the VLMM of Figure 5.8 as an $m = 3$ order Markov model is given by Figure 5.9.

Now, since we know how to express m^{th} order Markov chains as first order Markov chains, we can easily accommodate VLMMs using the methodology discussed above[2]. This is quite advantageous as it overcomes the principal, critical flaw of m^{th} order Markov models: for large m, it is difficult to estimate $\boldsymbol{A}'$ even with the sparsity required by the fact that $S' = \prod_{i=1}^{m} S$.

Estimation of the VLMM tree structure is conceptually similar to that of CART trees: we want to recursively partition a space and return a probability distribution

[2]There are clearly more computationally efficient ways to implement a VLMM than by extending it to a higher order Markov model. However, this representation is clean and elegant and computational efficiency is not the focus of this work.

on S that depends on that partition. There are two principal differences however. First, in a VLMMs, our data is a sequence $\boldsymbol{y}$ whereas in CART trees our data is a set of (y, x) pairs drawn $i.i.d.$ from some joint distribution. This leads to the second difference: in CART trees the partitions are of the space of the covariates x whereas in VLMMs the partitions depend on the most recent values of the sequence $\boldsymbol{y}$. Nonetheless, similar strategies can be followed. One can continue to partition until fewer than some number of observations would lie in a terminal bin. One can also estimate to some maximal depth size or maximum number of terminal bins. Finally, one can grow a large tree and prune based on various criteria. For an in-depth study of estimation of VLMMs, see Buhlmann and Wyner (1999).

While we do not fit VLMMs in this study, we discuss them because GMMs and TDGMMs (and GMM$^+$s and TDGMM$^+$s) can be expressed as VLMMs with a very strong set of restrictions placed on them. Namely, GMMs are VLMMs with each split after the first being a binary split. It is binary reflecting the fact that, in a GMM, the transition from state i to state j only depends on how long the sequence has spent in state i: hence, after the first split (which is a k-split, that is, has k branches), all splits are binary and of the form state i versus "all other states" (where state i refers to the branch of the first split). This is shown in Figure 5.10.

There are even further restrictions: at each terminal node, the α probability vectors are not free to vary unrestricted. Rather, they have structure imposed on them by the duration distributions $\boldsymbol{\delta}$ which are typically heavily parameterized.

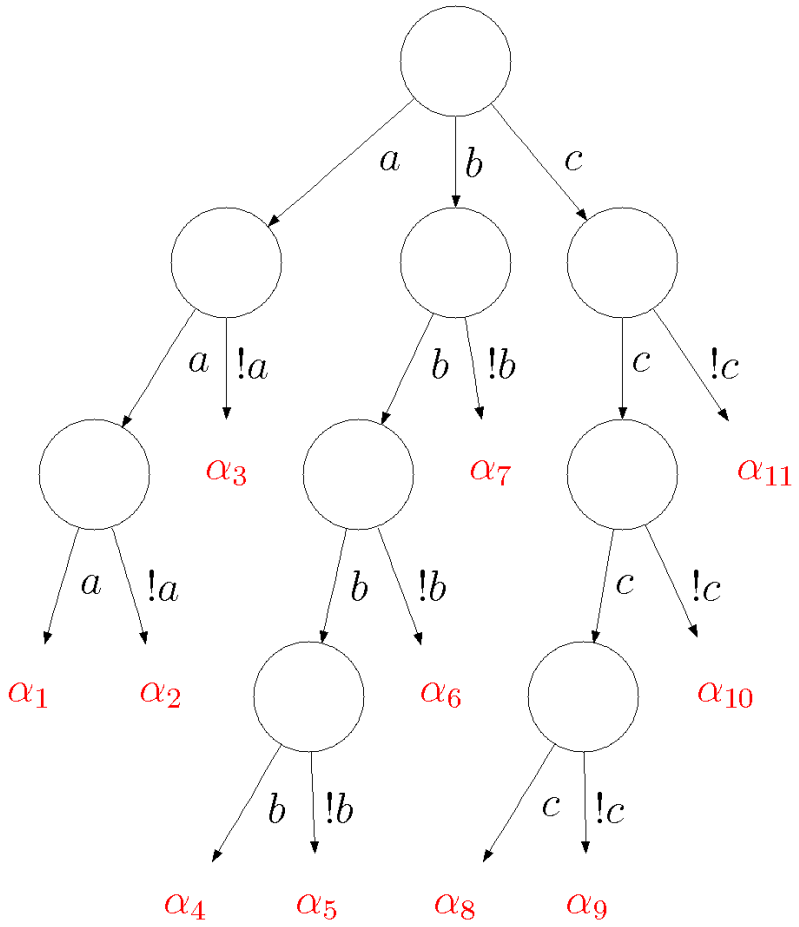

Figure 5.10: The Generalized Markov Model with Geometric Tail from Figure 5.6 represented as a Variable Length Markov Model. By $!i$, we denote all other states except state i. For example, $\alpha_2 = \mathbb{P}(Y_t|Y_{t-1} = a, Y_{t-2} = a, Y_{t-3} \neq a)$

Since GMMs are equivalent to NSMMs (that is, the δ_i of the GMM are reflected in the $\{\boldsymbol{A}^\tau\}_\tau$ of the NSMM), we will describe the α in terms of the latter since it is more straightforward. Namely, at each terminal node, the transition probabilities are given by the i^{th} row of $\boldsymbol{A}^\tau$ from an NSMM where τ is the depth of the terminal node and i reflects the initial branch. The distinction between a GMM and GMM$^+$ is reflected in the terminal node at depth M_i: if the self-transition probability is zero, the VLMM is a GMM whereas, if it is non-zero, it is a GMM$^+$.

Similarly, TDGMMs can be represented as VLMMs. However, in this case, the trees are grown fully as in an m^{th} order Markov chain. However, due to the fact that the transition probabilities at each t can only depend on how long the sequence has been in the current state j and the previous state i the sequence was in before j, one must take the j^{th} row of the more complicated NSMM formulation with transition probability matrices $\{\boldsymbol{A}^{i,\tau}\}_{i,\tau}$ touched on at the end of the previous section. As above, the distinction between a TDGMM and TDGMM$^+$ will depend on whether there are self-transitions at terminal nodes.

5.4 Conclusion

In conclusion, we have presented the PrAGMaTiSt, a methodology which overcomes some of the principal disadvantages of standard first order Markov models. Using our discriminative training formulation, we no longer have to estimate k p-variate probability distributions. This also allows us to introduce sliding window and long-

term features into the covariate space X_t. Furthermore, it allows us to use *any* standard classification algorithm to train the Markov model. Finally, by considering various generalizations of the first order Markov structure, we can introduce dependence directly into the Y_t sequence itself, no longer requiring the relationship between Y_t and Y_{t+k} to be indirect via $Y_{t+1}, ..., Y_{t+k-1}$.

We have also seen that it is possible to embed very complicated and general dependence structures into a first order Markov model, thereby allowing use of the forward-backward and Viterbi algorithms. This is absolutely critical to our endeavor because, otherwise, our model could be specified and estimated but we would be unable to apply it to test samples.

We have also seen that our most general model, the TDGMM$^+$, is related to and can embed many models considered in the literature. In fact, the only more general models than TDGMM$^+$ are the m^{th} order Markov chain and the VLMM which both embed it. The former is intractable in terms of both estimation and computation for large m. The latter, while quite useful, is also difficult to estimate when the dependence structures go back far in time. The PrAGMaTiSt has no trouble with long dependence structures because we can parameterize the $\delta_{i,j}(\tau)$ and efficiently estimate parametric distributions. While something similar in principle could be achieved by a "hierarchical" VLMM where the α in each branch are shrunk towards a common distribution, no methods exist currently to implement such a strategy.

In the sequel, we apply the PrAGMaTiSt to both simulated data and sleep

data. The former is useful to explore the properties of our methodology and, in particular, in what situations it thrives and fails. The results of this exploration are both interesting in their own right and also useful for understanding how the model performs on the sleep data. In the latter section, we show increased ability to provide automatic sleep scores based on video data and demonstrate notable success versus other methods on the difficult, rare, and interesting REM state.

Chapter 6

Simulations

6.1 Simulation Design

In this chapter, we evaluate the PrAGMaTiSt on simulated data considering various permutations of the base classifier (e.g., logistic regression, Random forests, etc.) and time series structure (first order Markov model, GMM, etc.) and compare its performance to competing methods. By using simulated data, we can see how well the true model performs when estimated as well how various incorrect models perform. Simulations also provide knowledge of the true marginal probabilities $\mathbb{P}(Y_t = i|X_{1:T} = \boldsymbol{X})$ on hold-out samples thus allowing us to evaluate our model's ability to estimate probabilities as well as classify. We lay out the simulation design in broad detail below.

For each simulation, we set the training sample size at $n = 100$, $n = 1000$,

and $n = 10000$ successively and fix the test set size at 200. We then repeat each simulation 100 times and average over the results in order to get (i) a better estimate of the expected errors and (ii) a measure of their variability. Our simulation features two covariate distributions (One and Two) for which we vary the amount of noise in the distribution and four time series structures (A, B, C, and D). All simulations use the state space $S = \{a, b, c\}$. In all simulations, the test data "continues" from the training data as in Figure 5.1 so that the test set initialization distribution can be obtained from the estimate of $\boldsymbol{A}'$.

In simulation one, the covariate structure $\boldsymbol{\mu}$ is $X_t \sim N(\mu_{Y_t}, \sigma^2)$ where $\mu_a = 0$, $\mu_b = 1$, and $\mu_c = 2$. We set σ to thirteen different values: $0, 1/4, 1/2, ..., 3$. The competing methodologies we consider are multinomial logistic regression; first order Markov models, GMMs, and TDGMMs (or their infinite geometric tail $^+$ equivalents where appropriate) trained discriminatively using multinomial logistic regression as the base classifier; and the MALLET implementation of CRFs.

In simulation two, the covariate structure $\boldsymbol{\mu}$ follows a complicated checkerboard pattern with five different levels of noise. The covariate vector X_t is ten-dimensional with two active dimensions and eight inactive, noise dimensions which are distributed $U(0, 1)$ regardless of Y_t. The conditional distributions of the two active dimensions for the five noise levels are shown in Figure 6.1. The unit square is divided into nine sub-squares and each of the three classes have three sub-squares for which they are the "dominant" class. Within each square, the dominant class

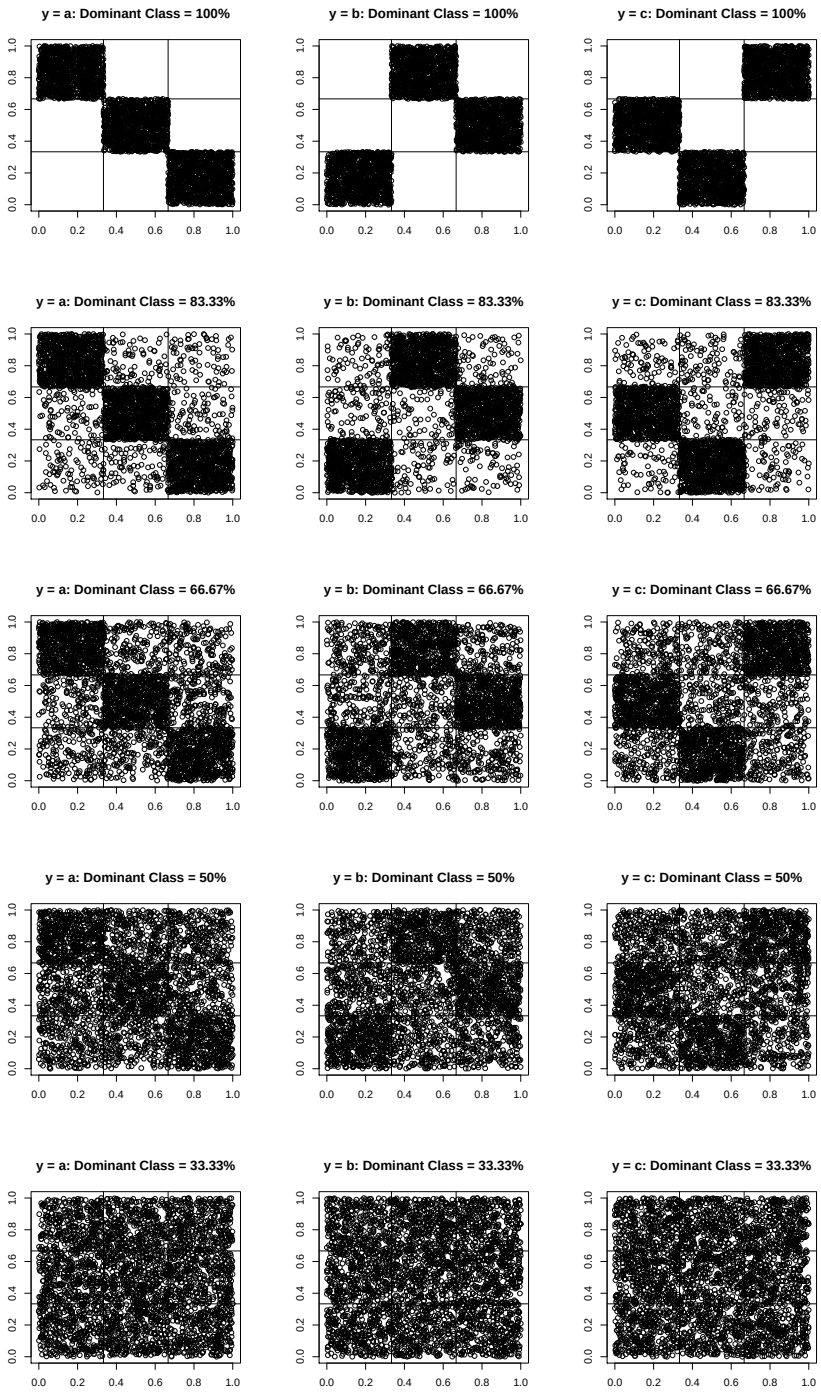

Figure 6.1: The covariate emission distribution for Simulation Two. Each column gives the distribution of the active two covariates conditional of the state Y_t. The rows show the five levels of noise considered. The eight inactive covariates are distributed $U(0, 1)$.

receives successively 100%, 83.33%, 66.67%, 50%, and 33.33% of the probability, with the other two classes receiving half of the remaining probability each. These probabilities correspond to a Bayes Error rate of 0%, 16.67%, 33.33%, 50%, and 66.67% in the non-sequential learning task where each of the three classes has equal marginal base rate 1/3.

The competing methodologies we consider for simulation two are multinomial logistic regression, AdaBoost, and Random forests; Markov models, GMMs, and TDGMMs (or their $^+$ equivalents where appropriate) trained discriminatively using multinomial logistic regression, AdaBoost, and Random forests respectively as the base classifiers; the MALLET implementation of CRFs (since the decision boundary is highly non-linear, we trained MALLET in three ways: once on the original covariates, once on the response surface of order two formed from the original covariates, and once on the response surface of order three formed from the covariates[1]); and TreeCRF, a tree-based version of CRFs[2].

The time series structure for simulation A is no time series structure at all, but the three classes have marginal base rates of $\boldsymbol{p} = (.3, .2, .5)^T$ respectively. Our initialization distribution is the uniform distribution $\boldsymbol{\pi} = (1/3, 1/3, 1/3)^T$.

For simulation B, the time series structure is a first order Markov Model with

[1]For $n = 10000$, training MALLET on the response surface of order three was computationally infeasible.

[2]TreeCRF can only accommodate binary features. We formed binary features from our continuous ones by binning the unit interval into equally sized bins for each dimension of the covariate space and creating an indicator for whether the covariate fell in the bin. Since there was no *ex ante* way to know how many to use, we tried 5, 10, 25, and 100. In addition, TreeCRF has a parameter for the number of terminal nodes or leaves. We set this to 8, 16, 32, 64, 128, and 256 and tried all combinations of bin and leave numbers.

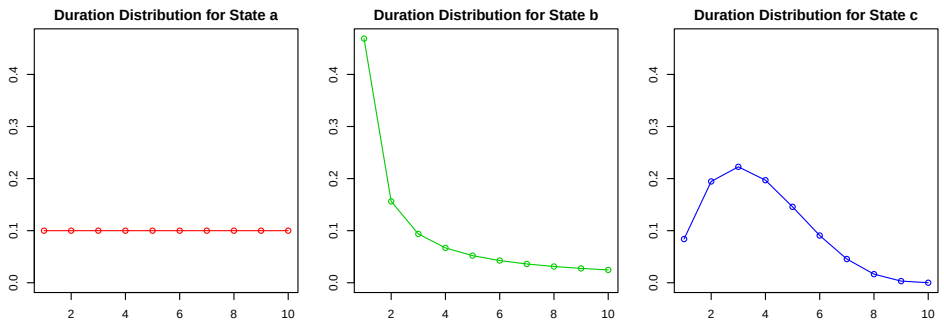

Figure 6.2: The duration distributions for Simulation C, a GMM. Each distribution is Discrete Beta with $M = 10$ and (α, β) set to $(0, 0)$, $(-15, 0)$, and $(.75, 1.5)$ respectively.

transition probability matrix given by

$$\boldsymbol{A} = \begin{pmatrix} \frac{3}{10} & \frac{2}{10} & \frac{5}{10} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{2}{10} & \frac{7}{10} & \frac{1}{10} \end{pmatrix}.$$

The initialization distribution is given by the uniform distribution $\boldsymbol{\pi} = (1/3, 1/3, 1/3)^T$.

For simulation C, the time series structure is a GMM with transition probability matrix given by

$$\boldsymbol{A} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & 0 & \frac{1}{4} \\ \frac{1}{3} & \frac{2}{3} & 0 \end{pmatrix}.$$

The duration distributions δ_i are given by a discretized version of the Beta distribution. The discrete Beta takes two parameters, the usual Beta parameters α and β, and is supported on $1, ..., M$ where M is a pre-specified integer. The probabil-

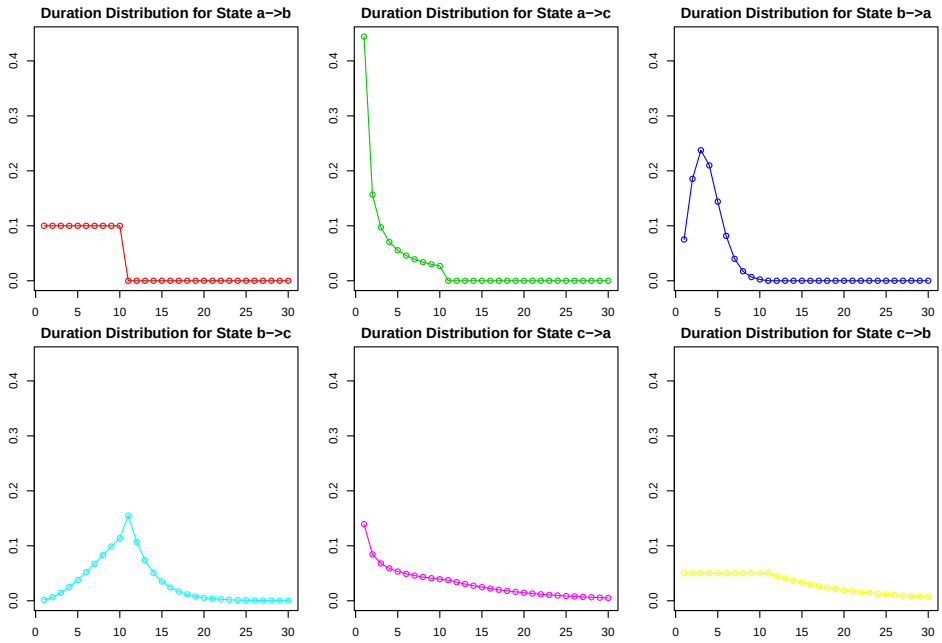

Figure 6.3: The duration distributions for Simulation D, a TDGMM$^+$. Each distribution is Beta Negative Binomial with Geometric Tail. See the main text for a description of the probability mass function and the parameter values. The label i -> j denotes distribution for the length of time spent in state j when arrived at from state i.

ity mass function given by $p_i / \sum_{j=1}^{M} p_j$, $i = 1, ..., M$ where p_i is value of the Beta probability density function with parameters α and β evaluated at x_i and where $\{x_i\}_{i=1}^{M}$ is the equally-spaced sequence of length M from $1/2M$ to $1 - 1/2M$. In this simulation, we set M to ten for all three states and set the parameters (α, β) to $(0, 0)$, $(-15, 0)$, and $(.75, 1.5)$ for states a, b, and c. The probability mass values are shown in Figure 6.2. The initialization distribution is given by the uniform distribution $\boldsymbol{\pi} = (1/3, 1/3, 1/3)^T$.

For simulation D, the time series structure is a TDGMM$^+$ with transition probability matrix given as in Simulation C. The duration distributions $\delta_{i,j}$ are of the

form

$$\delta(\tau|\alpha,\beta,r,q,s,M) = q\frac{1}{c(M)}f(\tau|\alpha,\beta,r)I(\tau \leq M) + (1-q)g(\tau|s)I(\tau > M) \quad (6.1.1)$$

with M set to 10 and f set to the Beta Negative Binomial distribution,

$$f(\tau|\alpha,\beta,r) = \frac{\Gamma(\alpha+\beta)\Gamma(\alpha+r)\Gamma(\tau+r-1)\Gamma(\tau+\beta-1)}{\Gamma(r)\Gamma(\alpha)\Gamma(\beta)\Gamma(\tau)\Gamma(\tau+r+\alpha+\beta-1)},\ x = 1,...,M. \quad (6.1.2)$$

As always, g is the geometric distribution (shifted to $M+1, M+2, ...$) and we define the constant $c(M) = \sum_{\tau=1}^{M} f(\tau|\alpha,\beta,r)$. The parameters (α,β,r,q,s) for the distributions $\delta_{i,j}$ were set to

State	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	**r**	**q**	**s**
a -> b	0.00	1.00	442413	1.00	0.00
a -> c	0.00	1.65	0.45	1.00	0.00
b -> a	1808	148.41	33.12	1.00	0.00
b -> c	0.00	7.39	7.39	0.50	0.69
c -> a	0.00	22026	0.61	0.62	0.90
c -> b	0.00	1.00	442413	0.50	0.90

and were chosen to provide a diversity of shapes (see Figure 6.3), some with finite support and some with infinite support. The initialization distribution is given by the uniform distribution $\boldsymbol{\pi} = (1/3, 1/3, 1/3)^T$.

In Simulations A, B, and D, the GMM and TDGMM models assume a Beta Negative Binomial with Geometric Tail for the δ duration distributions with M fixed to 10. In Simulation C, they assume a discrete Beta distribution with M again fixed to 10. In the sequel, we will not in general make the distinction between a (TD)GMM and a (TD)GMM$^+$ noting that context makes it obvious which one applies (e.g., Simulations A, B, and D feature duration distributions with infinite support and therefore utilize the (TD)GMM$^+$ whereas Simulation C features finite support duration distributions and therefore uses the (TD)GMM).

We evaluate our simulations in three ways. First, we look at the classification error of the various methods as well as that of the Bayes Rule which uses the true marginal probabilities $\mathbb{P}(Y_t|X_{1:T})$ and average over the 200 test observations and 100 repetitions. Second, we look at the classification error relative to the Bayes Rule. Since the Bayes Rule is the "gold standard", it is free of the noise in the Y_t and gives optimal classifications. Therefore, it is informative to see how well the various methods replicate it. Finally, we compare the probability estimates of the various methods and report the RMSE of the probability averaged over the 200 test observations and 100 repetitions. We also examined other probability loss functions including log loss and exponential loss and all yielded qualitatively similar results.

6.2 Simulation One Results

Simulation one is deliberately simple (one covariate, linear decision boundary) so that we can study how the model performs with variations in the sample size, time series structure, and noise level. This will lead to some interesting insights about model performance in real world settings.

6.2.1 Simulation 1A

Simulation 1A features a linear decision boundary and no time series structure. Hence, all the models considered are capable of fitting the truth. The results are given in Figure 6.4. A key feature evident in the first row is that, as the noise level σ increases, all methods converge to predicting the dominant class and giving probability estimates equal to the base rates: that is, the covariates are effectively useless when there is a large amount of noise. The second and third rows demonstrate that, for large samples, all methods replicate the Bayes Rule: as we move from the $n = 100$ plot towards the $n = 10000$ plot, the classification and probability errors are converging to zero for all noise levels.

An interesting feature of this simulation, particularly evident in the $n = 100$ plots for classification error relative to the Bayes Rule and for probability error, is efficiency of estimation. Since all models are capable of fitting the true distribution (because there is no time series structure), multinomial logistic regression performs by far the best since it is the simplest model and has the fewest num-

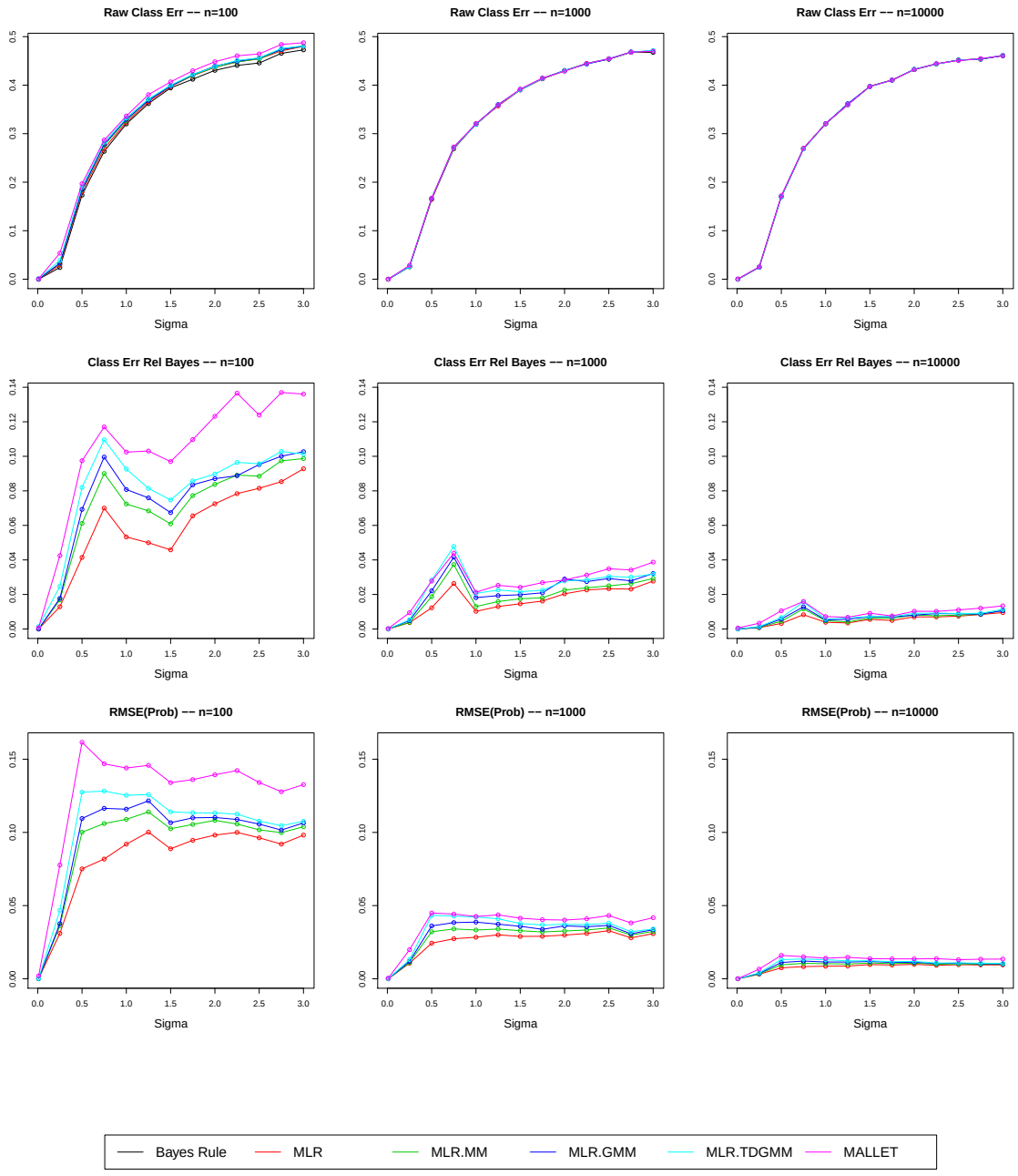

Figure 6.4: Results for Simulation 1A. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively.

ber of parameters. It is followed by the standard Markov model, then the GMM, then the TDGMM, and finally by MALLET, the exact ordering of model complexity (By complexity we mean number of parameters estimated. There is no fully nested structure here. That is, while GMMs, TDGMMs, and MALLET all nest first order Markov models, MALLET does not nest GMMs or TDGMMs nor do they MALLET. Of course, TDGMMs nest GMMs, they both nest standard first order Markov models, and all three nest the no time series model.). That said, the more complicated models (and particularly the variants of the Markov model) do not do much worse even with $n = 100$ suggesting that not much is lost by fitting a more complicated structure even when it does not exist. This is due to the fact that the duration distributions are parameterized in our model and therefore can be estimated efficiently.

6.2.2 Simulation 1B

For Simulation 1B, all models except multinomial logistic regression are capable of fitting the truth. This is without a doubt the most prominent feature of the results, which are plotted in Figure 6.5. It is also of note that the GMM and TDGMM are almost as efficient as the simple Markov model (which in this case is the correct model); MALLET lags only slightly behind.

The models appear to be converging on the true probabilities. However, when the noise level is high, they cannot quite mimic the Bayes Rule classifications even

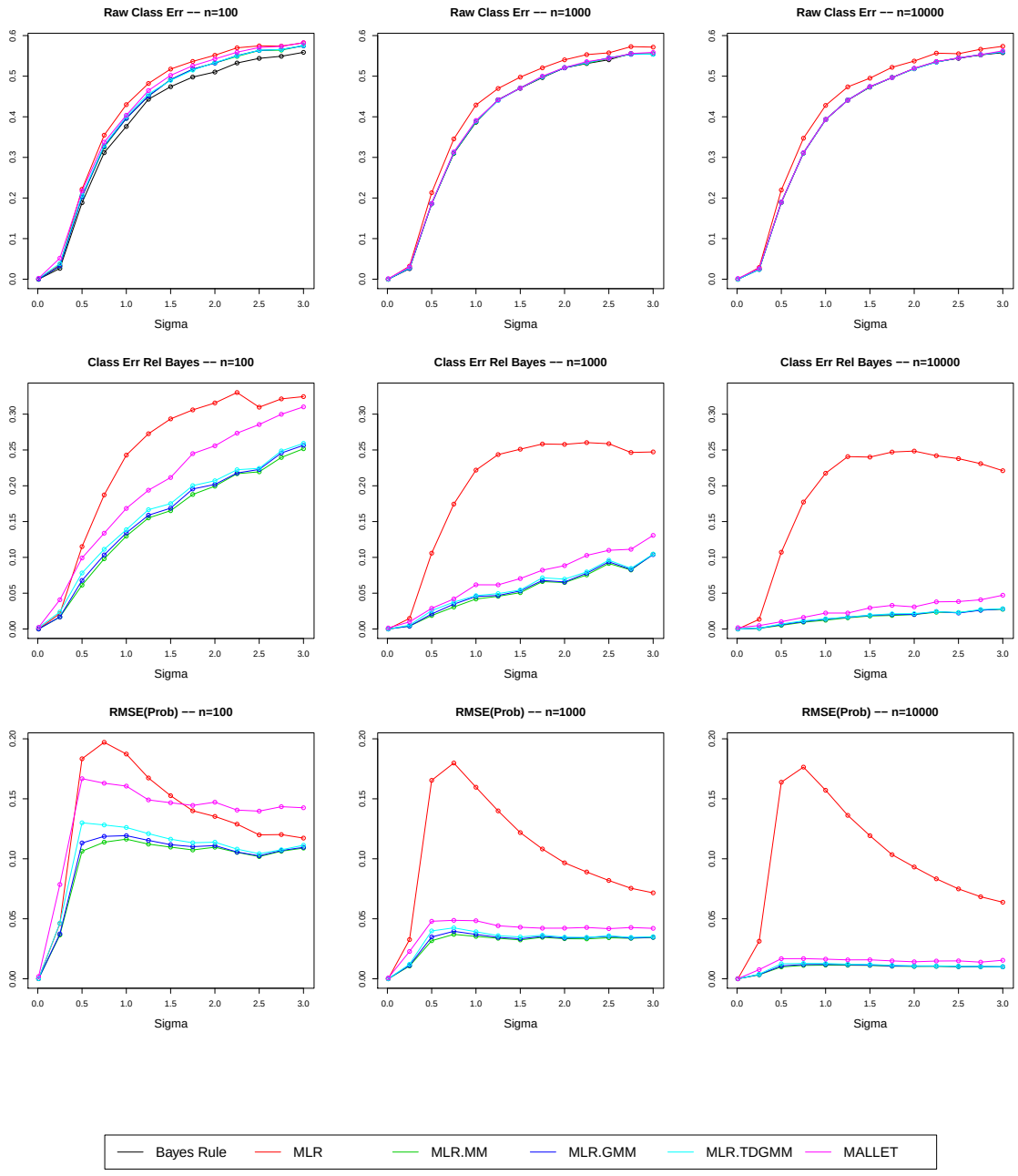

Figure 6.5: Results for Simulation 1B. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively.

with $n = 10000$ suggesting that massive amounts of training data may be required in very high noise settings, even when the model structure is as simple as a linear decision boundary in one dimension coupled with a simple Markov structure.

The trade-off between noise, model complexity, and training set size is interesting and difficult and the lower left panel demonstrates this. With fixed sample size $n = 100$, as the noise level increases, the incorrect logistic regression provides better probability estimates than MALLET, which is a complex model nesting the true structure. As $\sigma \to \infty$, logistic regression (and all models for that matter) will correctly estimate the true probabilities and hence the Bayes Rule classification (since the correct probabilities are the marginal base rates (i.e., the covariates are effectively useless with high enough noise)). Conversely, as $\sigma \to 0$, all models do well because the distribution of $Y_t|X_t$ places all probability on the dominant class (i.e., the covariates are effectively "oracles" when the noise is very low).

More interestingly, the more complicated variants of the Markov model (i.e., GMM and TDGMM) do almost as well as the standard one even with $n = 100$ suggesting that not much is lost by fitting a more complicated structure even when it does not exist.

6.2.3 Simulation 1C

A key feature of the results of Simulation 1C (Figure 6.6) is the fact that both the standard Markov model and MALLET (in addition to logistic regression) cannot

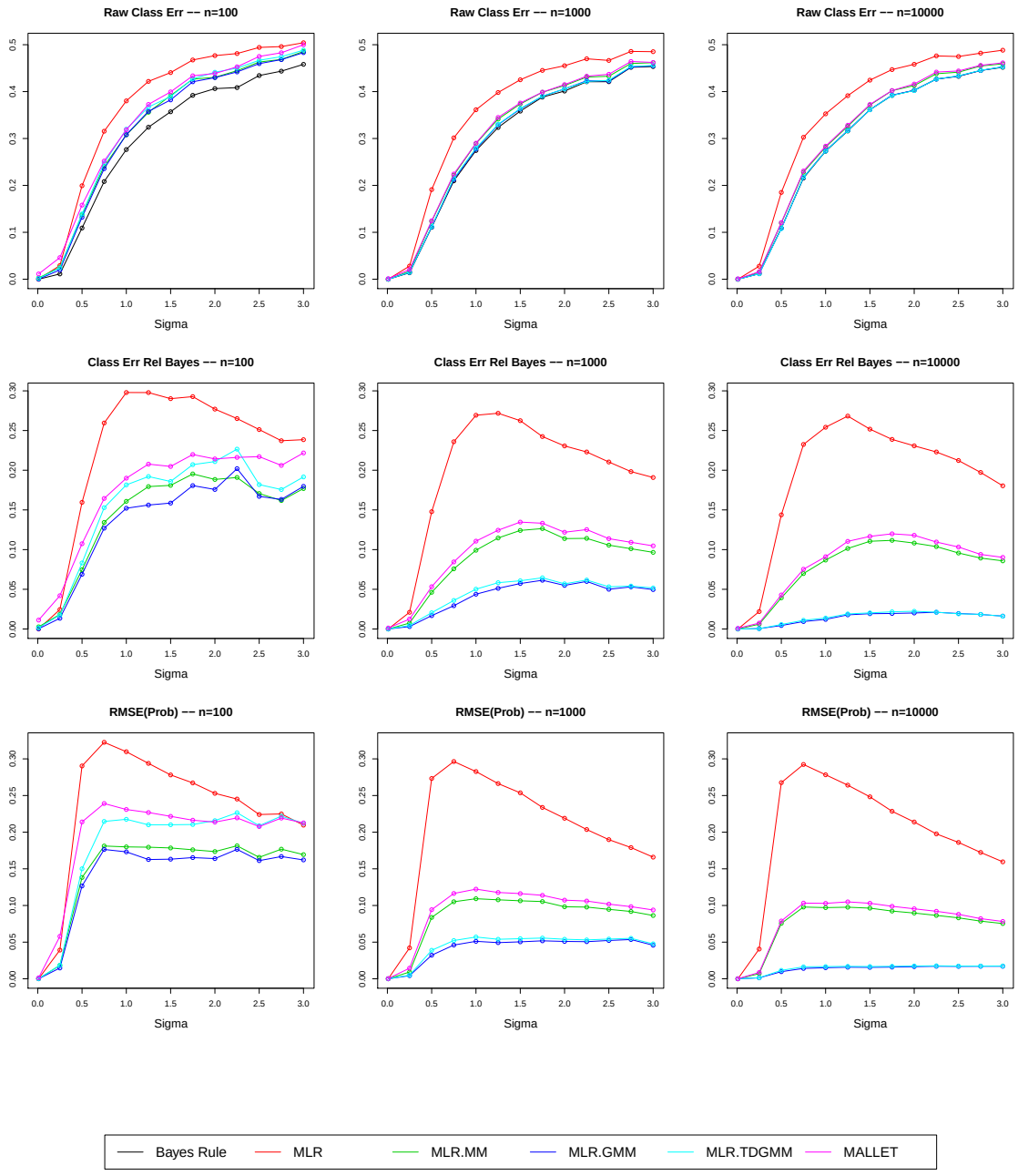

Figure 6.6: Results for Simulation 1C. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively.

estimate the true model. However, they do seem to be estimating the same thing and this provides improvement over a logistic regression which ignores the time series structure, particularly for moderate σ.

Again, the trade-off between model complexity and sample size proves interesting. For $n = 100$, the standard Markov is doing about as well as the GMM even though the former is false and the latter is true; on the other hand, the TDGMM which nests the truth is fairly uncompetitive. However, by $n = 1000$, both the GMM and the TDGMM seem to capture the true structure whereas the incorrectly specified models cannot. Also interesting is the fact that the incorrect models seem not to improve as n is increased from 1000 to 10000 whereas the two correct models do.

6.2.4 Simulation 1D

Most interesting and fruitful of all simulations are the results of Simulation 1D shown in Figure 6.7 where the true underlying model is a TDGMM$^+$. In terms of classification error, the correctly-specified TDGMM model does best for large sample sizes. However, even $n = 10000$ does not seem to be large enough for it to match the Bayes Rule for high noise levels. Again, the standard Markov model and MALLET seem to be estimating the same thing and they are fairly competitive. The GMM does not do very well, in this case because it is trying to estimate a single duration distribution for each state when here there are two (i.e., each state i

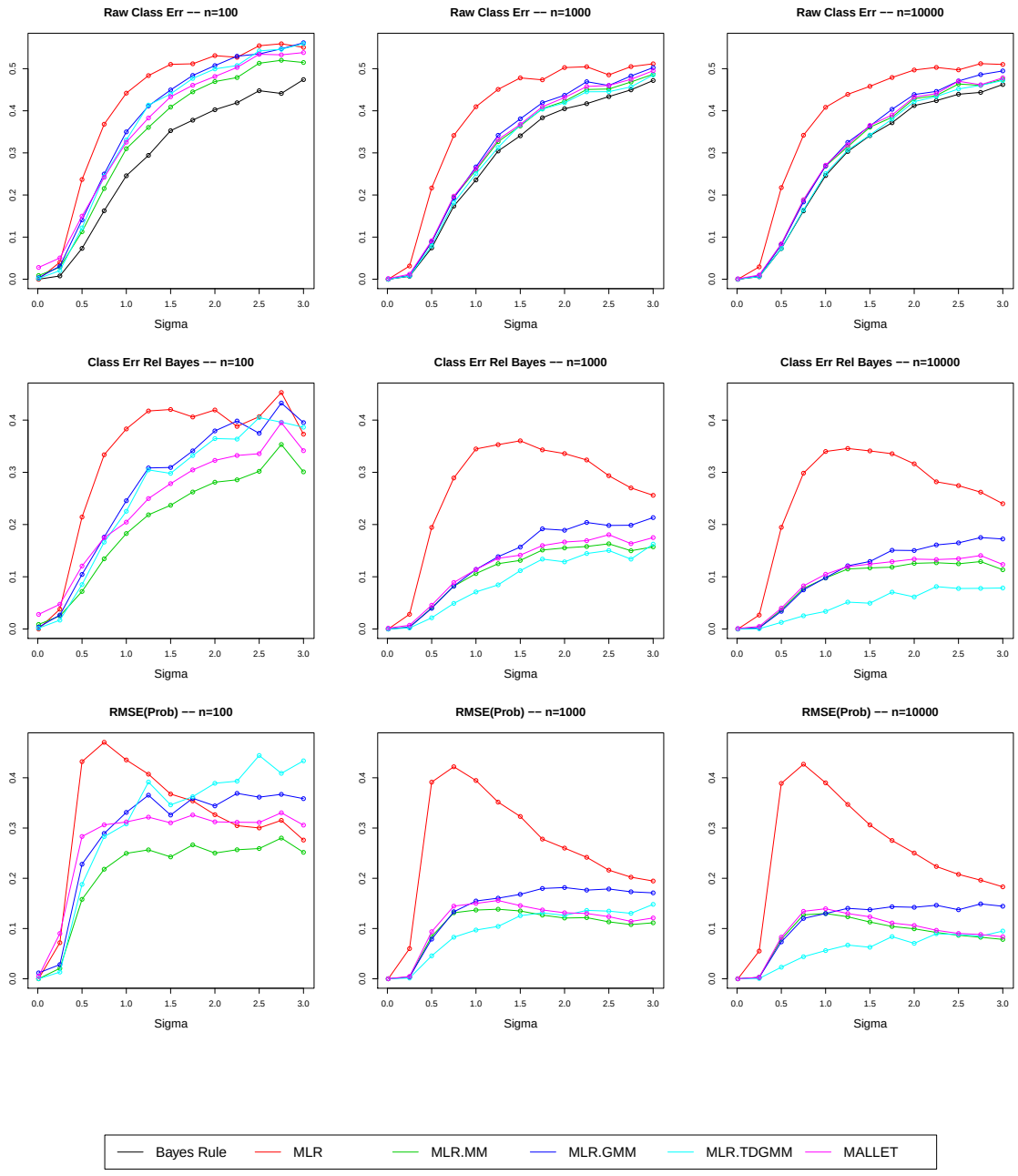

Figure 6.7: Results for Simulation 1D. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively.

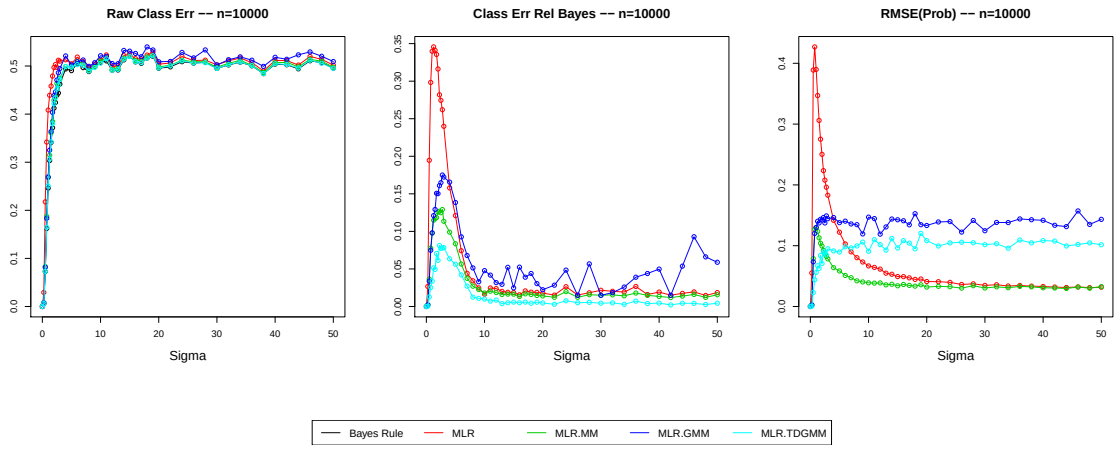

Figure 6.8: Results for Simulation 1D with Large σ. The first plot gives the Classification Error, the second plot the Classification Error Relative to the Bayes Rule, and the third plot the Root Mean Square Error of the Probability Estimates. In all simulations, the training set size was set to $n = 10000$.

has duration distribution which is actually a mixture of two distributions, reflecting the two other states j which the sequence previously was in before state i). While the Markov model does this (with the further restriction that the distribution is geometric), the simplified assumptions seem to allow it to perform better at least in this case.

The probability error plots are by far the most interesting. With large samples, the TDGMM appears to give the best results for a range of σ. However, eventually, once the noise level is high enough, simpler models such as the first order Markov model provide better probability estimates. This seems peculiar since the TDGMM is estimating the truth whereas these models are not.

This fact warranted further investigation so we fixed $n = 10000$, extended σ

out from 3 to 50, and estimated the models again (we excluded MALLET for these tests both because of the computational cost and the fact that it was effectively equivalent to the first order Markov model). These results are shown in Figure 6.8. For all σ, the TDGMM is still the best at matching the Bayes Rule classifications. However, for large σ, the first order Markov model and multinomial logistic regression produce equivalent probability estimates which are better than those produced by the TDGMM.

This is very peculiar because we saw the TDGMM produces the best probability estimates for low and moderate σ. Now, the TDGMM is composed of two sets of estimates: (i) the conditional class probability estimates of $\mathbb{P}(Y_t|X_t)$ and (ii) estimates of the time series structure (i.e., the duration distributions). The first set of estimates *are* the multinomial logistic regression estimates (i.e., these are fed into the Markov model and its variants). This suggests that for large σ, the TDGMM is making these estimates worse. This is compounded by another fact. Since the sample size is fixed at $n = 10000$ in these plots, the second set of estimates (of the time series structure) are just as good for low and moderate σ (when the TDGMM wins) as for high σ (when it loses). So, for large σ, the same time series estimates can degrade multinomial logistic regression probability estimates while, for low and moderate σ, they enhance them.

We conducted a further investigation of this phenomenon by considering various permutations on "oracle" models. Since the TDGMM is composed of two es-

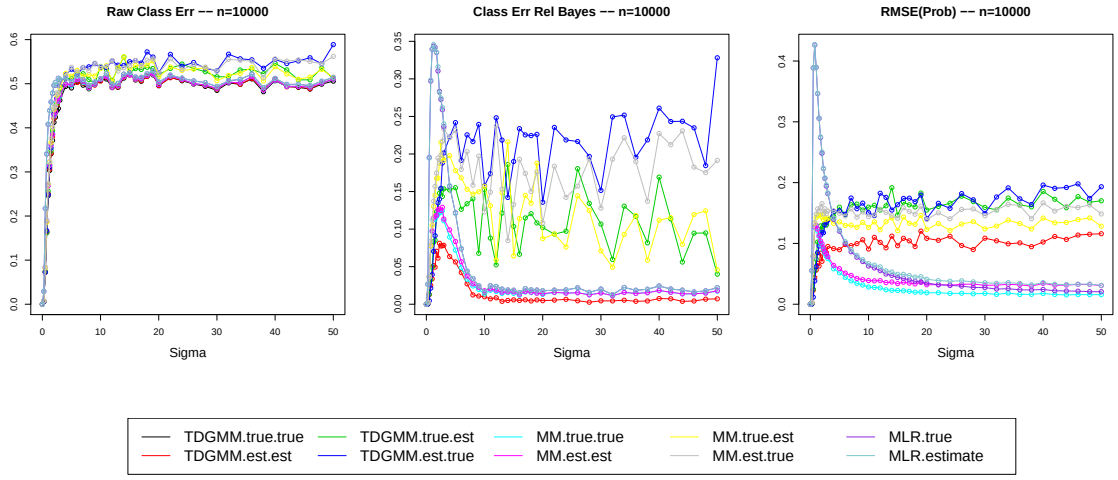

Figure 6.9: Results for Oracle Simulation 1D with Large σ. The first plot gives the Classification Error, the second plot the Classification Error Relative to the Bayes Rule, and the third plot the Root Mean Square Error of the Probability Estimates. In all simulations, the training set size was set to $n = 10000$. For a complete description of the models, see the main text.

timates, that of the conditional class probabilities and of the time series structure, we considered four versions of the TDGMM: (i) using the true probabilities for each (TDGMM.true.true, which is the true model or Bayes Rule in this case), (ii) using the true conditional class probabilities and estimates of the time series structure (TDGMM.true.est), (iii) using estimates of the conditional class probabilities and the true time series structure (TDGMM.est.true), and (iv) using estimates of both (TDGMM.est.est, a fully estimated TDGMM which we have labeled TDGMM and we have been considering in the previous plots). We did the same for the standard first order Markov model (there is no true first order Markov here since the model is TDGMM; however, the TDGMM induces a first order transition probability matrix on the sequence and this is what is titled the "true" Markov model time series structure here). Finally, we used two versions of multinomial logistic regression, the true model and the estimated one. These results are presented in Figure 6.9.

This first thing that is apparent is that the Markov model and TDGMM model which combine estimated probabilities and true probabilities perform by far the worst (i.e., the green, blue, yellow, and gray lines in the figure). Second, the doubly-estimated TDGMM provides the best classifications (almost matching those of the Bayes Rule or doubly-true TDGMM); however, for large σ, the doubly-true and doubly-estimated Markov model as well as the true and estimated multinomial logistic regression come close to it.

When we move to probability estimation, we see something similar to what we

saw in Figure 6.8: for large σ, the doubly-estimated TDGMM simply is not competitive. Simpler estimated models like the doubly-estimated Markov model and estimated logistic regression dominate (i.e., this is the equivalent to what was said of the results presented in Figure 6.8). More interesting is what happens to these simpler models. As σ gets large, the probabilities from the true multinomial logistic regression approach those of the "true" Markov model and likewise those of the estimated multinomial logistic regression approach those of the doubly-estimated Markov model. These latter two are quite close to the former two and would obviously equal them with infinite training data.

6.3 Simulation Two Results

Simulation two features a much more difficult decision boundary as well as eight covariates which behave identically regardless of the value of Y_t. These features, in tandem with the insights gleaned from simulation one about simpler, incorrect models performing better in noisy settings, suggest that the benefits of adding a time series approach may not be as dramatic for this setting or in the real world insofar as this simulation is representative of real world phenomenon.

In this simulation, however, there are two sources of "noise". The first is noise in the true sense (i.e., randomness) and corresponds to the decreasing prominence of the dominant class as one moves down the rows of Figure 6.1; this is the analogue of increasing σ in the first simulation. The other source of "noise" is not noise in

the statistical sense but rather is the uncertainty introduced by a difficult modeling phenomenon and the imprecise probability estimates which result from it: the checkerboard-like pattern of Figure 6.1 is simply difficult to fit.

6.3.1 Simulation 2A

Since there is no time series structure in this simulation, we see in Figure 6.10 that all the various methods that use a given classifier (e.g., Random forests, a first order Markov model trained discriminatively by Random forests, GMM trained discriminatively by Random forests, and TDGMM trained discriminatively by Random forests) perform almost identically. This means that not much is lost by fitting the more complicated structure even when it does not exist (as we will see in the sequel, this may be as much a curse as a blessing).

Other than that, it seems the linear methods (MLR, MLR.MM, MLR.GMM, MLR.TDGMM, and MALLET) fail completely. Even when MALLET is augmented with higher order terms (i.e., MALLET2 and MALLET3), these methods fail. On the other hand, the tree-based methods (Random forests, AdaBoost, and their various Markov model counterparts) appear to perform fairly equally at classification and, with sufficient training data, provide reasonable error rates relative to the Bayes Rule (all methods perform well relative to the Bayes Rule for the highest noise setting because the Bayes Rule always chooses the class with highest marginal base rate regardless of X_t and all methods do indeed do this).

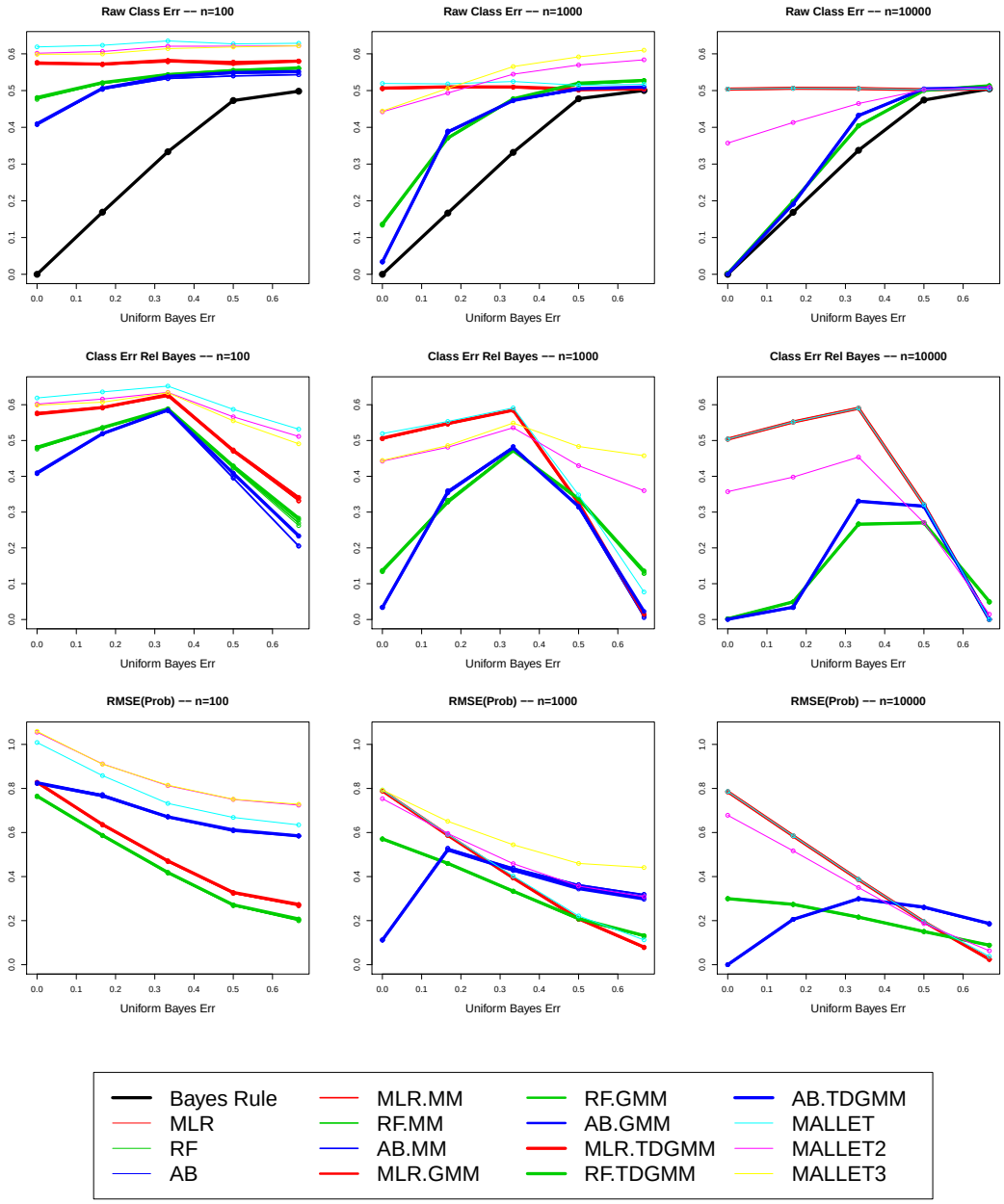

Figure 6.10: Results for Simulation 2A. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively. The x-axis is a general measure of the noise level and gives the Bayes Error rate that would be encountered in a non-sequential learning task with the covariate emission distribution given in Figure 6.1 and where each of the three classes has equal marginal base rate 1/3.

As suggested in Chapter 3, AdaBoost tends to push probability estimates towards zero or one. Hence, in the zero noise setting, it provides good probability estimates. However, when there is noise, Random forests produces superior probability estimates. In the highest noise setting and with sufficient training data, all methods other than AdaBoost provide good estimates of the probabilities because the true probabilities are the base rates; AdaBoost nonetheless overfits.

In Figure 6.11 we provide the same plots for various permutations of TreeCRF. We also provide our preferred model, RF.TDGMM, as a reference point. Our method dominates despite the number of variants of the TreeCRF methodology. Even though TreeCRFs nest the true model, they are not very successful at fitting it here.

6.3.2 Simulation 2B

Figure 6.12 shows the results for the simulation when the Y_t sequence exhibits an first order Markov structure. Again, there is a great deal of "clumping" for all the various methods that use a given classifier (e.g., Random forests, first order Markov model trained discriminatively by Random forests, GMM trained discriminatively by Random forests, and TDGMM trained discriminatively by Random forests). It seems of the two kinds of estimates entered into the models (i.e., the conditional class probability estimates and the time series estimates), the former play the dominant role. This is akin to what we saw with Simulation 1D where the simpler, incorrect

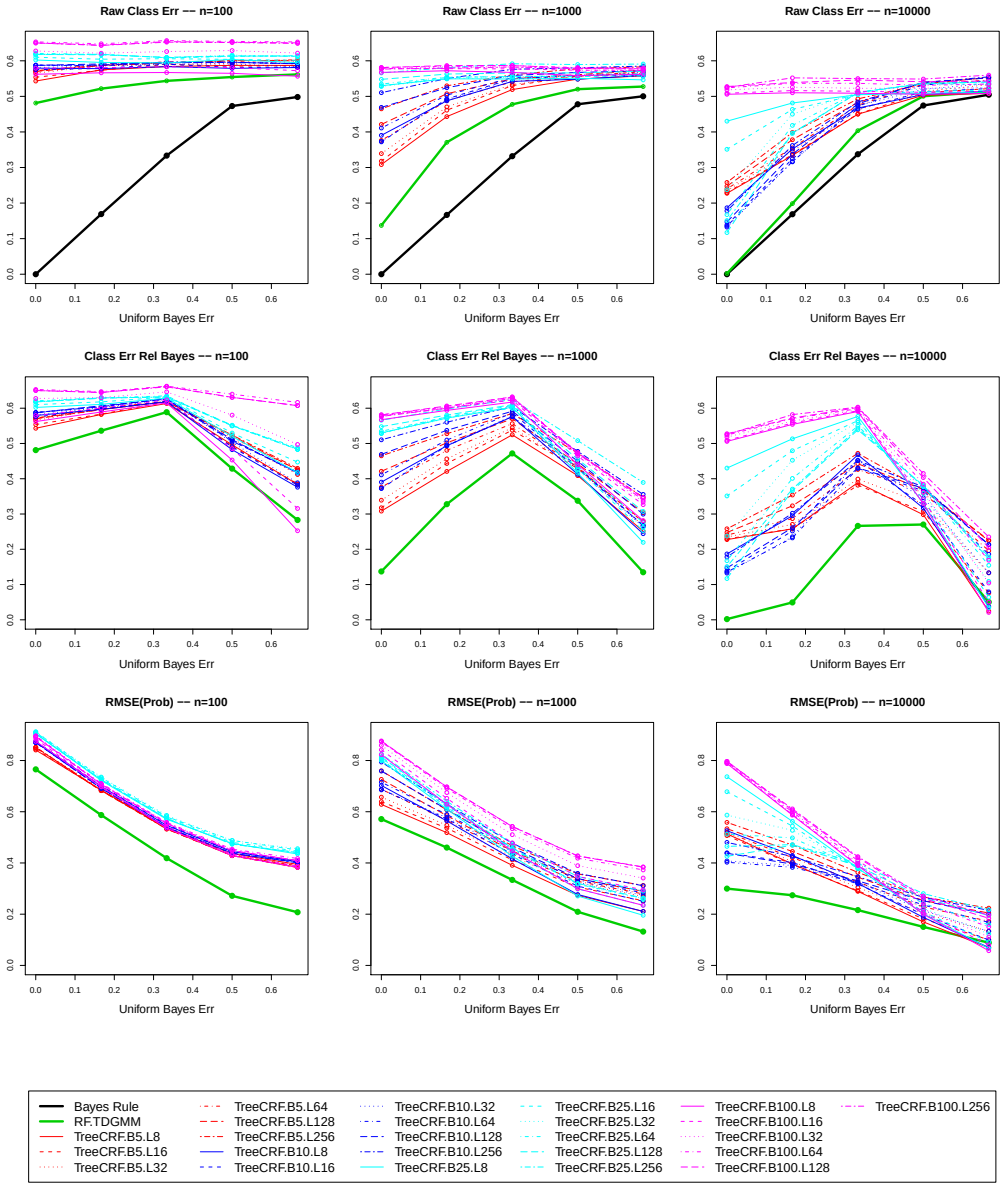

Figure 6.11: TreeCRF Results for Simulation 2A. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively. B denotes number of bins and L number of leaves. The x-axis is a general measure of the noise level and gives the Bayes Error rate that would be encountered in a non-sequential learning task with the covariate emission distribution given in Figure 6.1 and where each of the three classes has equal marginal base rate 1/3. RF.TDGMM from Figure 6.10 is provided in green as a point of reference.

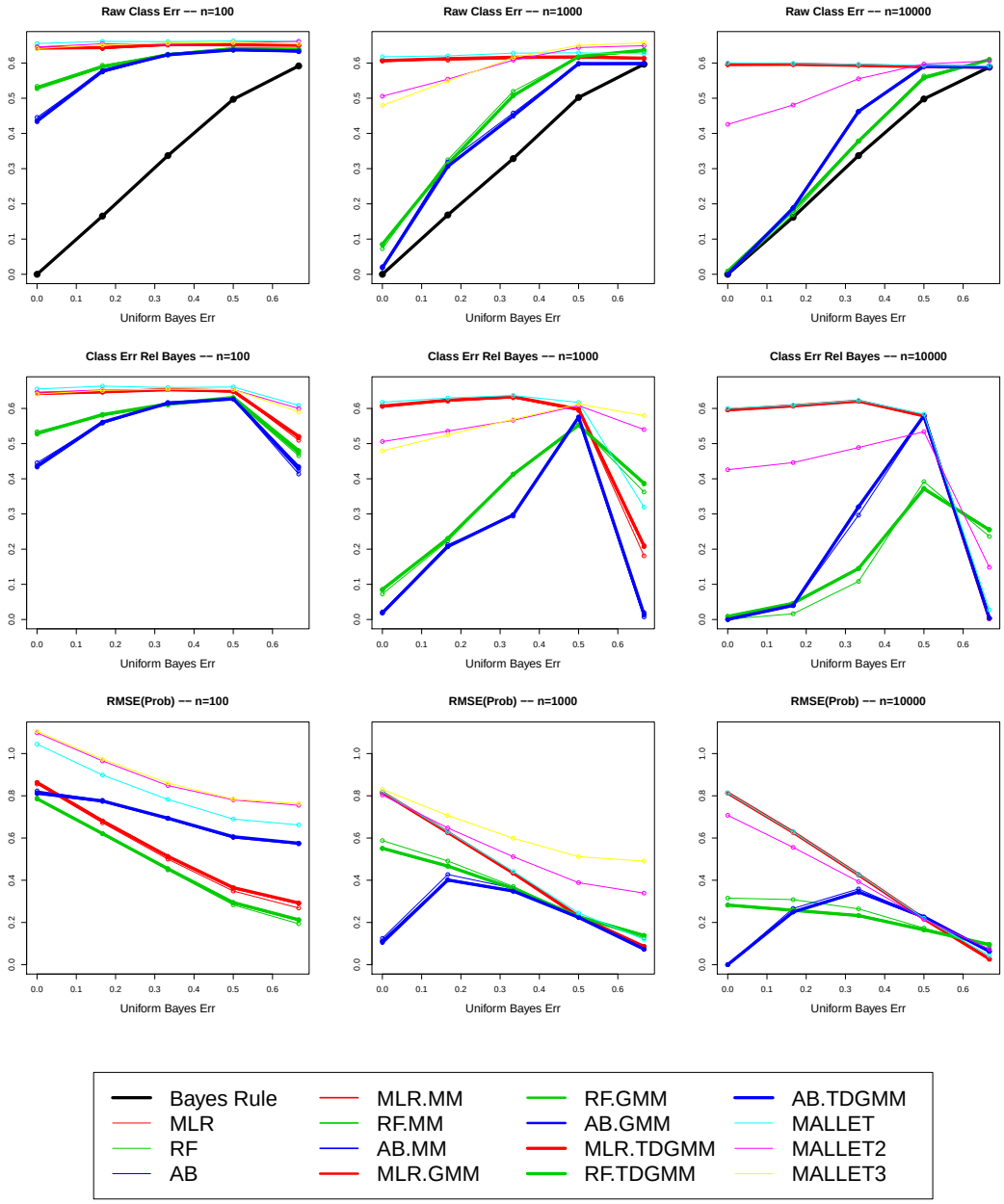

Figure 6.12: Results for Simulation 2B. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively. The x-axis is a general measure of the noise level and gives the Bayes Error rate that would be encountered in a non-sequential learning task with the covariate emission distribution given in Figure 6.1 and where each of the three classes has equal marginal base rate 1/3.

models were beating the correct model.

Again, the linear and augmented linear methods fail. The RF.MM appears to provide a small gain in classification for the large, $n = 10000$ sample size. However, this gain is accompanied by a small loss in terms of probability estimation. On the whole, however, Random forests and its variants appear to be the most successful.

An interesting feature is shown in the $n = 10000$ classification error relative to the Bayes Rule and probability estimation plots. AB.MM sticks out slightly from AB, AB.GMM, and AB.TDGMM. However, it sticks out much less than the corresponding RF.MM lines do from the RF, RF.GMM, and RF.TDGMM lines. There is a simple explanation for this. Since AdaBoost pushes probabilities towards zero and one, adding the time series probability model can accomplish little: the probability estimates are so far towards the extreme that they are insensitive to time series modification.

The TreeCRF results are given in Figure 6.13 and, again, our preferred method, RF.TDGMM, dominates. Again, TreeCRFs can nest the underlying structure of this simulation, but appear unable to provide good fits here.

6.3.3 Simulation 2C

In this simulation, where the underlying time series structure is GMM, we begin to notice greater variation among methods as seen in Figure 6.14. Again, the linear and augmented linear methods all perform more or less equally poorly.

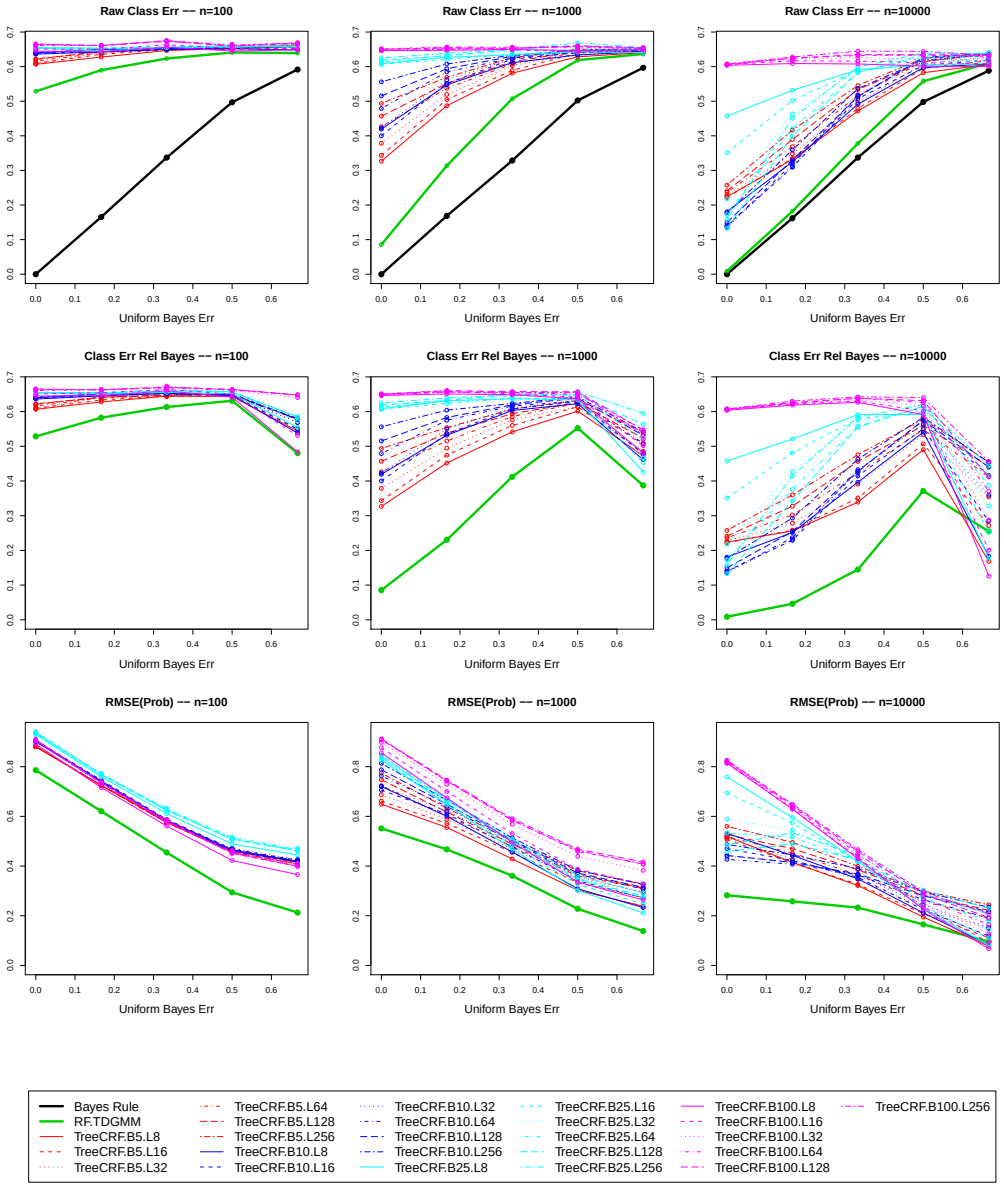

Figure 6.13: TreeCRF Results for Simulation 2B. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively. B denotes number of bins and L number of leaves. The x-axis is a general measure of the noise level and gives the Bayes Error rate that would be encountered in a non-sequential learning task with the covariate emission distribution given in Figure 6.1 and where each of the three classes has equal marginal base rate 1/3. RF.TDGMM from Figure 6.12 is provided in green as a point of reference.

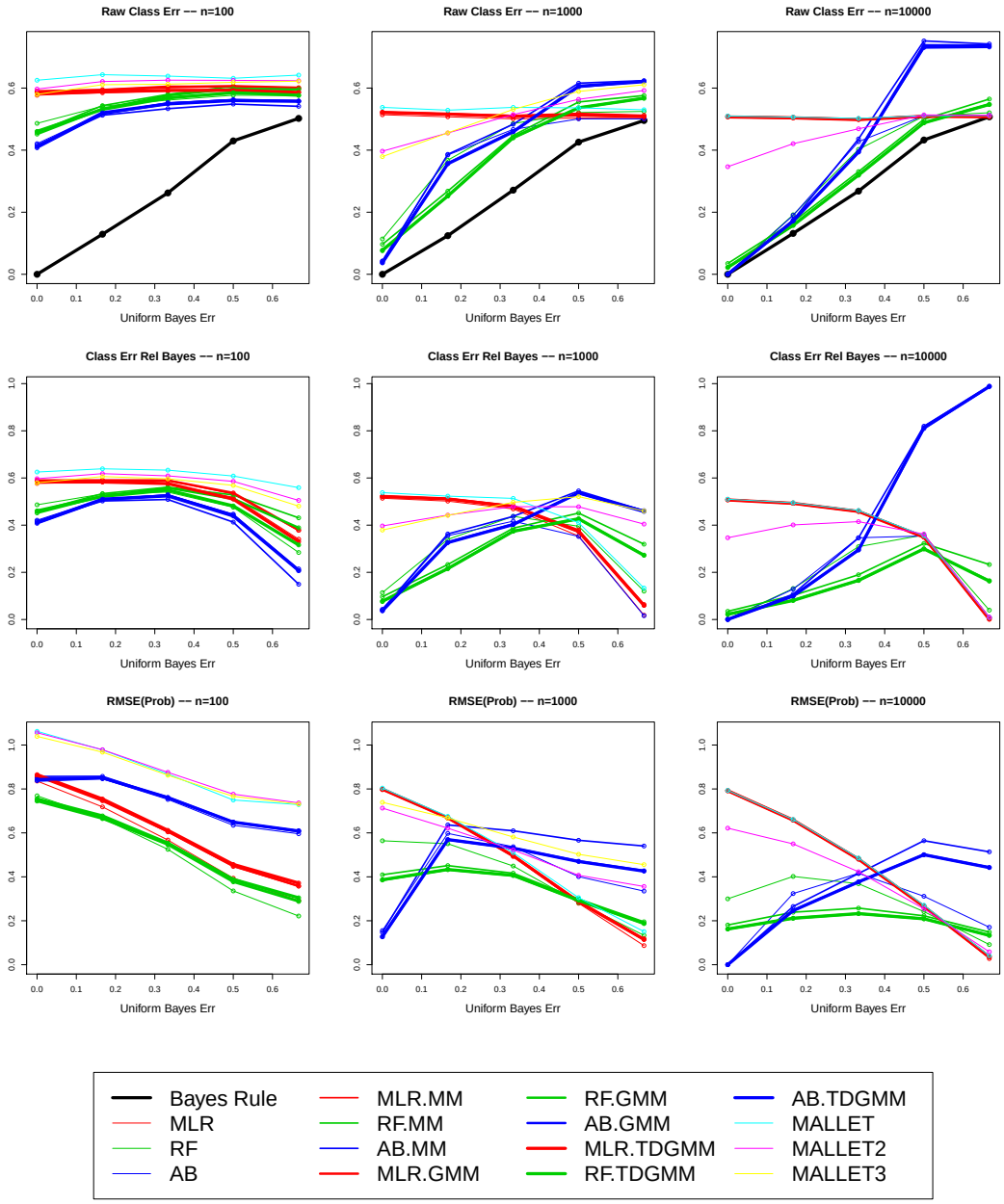

Figure 6.14: Results for Simulation 2C. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively. The x-axis is a general measure of the noise level and gives the Bayes Error rate that would be encountered in a non-sequential learning task with the covariate emission distribution given in Figure 6.1 and where each of the three classes has equal marginal base rate 1/3.

Interestingly, AdaBoost and its variants perform terribly as the noise level increases, even with large sample sizes. Again, due to the overfit probability estimates, the Markov variants of AdaBoost provide almost identical fits to plain vanilla AdaBoost.

The Markov variants of Random forests now perform differently than plain vanilla Random forests, providing moderate improvement in relative classification error and large improvements in probability error. Though the GMM and TDGMM are the only methods capable of nesting the true structure and though they do beat the first order Markov model, they do so by only slight amounts. That is, in the presence of such a difficult covariate pattern, the incorrect first order model provides substantial improvement over the non-sequential model but adding more complex time series structure such as that found in a GMM and TDGMM is not particularly helpful even when it exists and is well-estimated.

The TreeCRF results are given in Figure 6.15 and, again, our preferred method, RF.TDGMM, wins in almost all categories. Only with little data or very high noise are variants of the TreeCRF superior and even this is unclear due to the "snooping" resulting from running many permutations of TreeCRFs. At least in this case, however, the TreeCRF is unable to fit the true functional form of the data. Hence, it is not surprising it is defeated by a method which can.

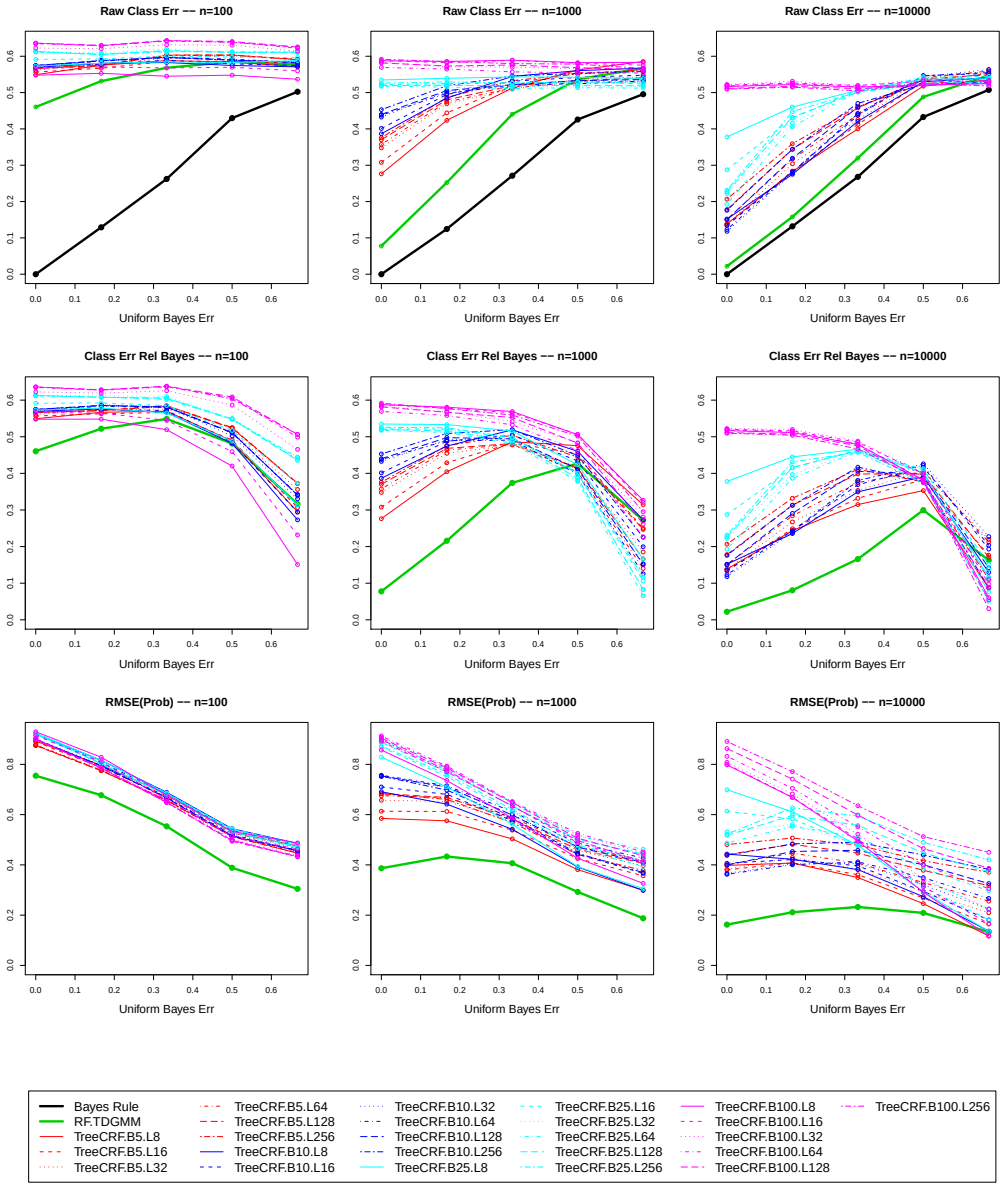

Figure 6.15: TreeCRF Results for Simulation 2C. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively. B denotes number of bins and L number of leaves. The x-axis is a general measure of the noise level and gives the Bayes Error rate that would be encountered in a non-sequential learning task with the covariate emission distribution given in Figure 6.1 and where each of the three classes has equal marginal base rate 1/3. RF.TDGMM from Figure 6.14 is provided in green as a point of reference.

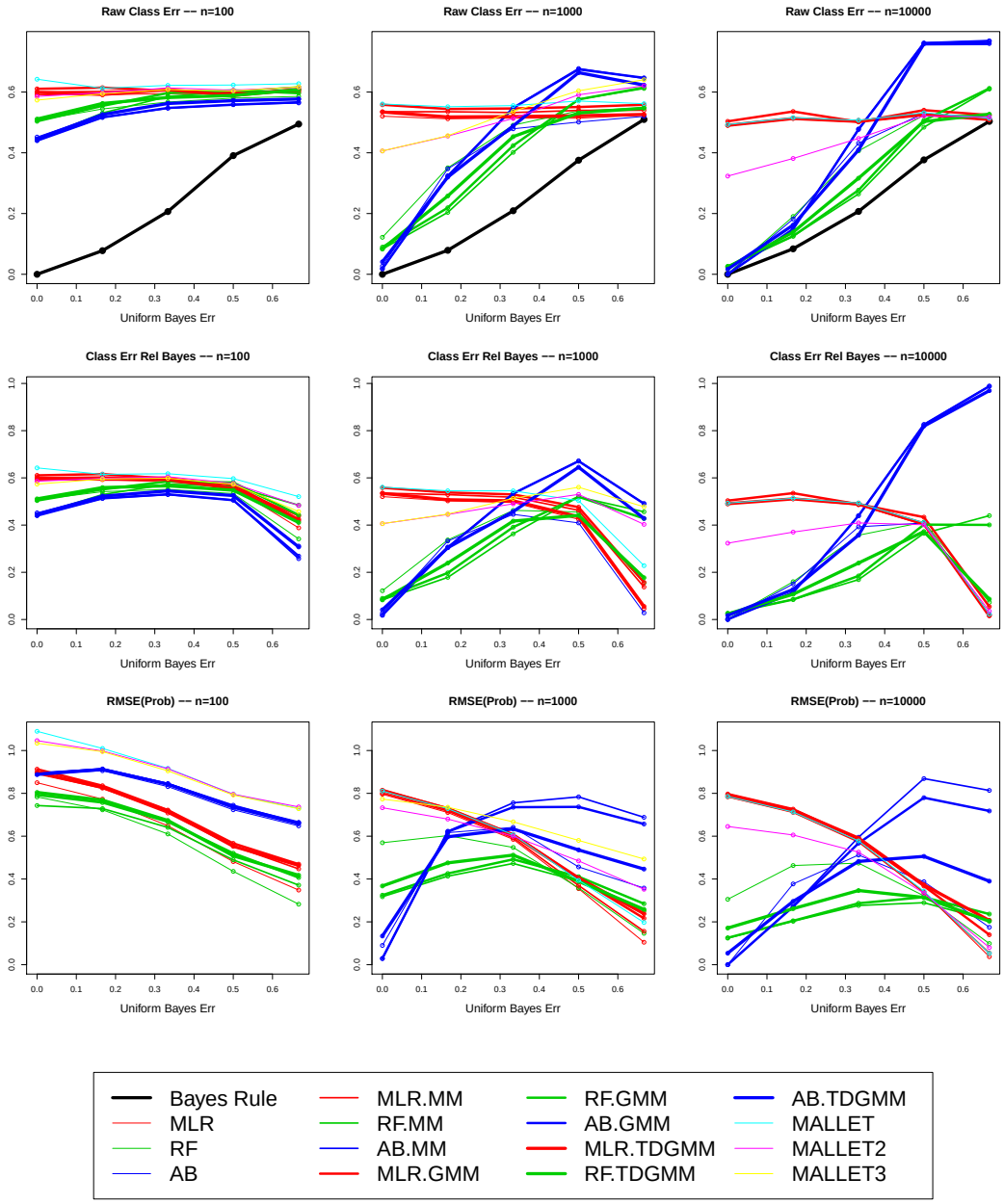

Figure 6.16: Results for Simulation 2D. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively. The x-axis is a general measure of the noise level and gives the Bayes Error rate that would be encountered in a non-sequential learning task with the covariate emission distribution given in Figure 6.1 and where each of the three classes has equal marginal base rate 1/3.

6.3.4 Simulation 2D

The fascinating results of this simulation appear in Figure 6.16. As the time series structure becomes increasingly complicated, we see less and less clumping of each method around its base classifier. There are substantial differences amongst the variations of AdaBoost and Random forests.

Again, the linear methods and their augmented counterparts fare poorly which is not surprising given the nature of the covariate emission distribution. Also, again, AdaBoost does poorly at classification. However, what is interesting is the distribution of its probability errors by method: plain AdaBoost performs similarly to or beats AB.TDGMM (which has the right time series structure) and these two are followed by AB.GMM and AB.MM respectively. Given the way AdaBoost overfits the probabilities, this degree of separation is somewhat surprising and shows the time series structure is having an effect. It also demonstrates the "garbage in, garbage out" principle: feeding the discriminative Markov model (or one of its more general variants) bad conditional class probability estimates might lead to even worse ones even if the times series estimates are reasonably good.

Again, Random forests and its variants perform best however, here, the time series structure performs more or less as anticipated. The non-sequential Random forest performs the worst, particularly on probability estimation. The three times series variants provide substantial improvement with the RF.MM and RF.TDGMM more or less tied, marginally beating out the RF.GMM. Now, it is not necessarily

surprising that the first order Markov model beats the GMM. Something similar happened in Simulation 1D where we saw that estimating a single geometric can beat estimating a single more complicated distribution when the true distribution is actually a mixture of complicated distributions (i.e., when both are wrong; here each state duration distribution is a mixture of two Beta Negative Binomials with Geometric Tails). Given what we have seen so far, it is also not surprising that the incorrect first order model is competitive with the correctly-specified TDGMM.

Finally, the TreeCRF results are given in Figure 6.17 and, again, our preferred method, RF.TDGMM, wins in almost all categories. Again, the TreeCRF is unable to fit the true functional form of the data so perhaps it is not surprising it gets beat by a method which can. That said, it is a bit staggering how uncompetitive this sophisticated method has been through these simulations.

6.4 Conclusion

In conclusion, the simulation study has revealed some very interesting facts about the performance of the PrAGMaTiSt, most pertaining to how variants of it (as well as other methodologies) perform as the noise level in $Y_t|X_t$ varies. In particular, for very high noise settings or settings where the conditional distribution of $Y_t|X_t$ is very difficult to model, the covariates become rather useless for prediction and the base rates of the classes yield good classifications and probability estimates, even in the presence of time series structure. This has a further implication: since all

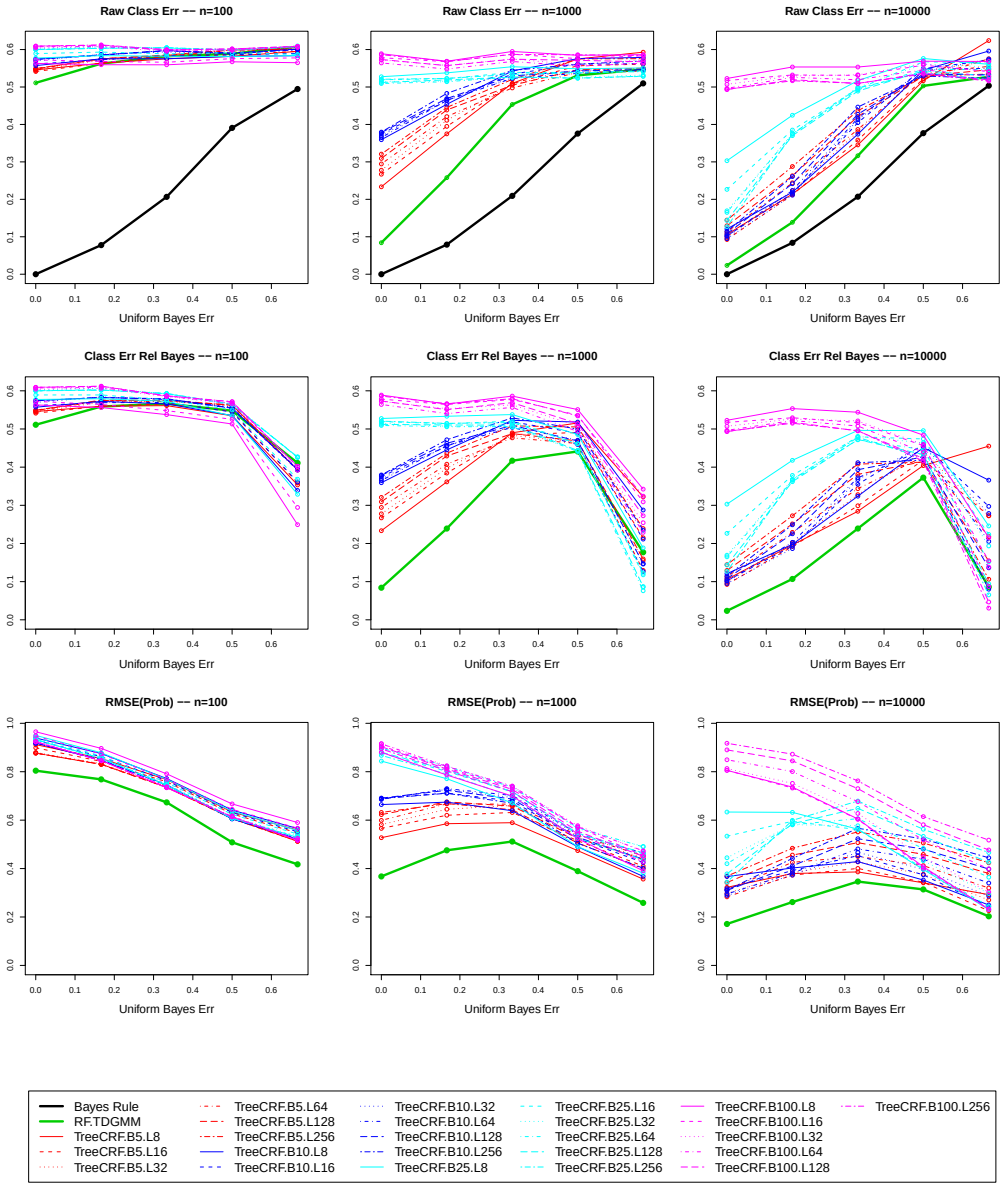

Figure 6.17: TreeCRF Results for Simulation 2D. The first row gives the Classification Error, the second row the Classification Error Relative to the Bayes Rule, and the third row the Root Mean Square Error of the Probability Estimates. The columns show results for training set size of $n = 100$, $n = 1000$, and $n = 10000$ respectively. B denotes number of bins and L number of leaves. The x-axis is a general measure of the noise level and gives the Bayes Error rate that would be encountered in a non-sequential learning task with the covariate emission distribution given in Figure 6.1 and where each of the three classes has equal marginal base rate 1/3. RF.TDGMM from Figure 6.16 is provided in green as a point of reference.

methods will perform more or less the same in high noise settings, simple methods will tend to dominate with finite data because they can provide better estimates. On the contrary, in very low noise settings, all methods also perform the same regardless of the time series structure because $Y_t|X_t$ puts almost all of its mass on one class. Hence, at the extremes, simple methods, even ones which lack time series structure, seem best.

The A and B simulations showed that we do not lose much with a moderate amount of data if we fit GMMs and TDGMMs when the truth either has no time series structure or is first order Markov. However, the C and D simulations showed a contrary result: with sufficient noise, complexity of $Y_t|X_t$, or complexity of the time series structure, the first order model can perform as well as or even beat the GMM or TDGMM even when the latter is true. Hence, there is a delicate tradeoff and sometimes one will want to fit a model which one knows to be incorrect.

Finally, we saw that AdaBoost in general performed poorly and, due to the fact that it overfits the probability estimates, it was generally insensitive to variants of the Markov model. Random forests was in general very competitive and probably the best method overall when combined with various Markov models. Finally, TreeCRF was not particularly competitive on these simulations.

Chapter 7

Sleep Data

7.1 Introduction

In Chapter 1, we noted that scientists are interested in the genetic bases for sleep and use mice as a model organism for this purpose. The existence of various strains and the ease of breeding knock-out mice means mice can differ markedly in both sleep behavior and prevalence of sleep disorders. In particular, sleep researchers are interested in which genes contribute to wakefulness versus sleep and, within sleep, REM sleep versus non-REM sleep. Sleep scientists are especially interested in the REM state of sleep which occurs much less frequently than either non-REM sleep or wakefulness.

This is important because approximately 40 million Americans are afflicted with various sleep disorders such as insomnia, sleep apnea, and narcolepsy. As knowl-

edge of these disorders grows amongst the populace, sleep medicine is becoming an increasingly important field of medical inquiry.

Unfortunately, large-scale sleep studies of mice are not currently feasible because the gold standard methodology for the study of sleep behavior–manual scoring of 10-second sleep epochs as REM, NREM, and WAKE based on EEG and EMG recordings–is invasive, expensive, and time-consuming. Furthermore, manual scoring is both internally and externally inconsistent: different scorers can disagree by as much as 8% overall and up to 15% within the important REM sleep stage and even the same scorer frequently disagrees with himself when he revisits the data at different times.

Initial forays into replacing the manual scoring process with an automated process based on video data have had only limited success: they can differentiate wakefulness from sleep but have no power to detect REM versus NREM (Pack *et al.*, 2007). Nevertheless, subtle, if noisy, signal does exist in the data as shown in Figure 1.2 and there is hope that more sophisticated methods will be able to accomplish this discrimination.

7.1.1 Data Collection

One inbred strain of male mice was used in this study: C57BL/6J (n=8), age: 10-12 weeks, weight: 18-23 gm., purchased from Jackson Laboratory, Inc. (Bar Harbor, ME). Mice were individually housed in Plexiglas cages (4" wide x 8" long x 12" high)

and maintained on a 12-hour light/dark cycle (lights on 0700; 80 Lux at the floor of the cage) in a sound attenuated recording room, temperature 22-24 °C. Food and water were available ad libitum. Animals were acclimated to these conditions for 10-14 days before beginning any studies. All animal experiments were performed in accordance with the guidelines published in the NIH Guide for the Care and Use of Laboratory Animals and were approved by the University of Pennsylvania Animal Care and Use Committee.

Mice were implanted with EEG/EMG electrodes under deep anesthesia (i.p. injection of Ketamine (100 mg/kg)/Xylazine (10 mg/kg)). For EEG recordings, three stainless steel miniature screws (0-80 x 1/16, Plastics One, Inc., Roanoke, VA) were placed epidurally in the following locations: (1) right frontal cortex (1.7 mm lateral to midline and 1.5 mm anterior to Bregma), (2) right parietal cortex (1.7 mm lateral to midline and 1 mm anterior to lambda), and (3) a reference electrode over the cerebellum (1 mm posterior lambda on the midline). Two EMG electrodes were sutured onto the dorsal surface of the nuchal muscles immediately posterior to the skull. All leads from the electrodes were connected to an 8-pin plastic connector/pedestal (Plastics One, Inc., Roanoke, VA) and then bonded to the skull with dental acrylic. After the bonding agent cured, the animals were connected to our signal amplifier system using a connecting cable and swivel-contact (Plastics One, Inc., Roanoke, VA) mounted above each cage. All mice had a 10-14 day post surgery recovery and habituation period before beginning any recording.

EEG and EMG signal were amplified using the Neurodata amplifier system (Model M15, Astro-Med, Inc., West Warwick, RI). Signals were amplified (2000x) and conditioned using the following settings for EEG signals: low cut off frequency (-6dB), 0.3 Hz and high cut-off frequency (-6dB), 30 Hz; for EMG signals: low cut-off frequency (-6dB), 10 Hz and high cut-off frequency (-6dB), 100 Hz. Signals were digitized at 100 Hz. All data were acquired and analyzed using Gamma software (Astro-Med, Inc., West Warwick, RI) and converted to European Data Format (EDF) for manual scoring and analysis in the Somnologica science software (Embla, Inc., Denver, CO).

WAKE, NREM, and REM sleep were manually scored in 10-second epochs during 24-hour baseline recordings. Stages were determined as follows: epochs were scored as wake when the EMG amplitude ranged from activity slightly higher than baseline during quiet wakefulness to higher amplitude activity during exploratory behavior. EEG amplitude was low with frequencies mostly above 10 Hz. NREM was characterized by high amplitude delta (1-4 Hz). EMG was constant with low amplitude activity. REM was characterized by low amplitude rhythmic theta waves (6-9 Hz) with the EMG remaining at baseline levels.

Twenty-four hours of data divided into 10-second epochs implies 8,640 epochs for each of the eight mice[1]. Thus, we have a total of 69,120 epochs manually scored as REM, NREM, or WAKE by trained technicians examining EEG and EMG waves.

[1]Our mice are named M1, M2, ..., M9. M7 died after surgery and therefore we do not have data for him, thus bringing the total number of mice to eight.

Figure 7.1: One frame of video data with an ellipse imposed by the tracking program that is used to calculate size, aspect ratio, and velocity of the mouse. Subtle differences between the three states can be detected visually suggesting some hope for the endeavor of automated sleep scoring.

In addition to this, we have video recordings captured at 10 frames per second. Thus, for a given epoch, there are 100 corresponding frames of video data, an example of which is given in Figure 7.1. Tracking software is used to calculate six numerical features X_t: the within-epoch mean and standard deviation of velocity, aspect ratio, and size of the mouse (where the mouse is approximated by a tracking ellipse). For velocity and size, we used the natural logarithms of the means and standard deviations rather than the raw ones as covariates. We also had one binary feature which indicates whether or not the light in the cage was turned on (lights were on from 7AM - 7PM).

Table 7.1: Summary Statistics By Sleep State

Sleep State	Fraction of Time	Number of Bouts	Average Duration
NREM	44.00%	1,998	15.19
REM	4.83%	451	7.39
WAKE	51.17%	1,904	18.53

Table 7.2: Summary Statistics By Conditional Sleep State

Sleep State	Fraction of Time	Number of Bouts	Average Duration
NREM->REM	4.87%	446	7.45
WAKE->REM	0.01%	5	1.80
REM->NREM	3.10%	101	20.90
WAKE->NREM	41.35%	1,895	14.88
NREM->WAKE	48.08%	1,548	21.18
REM->WAKE	2.59%	350	5.04

7.2 Exploratory Data Analysis

7.2.1 Introduction

In Figure 1.2, we presented evidence that there was some marginal signal in X_t useful for predicting the three classes of Y_t. In this section, we continue exploring our data but focus on the Y_t themselves. To give an overall sense of the data, we first computed some summary statistics for the three sleep states (REM, NREM, and WAKE) and present them in Table 7.1. As mentioned above, REM sleep is a very rare state occupying less than 5% of all epochs. Furthermore, it has many fewer bouts and a lower average bout duration.

Taking inspiration from the distinction between a GMM and a TDGMM, we computed the same set of summary statistics across all mice for the three sleep states conditional on the previous state. These are given in Table 7.2 and reveal some interesting differences. For REM, almost all of the bouts come from NREM. This is actually a biological necessity as transitions from WAKE to REM are more or less impossible[2]. For NREM, the bouts seem to be longer when entered into from REM as opposed to WAKE whereas WAKE bouts tend to be longer when entered into from NREM rather than REM.

Since these summary statistics suggest a difference in state duration depending on the previous state, we examined the entire distribution of bout durations conditional on the previous state via Q-Q plots. In Figure 7.2, we show the Q-Q plots for the three states along with null bands formed by taking non-parametric bootstrap samples which permute the true labels. As can be seen, for REM and WAKE (i.e., the upper left and lower left plots), the black Q-Q lines depart from the gray regions for large portions of the plot allowing us to reject the null hypothesis of same bout duration regardless of the previous state. For NREM, the Q-Q line teeters on the edge of the null bands and even departs from it briefly thus suggesting the null hypothesis is false.

Another feature evident in these plots is the length of the bout durations, with

[2]They are indicative of sleep disorders, incorrectly scored epochs, or are so-called DREM bouts. DREM is a direct transition from WAKE to REM that occasionally occurs in wildtype mice. Such episodes occur almost exclusively during the lights on period and are the result of brief awakenings interrupting a sustained period of REM sleep (Fujikia *et al.*, 2009).

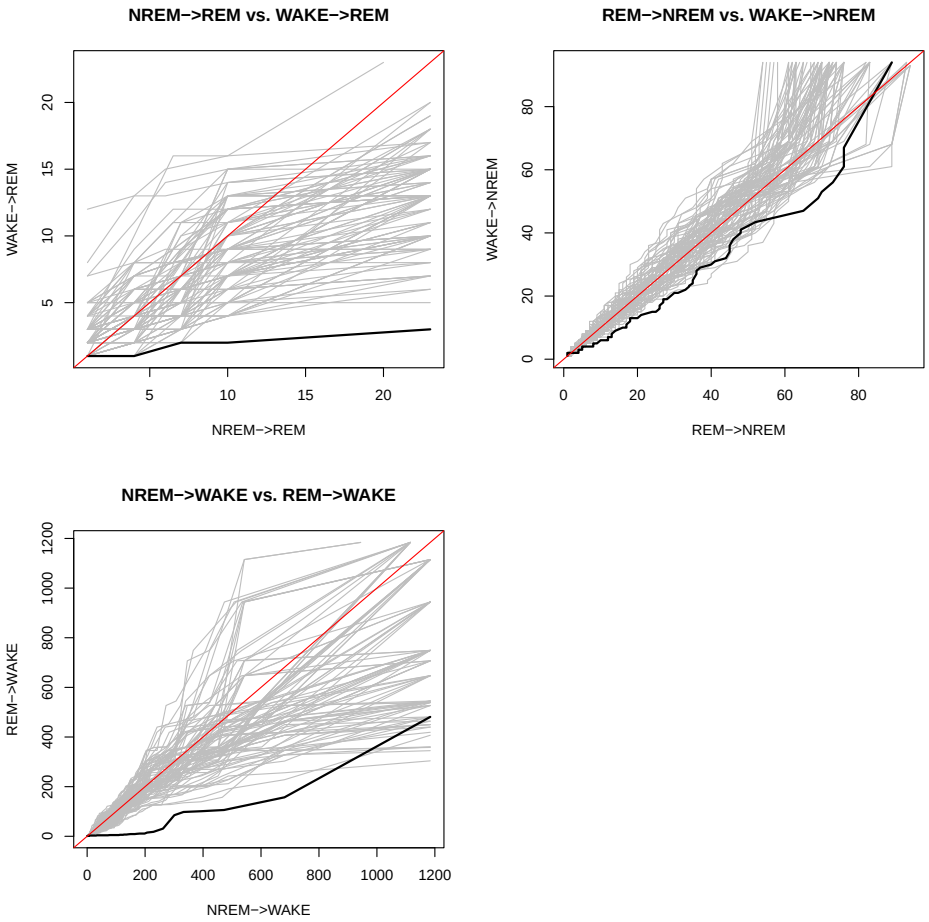

Figure 7.2: Bout duration Q-Q plots for each sleep state conditional on the previous state. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from non-parametric bootstrap samples which permute the true labels.

some bouts lasting many times the length of the mean durations presented in Tables 7.1 and 7.2. In fact, mouse sleep bout durations are distributed according to a "spike and slab" distribution which feature (i) most of the probability mass at one or two epochs and (ii) long right tails (McShane *et al.*, 2010). These features actually make the summary statistics presented in Tables 7.1 and 7.2 misleading (or at least highly variable) because they (in particular, the number of bouts and average bout duration) can be tremendously influenced by one or two aberrantly long bouts. Thus, the Q-Q approach presented here is much more appropriate.

7.2.2 Mouse-to-Mouse Variation

The conditional approach appears to be superior to the unconditional approach. However, the plots we considered combined data for all mice. It is possible that the mice themselves display vastly different behavior and that this could be the cause of the results above. In order to see whether this was the case, we made Q-Q plots for each mouse against each other mouse for each of the six conditional states listed in Table 7.2. For reasons of space, we do not include all such plots; however, we show them for mice one, three, and nine for every state (except, for obvious reasons, REM from WAKE) in Figures 7.3, 7.4, 7.5, 7.6, 7.7.

Generally, the null hypothesis of equal bout duration distributions cannot be rejected. While the black lines occasionally depart from the gray null regions, these departures tend to be rare. This fact also held up when looking at the plots for all

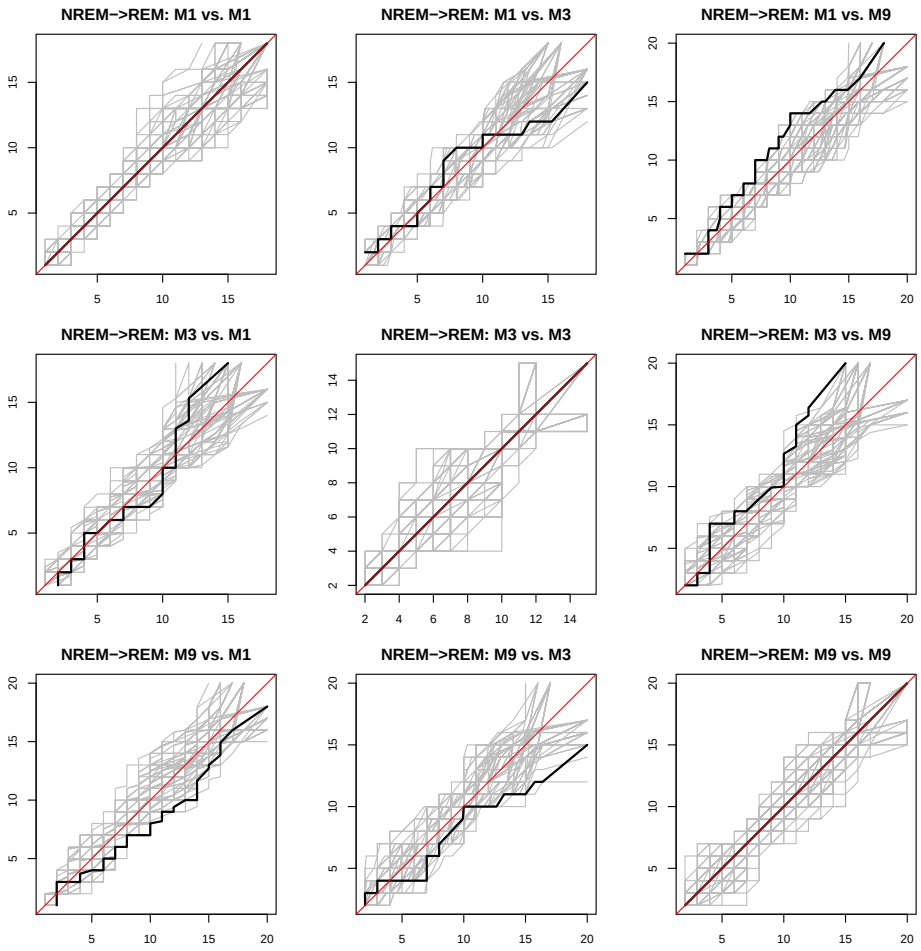

Figure 7.3: NREM->REM bout duration Q-Q plots for three mice. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from non-parametric bootstrap samples which permute the true labels.

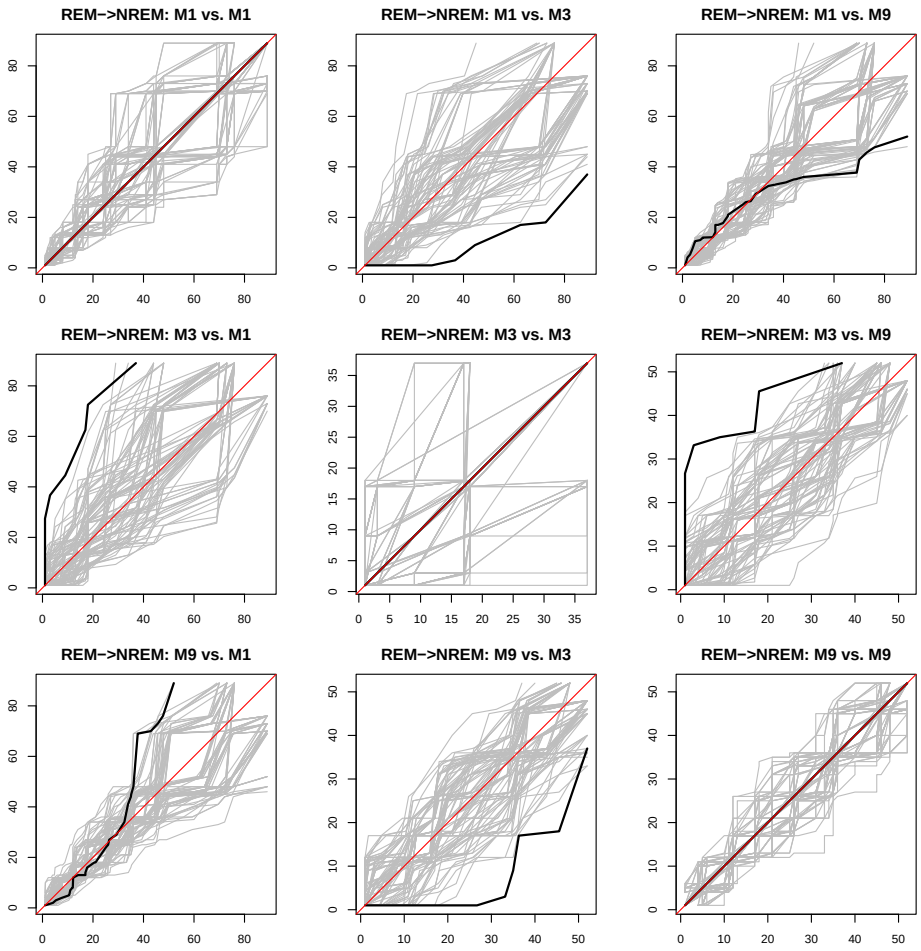

Figure 7.4: REM->NREM bout duration Q-Q plots for three mice. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from non-parametric bootstrap samples which permute the true labels.

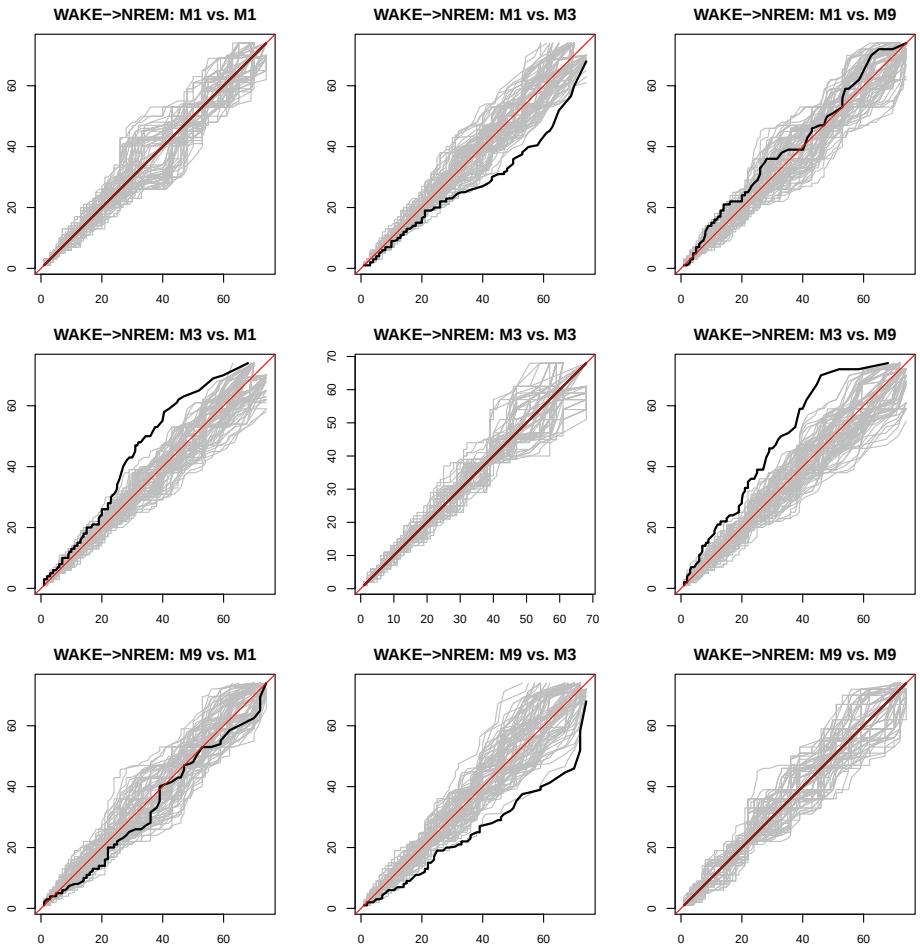

Figure 7.5: WAKE->NREM bout duration Q-Q plots for three mice. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from non-parametric bootstrap samples which permute the true labels.

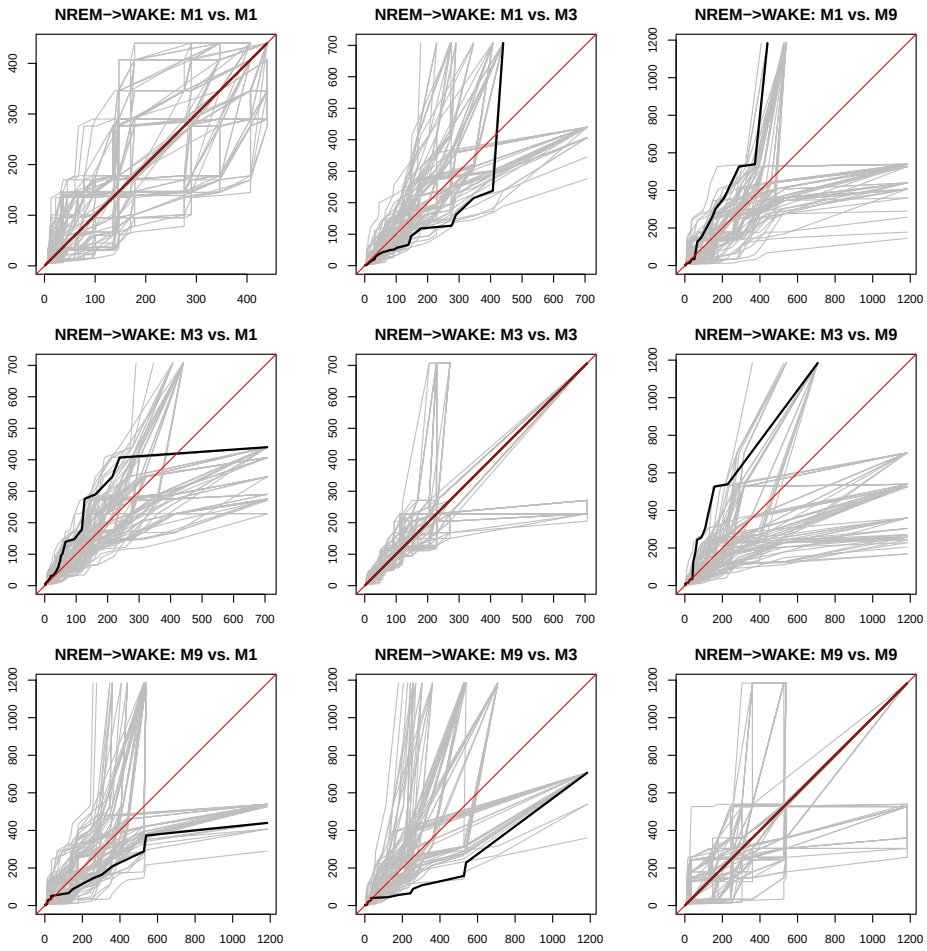

Figure 7.6: NREM->WAKE bout duration Q-Q plots for three mice. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from non-parametric bootstrap samples which permute the true labels.

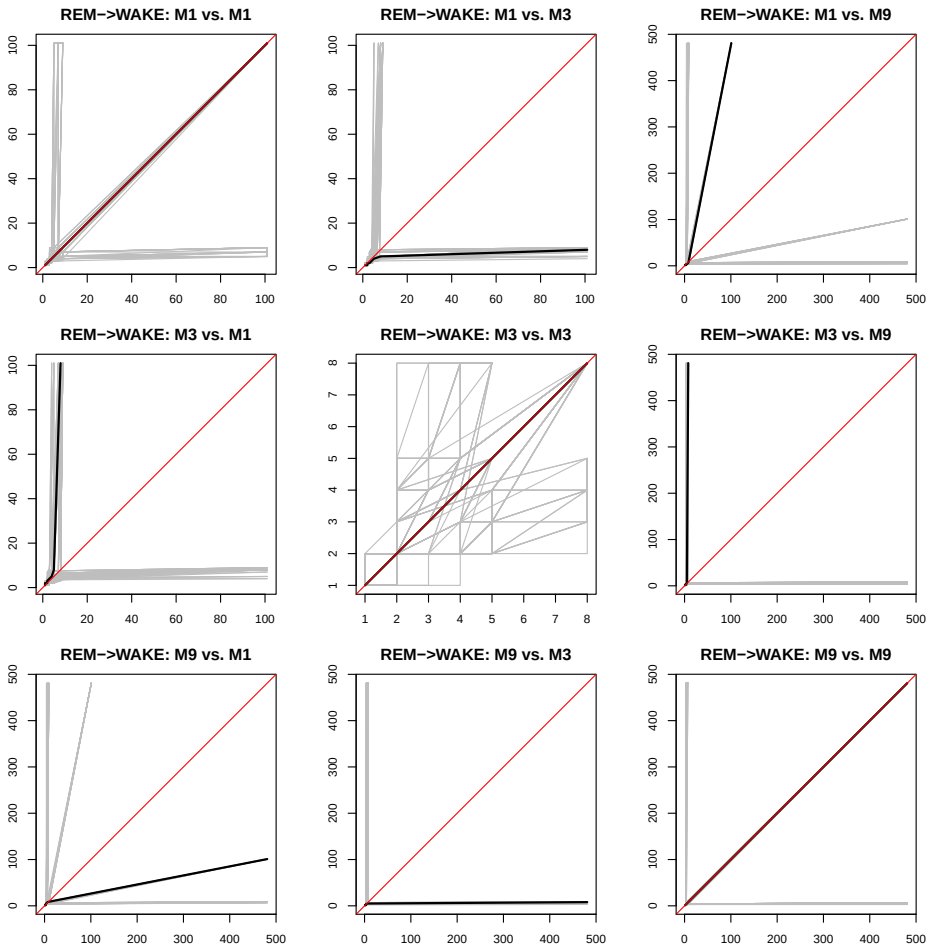

Figure 7.7: REM->WAKE bout duration Q-Q plots for three mice. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from non-parametric bootstrap samples which permute the true labels.

eight mice: the general pattern was that the null hypothesis of equal bout duration distributions for various mice-conditional state combinations could not be rejected. There were some exceptions: mouse five, for instance, seemed to display vastly different behavior for several conditional states. However, since the general pattern was equality of distribution, we assume it for the remainder of this chapter. An interesting direction for future research would be to allow for heterogeneity among mice in terms of these duration distributions.

7.2.3 Fitting Distributions

In order to use a TDGMM, we must first fit distributions to the bout duration sequences. Since the various mice appear to have relatively similar conditional state bout duration distributions, we can combine data across mice in order to do this. We first start with the simplest possible distribution: the geometric (all distributions considered in this section are shifted to have support on $1, 2, ...$). We again use Q-Q plots to assess the quality of the fit, this time providing null bands based on the parametric bootstrap. Each distribution is fit using the maximum likelihood estimate and then repeated bootstrap samples are drawn from that distribution using the MLE as a plug-in estimate.

Geometric fit results are shown in Figure 7.8. As can be seen, the geometric seems to provide quite a good fit for both NREM states. However, it is woefully inadequate for REM and WAKE. For REM, the empirical bouts are in general much

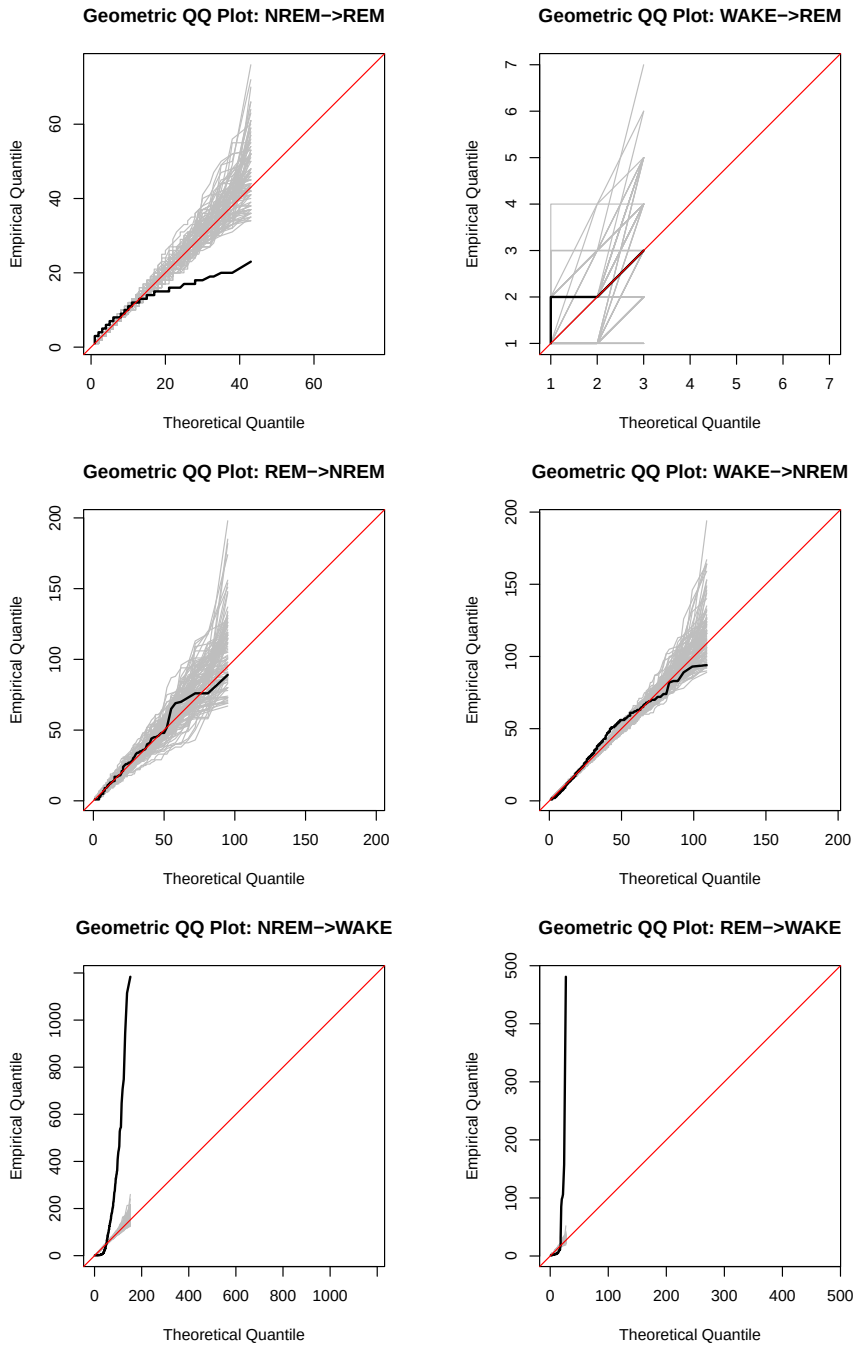

Figure 7.8: Q-Q plots for Geometric Distribution Fits to the Conditional States. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from parametric bootstrap samples based on the maximum likelihood estimate.

shorter than the theoretical ones obtained from resampling based on the MLE. On the other hand, for both WAKE states, the opposite is true: the geometric cannot accommodate the long bouts observed in practice.

Since the geometric is incapable of fitting these distributions, we consider two generalizations of it: the negative binomial distribution and the beta geometric distribution. The former generalizes the geometric as a sum of an arbitrary number of geometrics whereas the latter adds heterogeneity by forming a mixture of geometrics (with the beta distribution providing the mixture distribution).

We begin with the NBD fits in Figure 7.9. Not surprisingly, the NBD fits the NREM states well because it generalizes the geometric and the geometric fit them well. It also, however, fits REM surprisingly well. Again, the theoretical quantiles are a bit larger than the empirical ones but they are now within the range of sampling variability. However, the two WAKE states are still off: the NBD does not accommodate the long bouts observed in practice.

The beta geometric fits are given in Figure 7.10. Unfortunately, the beta geometric does not provide particularly good fits to the data.

Finally, we consider a more complicated generalization of the geometric (which also generalizes both the NBD and the beta geometric), the beta negative binomial distribution. This distribution generalizes the geometric both with a beta mixture distribution on the underlying parameter and by allowing for an arbitrary sum. The results are shown in Figure 7.11 and appear somewhat promising. The REM and

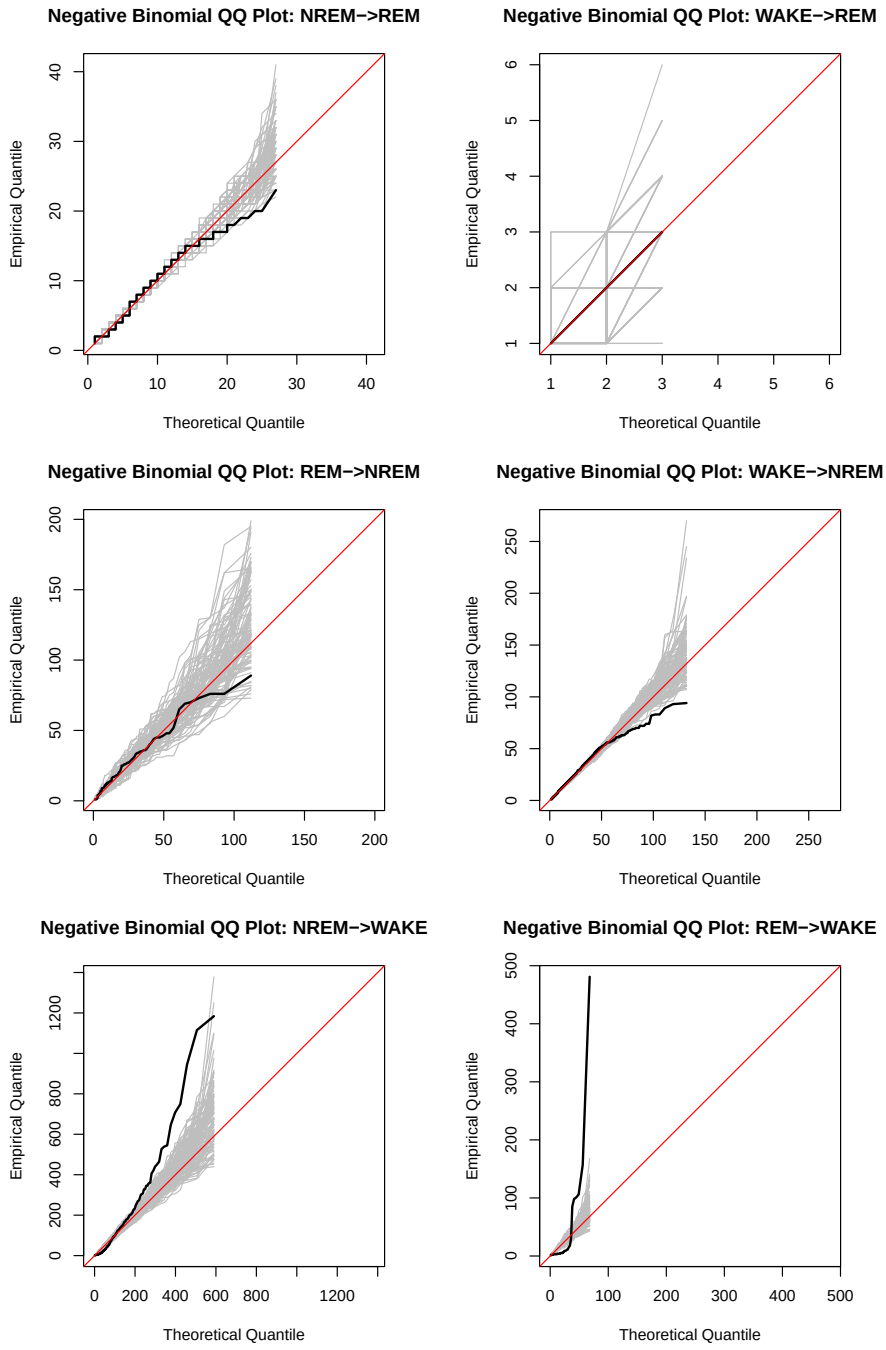

Figure 7.9: Q-Q plots for Negative Binomial Distribution Fits to the Conditional States. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from parametric bootstrap samples based on the maximum likelihood estimate.

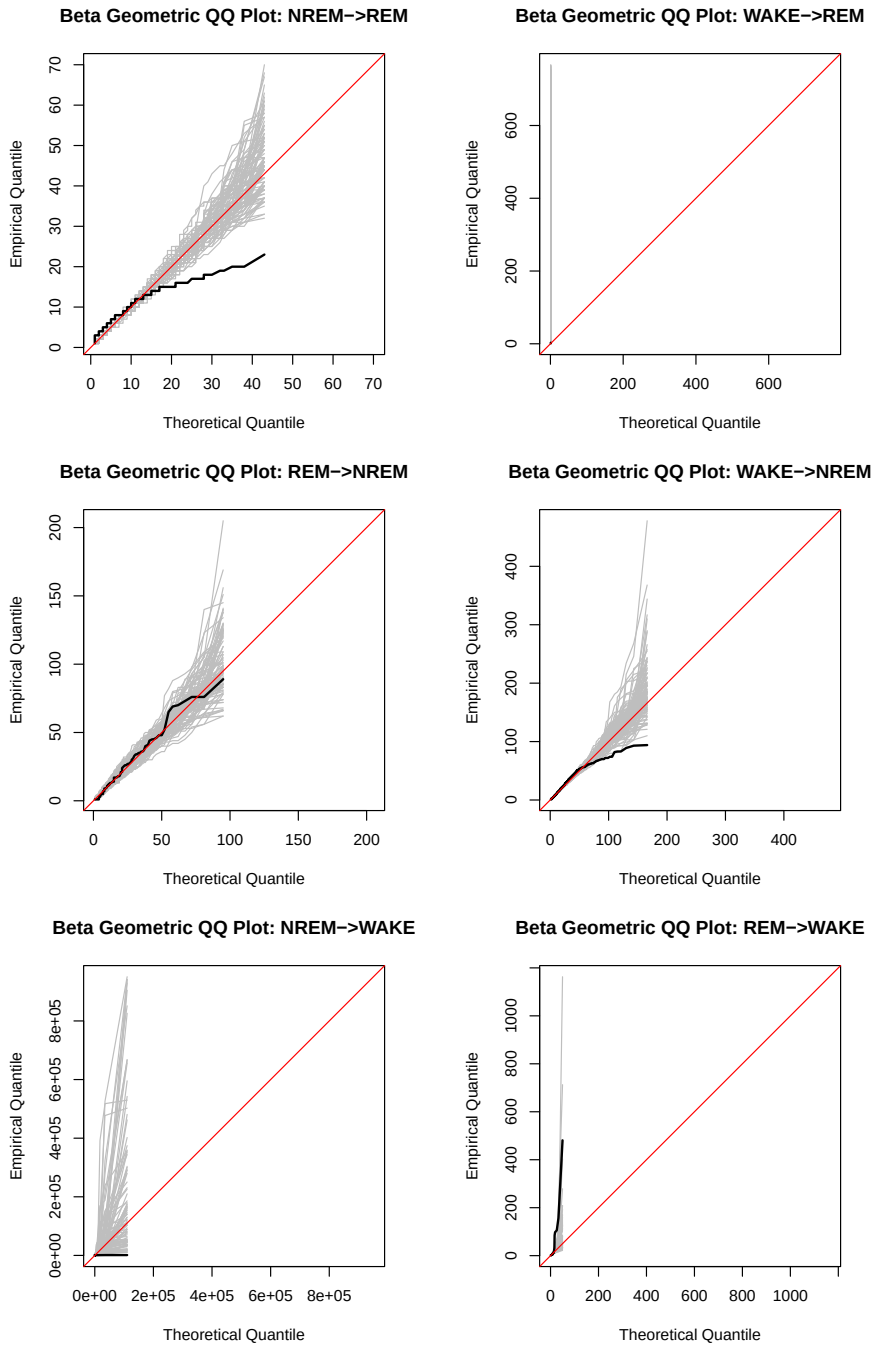

Figure 7.10: Q-Q plots for Beta Geometric Distribution Fits to the Conditional States. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from parametric bootstrap samples based on the maximum likelihood estimate.

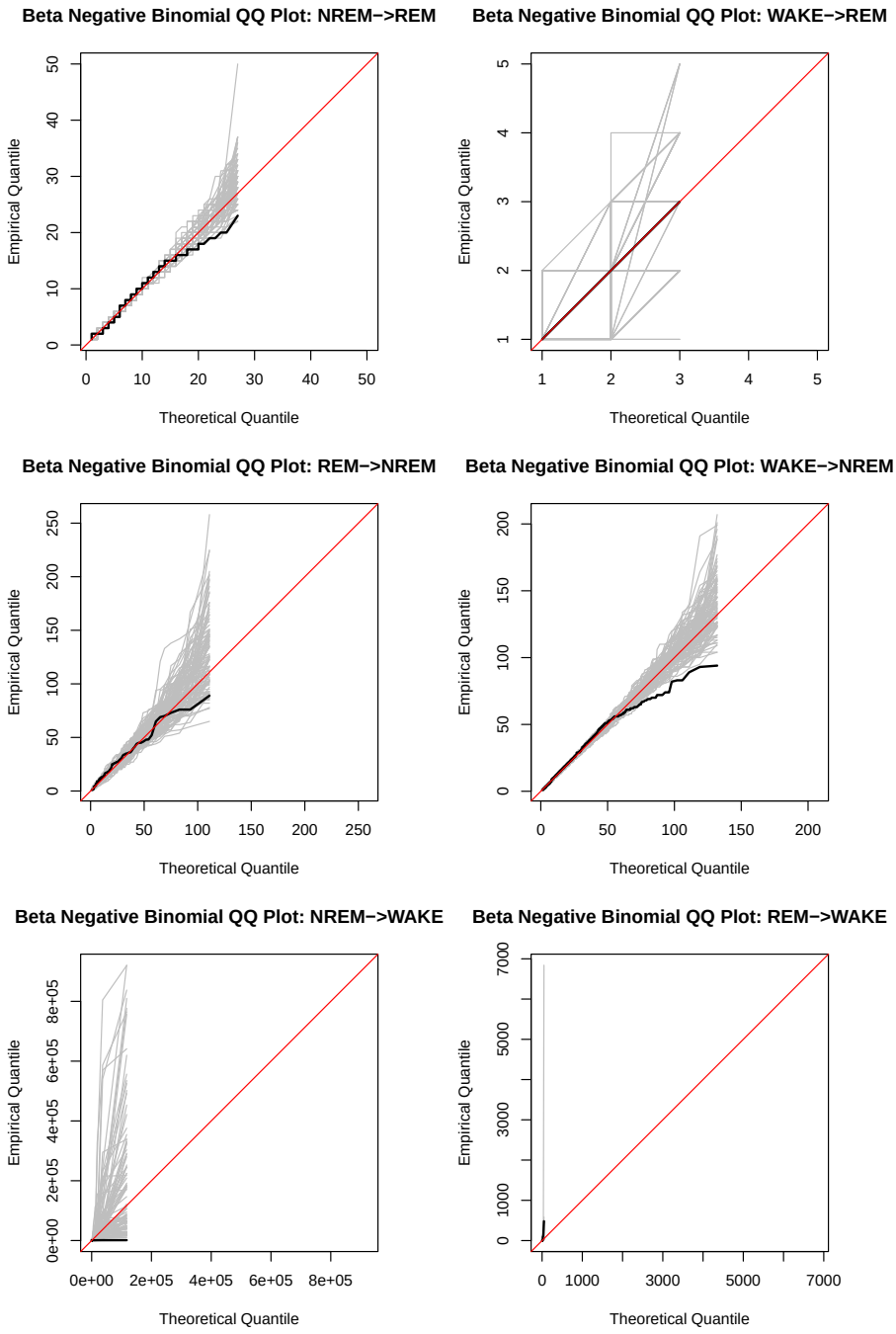

Figure 7.11: Q-Q plots for Beta Negative Binomial Distribution Fits to the Conditional States. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from parametric bootstrap samples based on the maximum likelihood estimate.

NREM plots seem to fit reasonably well. The WAKE plots are hard to examine due to scale issues: there are mismatches between the maximum empirical and maximum theoretical observations.

Now, technically, none of these distributions other than the geometric can be accommodated by our framework. The TDGMM requires duration distributions with either finite support or geometric tails beyond some threshold value. Hence, since the beta negative binomial seemed to provide the best fits, we consider using this as the base distribution as in Simulation D (see Equations 6.1.1 and 6.1.2).

The first step in this process was to provide a definition for the "tail" of the distribution. We did this based on empirical quantiles and fit the beta negative binomial distribution with geometric tails for tail definitions of 50%, 60%, 70%, 80%, 90%, 95%, and 99%. After examining the Q-Q plots, we chose the tail definition of 95% for both NREM->REM and WAKE->REM, 50% for REM->NREM, 90% for WAKE->NREM, and 99% for both NREM->WAKE and REM->WAKE states (the empirical quantiles were 16, 2, 14, 38, 352, and 99 respectively).

These fits are shown in Figure 7.12. As can be seen, this distribution appears to fit all six conditional states quite well, with all Q-Q lines staying within the gray null region. As a final consideration, we provide the same plot again, however, only plotting durations less than or equal to ten epochs in Figure 7.13. That is, we zoom in on the short bouts allowing us to examine the "spike" part of the distribution (McShane *et al.*, 2010). As can be seen, the beta negative binomial

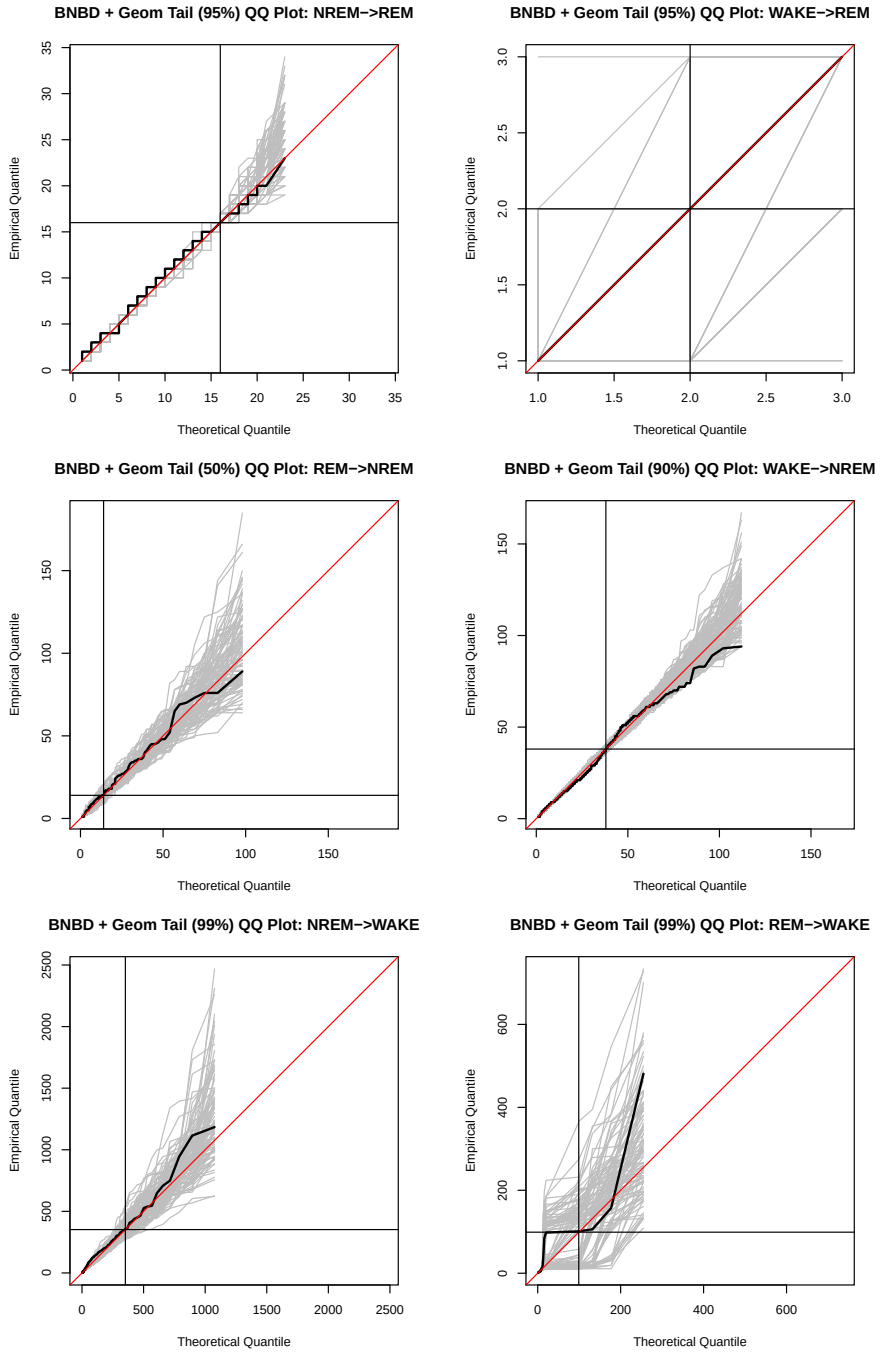

Figure 7.12: Q-Q plots for Beta Negative Binomial Distribution with Geometric Tail Fits to the Conditional States. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from parametric bootstrap samples based on the maximum likelihood estimate. The black horizontal and vertical lines show the tail cut-off.

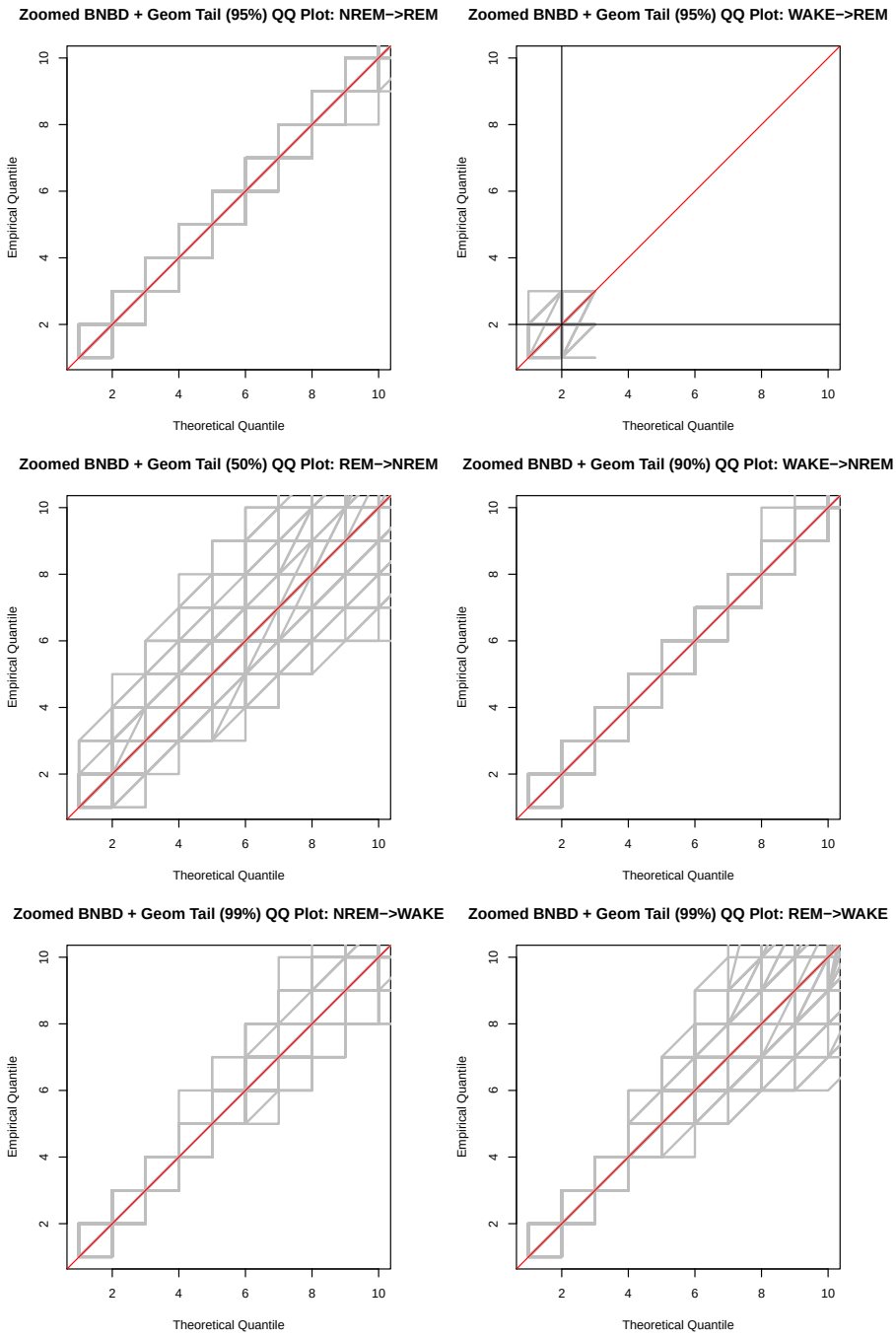

Figure 7.13: Zoomed Q-Q plots for Beta Negative Binomial Distribution with Geometric Tail Fits to the Conditional States. The Q-Q line is given in black and the $y{=}x$ line in red. The gray region represents null bands formed from parametric bootstrap samples based on the maximum likelihood estimate. The black horizontal and vertical lines show the tail cut-off.

distribution with a geometric tail does quite well on the spike: the black Q-Q lines are contained fully within the null regions. Hence, we can conclude this distribution provides an adequate fit and we use it for our TDGMM.

7.3 Methods Considered

7.3.1 Competing Methods

As discussed in Chapter 2, there are a variety of approaches used to fit sequential time series data. Here, we briefly go over the methods we apply to the sleep data.

Since the PrAGMaTiSt is a hybrid machine learning / Markov model approach, we first consider its two isolated components: standard machine learning methods and stand-alone Markov models. The non-sequential machine learning procedures we consider are logistic regression, Random forests, AdaBoost, LogitBoost, bagged classification trees, and bagged probability trees. These methods ignore the sequential nature of the data and therefore we expect them to perform poorly because they lack the power to take advantage of correlations that exist among nearby Y_t and X_t. We also fit standard generative Markov models, implemented as both an unconstrained multivariate Gaussian and as a three-component mixture of Gaussians with covariance matrices constrained to be diagonal. These methods should also perform poorly but for the opposite reason: though they take the time series nature of the data into account, they will be unable to capture the complex structure of the

conditional covariate distributions (i.e., the multidimensional μ_i cannot be modeled as a simple distribution).

We also consider a variety of methods which have been proposed for sequential data. Since these methods have been applied with success to a variety of tasks such as part-of-speech tagging, text-to-speech mapping, biological sequence analysis, and information extraction from web pages, they may indeed perform well on sleep data. The principle method we consider are CRFs implemented in the same two ways as in Chapter 6. Again, in a standard implementation, CRFs are a linear method and we would not expect them do well in the noisy, non-linear setting of sleep behavior in mice. Higher order terms can be added to the feature space of CRFs to capture non-linearities. However, this augmentation strategy vastly increases the dimensionality of the parameter space (sometimes even to the point that the model cannot be estimated) and risks over-fitting to extreme probabilities. Thus, we also consider the TreeCRF approach.

For the linear MALLET CRF, we used the default parameter settings and trained the CRF three times: once using the data described above, once augmenting the data to form a response surface of order two, and once augmenting the data to form a response surface of order three. TreeCRF requires discrete data. We binned our data using normal quantiles for each feature; bin numbers considered were 5, 10, 25, and 100. TreeCRF also has a number of leaves parameter which we set to 8, 16, 32, 64, 128, and 256. For brevity, only the subset of the TreeCRF results

which showed the best performance are given, so that the results shown represent the *best-case scenario* for this method.

7.3.2 PrAGMaTiSt

Sequential classification and class probability estimation procedures, such as the neural networks used in Smyth (1994), are known for producing probability estimates close to zero or one. In many applications, such as the original one of signal failure detection, overfit conditional class probability estimates are not problematic since the true conditional class probabilities $\mathbb{P}(Y_t|X_{1:T})$ are often close to zero or one (that is, the signal is strong). In these settings, combining machine learning methods with an Markov model for time dependence serves to locally smooth conditional class probability estimates.

Many applications have much more noise. Sleep stages, as noted earlier, are hard to classify even with EEG/EMG data. Consequently, over-fit estimates of the conditional class probabilities that are driven to the 0/1 boundary will be insensitive to the remaining, less influential local time series dependence in the data. This problem is compounded by the fact that transitions between states are quite frequent in mouse sleep. Mice transition between sleep states on average 550 times per day (*i.e.*, about 4-10% of all epochs). We therefore rely on the Random forest procedure that is known to produce reasonably-calibrated probability estimates (Bostrom, 2007, 2008), which will form a more effective combination with the time

series dependence in the data.

We combine the Random forest with the TDGMM as discussed above. In particular, we estimate each conditional state duration distribution as beta negative binomial with geometric tail where the definition of the tail depends on the state. We also combine the Random forest with an first order Markov model since, in Chapter 6, we saw that sometimes the mis-specified first order model can outperform more complicated models even when the latter are correct. We omit consideration of the GMM since Figure 7.2 showed that the states had duration distributions which differed depending on the previous state.

We prefer the proposed PrAGMaTiSt methodology to the three broad classes of methods–standard machine learning algorithms, generative Markov models, and sequential machine learning methods like CRFs–for several reasons. Our method should be superior to standard machine learning algorithms since it updates them with information about the sequential nature of the problem. It should also outperform generative Markov models because it does not make covariate emission distribution assumptions which are likely to be false in practice.

We have several reasons to prefer the PrAGMaTiSt to other sequential machine learning methods like CRFs. First, the generative or "emission" assumptions of the Markov model are quite natural for our particular application. That is, it is natural, for example, to assume that a mouse "emits" a velocity based on its current sleep stage. While generative models can often be inferior to discriminative ones, this is

a setting where the generative approach is appropriate (and we get the benefits of discrimination via the Random forest). In addition, the classification boundaries in this application are highly non-linear, a drawback for standard CRFs, though tree-trained CRFs as well as our procedure based on Random forests should adequately handle non-linearities.

Our general Markov modeling approach is also naturally generalizable to higher order Markov chains, variable length Markov chains, and generalized Markov Models. Hence, we can accommodate complicated dependence patterns in the Y_t that CRFs cannot. We saw this was important in Simulations 1C, 1D, 2C, and 2D and have reason to believe it is important for the sleep application.

7.4 Evaluation Criteria

We evaluate the various methodologies by training on the full set of time-points from one mouse and then testing our classifications on the full set of time-points from *every other* mouse. This validation scheme is designed to match the experimental situation, in which a model would be trained on time segments for one (or more) mouse and used completely out of sample on data obtained from *different* mice. Of course it is possible to train on randomly selected subsets of data (ideally, blocked subsets) from one mouse and test on hold out subsets reserved from the same mouse. The comparative performance of "same mouse but out-of-sample" is much better, but also irrelevant to the scientific enterprise.

There are only 8 mice in the data set. However, we train on one, fit to the other seven, and then repeat this procedure over all eight mice yielding nearly 500,000 out of sample epochs. Thus, all results presented are statistically significant under the standard multinomial models, even though these models do not apply in the presence of strongly autocorrelated time series data. Consequently, we do not provide standard errors because it is not clear what the probability model is for the test statistic.

The EEG/EMG-based manual scoring of these eight mice is the "gold standard" for classification. There is an issue with internal inconsistency of manual scoring: classifications from different scorers agree in only about 92% of the epochs. Nevertheless, any model classifications which did not substantially match manual scoring would not be considered useful by sleep researchers. In addition to this overall error rate, we also consider the false positive and false negative rate for the rare and important REM state which is of special interest to sleep researchers. They prefer a low REM false negative rate but can tolerate a high REM false positive rate because there are so few REM epochs.

A second way we evaluate our predictions is by comparing the fitted duration distributions to the actual ones. This is important because sleep researchers are sometimes interested in estimating the parameters of these distributions and seeing how they vary across mice (McShane *et al.*, 2010). In this case, fitting the distributions well is what is important, even if the epoch-by-epoch classifications themselves

are not particularly accurate. We did this via χ^2 goodness-of-fit statistics. First, we formed bins for each of the six conditional states based on the empirical distributions. We started with a single bin for durations of length one and added individual durations until the bin had greater than 5% of the empirical bouts in it. We iterated this process ensuring that all bins (including the terminal bin) had 5% or more data.

As an example, consider the state NREM->WAKE. 44.83% of the empirical NREM->WAKE bouts were of length one hence this became its own bin. Likewise 19.57% and 6.13% were of length two and three epochs respectively thus leading to those durations being their own bins. 5.81% of empirical bouts had duration of either four or five epochs thus defining the fourth bin. 5.56% had durations of six, seven, eight or nine epochs yielding the fifth bin. 5.04% had durations between ten and twenty-three epochs and 5.10% were between twenty-four and sixty-five epochs yielding the sixth and seven bins respectively. Finally, the last 7.95% of epochs were sixty-six epochs or longer in duration thus defining the terminal bin.

For each of the eight mice, we fit the model. We then computed the observed empirical duration distributions and fitted durations distributions on the other seven mice for each of the six conditional states. From these fits and the bins described above, we obtained six χ^2 statistics (one for each conditional state). We averaged each of these six across the eight mice used to fit the model yielding six average χ^2 statistics per method considered.

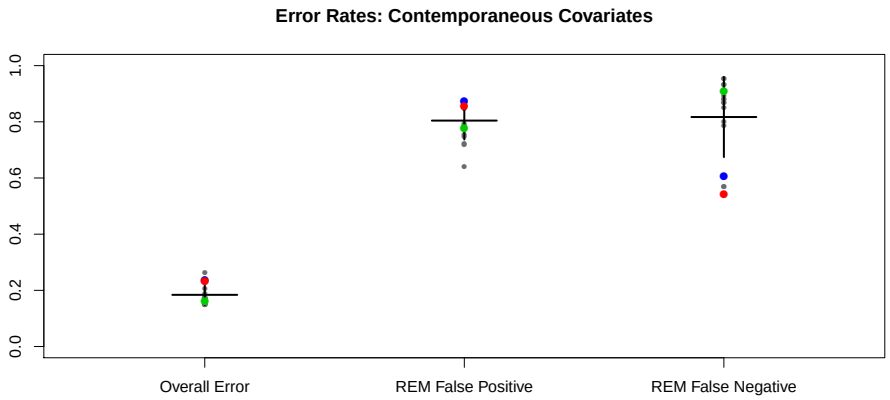

Figure 7.14: Error Rates for Various Methods. The black horizontal lines indicate the mean across all methods and the black vertical lines denote ±1 standard error. Random forests error rates are given in green, Random forests with Markov model in blue, and Random forests with TDGMM in red.

These two sets of metrics, the three error rates and the six χ^2 statistics, allow us to evaluate the various methods "locally" and "globally". The error rates show how well the methods perform on an epoch-by-epoch basis, something which is important for replicating EEG/EMG-based manual scoring. On the other hand, the χ^2 statistics show how the methods perform in terms of fitting the entire curve of duration distributions, a task also relevant for sleep scientists and one that does not necessarily require good fits on an epoch-by-epoch basis.

7.5 Results

The three error rates for the various methods on held-out epochs are presented in Table 7.3 and Figure 7.14. This table reveals what is already well known to

Table 7.3: Error Rates for Various Methods

Method	**Overall Error**	**REM False Positive**	**REM False Negative**
RF	0.161	0.779	0.910
RF + MM Mode	0.237	0.872	0.605
RF + TDGMM Mode	0.233	0.857	0.543
Logistic Regression	0.149	0.723	0.953
LogitBoost	0.205	0.860	0.869
AdaBoost	0.169	0.787	0.908
Bagged Classification Trees	0.152	0.749	0.934
Bagged Probability Trees	0.151	0.756	0.934
Gaussian Markov Model	0.262	0.865	0.571
Mixture Markov Model	0.242	0.859	0.568
TreeCRF: Bins=5, Leaves=32	0.172	0.863	0.867
TreeCRF: Bins=5, Leaves=64	0.175	0.872	0.878
TreeCRF: Bins=10, Leaves=32	0.173	0.853	0.882
TreeCRF: Bins=10, Leaves=64	0.169	0.847	0.893
MALLET CRF	0.151	0.639	0.852
MALLET CRF Order 2	0.175	0.784	0.802
MALLET CRF Order 3	0.187	0.793	0.785
40 Second Rule	0.146	NA	NA
Gold Standard EEG	$\approx$0.080	NA	NA

sleep scientists: REM is very difficult to classify correctly, with high false positive and false negative rates across all methods relative to the overall error rates. The challenge here is to discover a method which has power to detect REM sleep, a task which sleep scientists believed to be impossible through video analysis alone.

Our methodology (RF + MM and RF + TDGMM) is competitive at the overall classification task. More importantly, we see dramatic improvement over other methods in terms of false negative predictions for the REM state. By accounting for the local time-series dependence of the data, our procedure is able to capture

a much greater proportion of the subtle REM signal. Furthermore, as shown in Figure 7.14, our procedure retains a reasonable false positive rate relative to the other methods, suggesting that specificity is not being sacrificed in order to gain substantial improvements in sensitivity. In fact, there is a dramatic improvement in REM false negative rates obtained by moving from the standard Random forest to the random-forest trained Markov model; the benefits of moving from the first order Markov model to the TDGMM are modest but certainly non-trivial.

One interesting result is the performance of the so called "40-second Rule", prominent in the sleep literature (Pack *et al.*, 2007). The 40-second Rule considers a mouse inactive in a given 10s epoch if the mean intra-epoch velocity is less than 3 pixels/second and rules a mouse asleep when there are four or more consecutive inactive epochs. The 40-second Rule does not distinguish between REM and NREM sleep. For ease of comparison, all epochs classified as sleep by this method were labeled as NREM. Thus, the 40-second Rule is not applicable to REM detection at all. On the other hand, this simple rule has the best overall error rate. This suggests that methods which simply ignore the REM state can do very well on the overall error rate even though they perform dreadfully on the state of most interest to sleep researchers.

We evaluate the distributional fits on held-out data in Table 7.4 and Figure 7.15. Here the advantage of the TDGMM over the first order Markov model becomes more apparent: the TDGMM tends to provide dramatic improvement in the

Table 7.4: χ^2 Goodness of Fit Statistics for Various Methods

Method	**NREM-> REM**	**WAKE-> REM**	**REM-> NREM**	**WAKE-> NREM**	**NREM-> WAKE**	**REM-> WAKE**
RF	335.4	4.1	88.2	543.2	89.4	18.5
RF + MM	208.7	44.8	18.5	112.4	369.2	135
RF + TDGMM	209.9	4.7	19.3	71.6	164.5	40.5
Logistic Regression	108.8	2.4	45.5	421.1	93	41.7
LogitBoost	900.1	7.6	161.3	1049.7	181.4	83.3
AdaBoost	238.1	5.3	72.1	520.1	91.9	32.2
Bagged Classification Trees	121.9	2.5	64.6	487.3	95.6	24.5
Bagged Probability Trees	131.6	2.2	61.9	478.6	95.7	22.6
Gaussian Markov Model	177.5	10.4	18.5	81.4	206.5	71.8
Mixture Markov Model	143.5	9.8	19.9	156.8	174.4	72
TreeCRF: Bins=5, Leaves=32	52.5	6.7	20.1	71.8	241.1	97.2
TreeCRF: Bins=5, Leaves=64	50.6	6.2	21.9	61.1	228.9	96.4
TreeCRF: Bins=10, Leaves=32	36.5	7.2	23.1	65.5	273.4	106.8
TreeCRF: Bins=10, Leaves=64	35	8.8	22.1	77.4	272.6	95.9
MALLET CRF	125.2	2.9	13.6	122.3	213.8	52.4
MALLET CRF Order 2	119.8	5.5	16.6	93	160.4	49.6
MALLET CRF Order 3	111.8	5	18.2	86.2	206.8	62
40 Second Rule	NA	NA	NA	125.7	1408.4	NA
Gold Standard EEG	NA	NA	NA	NA	NA	NA

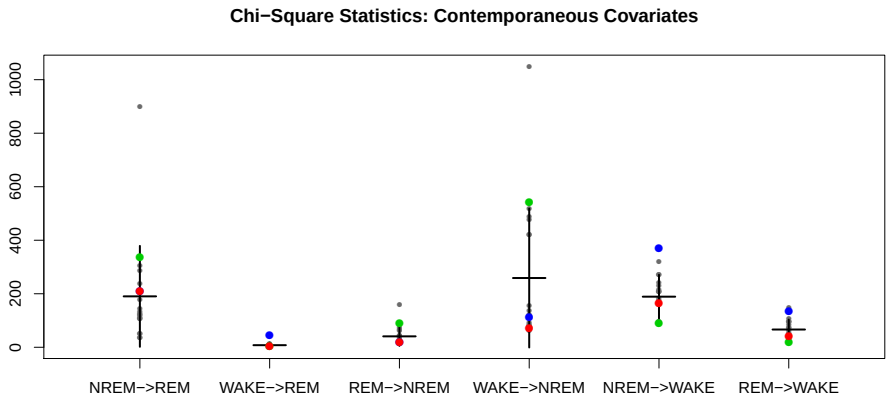

Figure 7.15: χ^2 Statistics for Various Methods. The black horizontal lines indicate the mean across all methods and the black vertical lines denote ± 1 standard error. Random forests is given in green, Random forests with Markov model in blue, and Random forests with TDGMM in red.

χ^2 goodness-of-fit statistic relative to the simpler model. Thus, the more complicated distribution information captured by a TDGMM really does matter quite considerably for this more "global" task. Figure 7.15 prominently shows another feature of the TDGMM approach: it is consistently near or better than the mean performance for all six states. No other method is able to do this. All seem to provide good fits to one or two states at the expense of the other states. Hence, our method is quite successful.

7.6 Results: Augmented Covariates

One principal advantage of standard machine learning methods, CRFs, and our discriminatively-trained Markov models had over standard Markov models was the ability to introduce sliding-window and long-distance features into the covariate space in a natural way. In this section, we explore whether such features are helpful here and how they affect the relative performance of the various methods.

The standard covariates consist of six continuous variables (means and standard deviations of intra-epoch velocity, aspect, ratio and size; log mean and log standard deviation for velocity and size) and one binary feature (whether or not the light in the cage is turned on or off). To this space, we added twelve additional covariates: forward moving averages of order ten and backward moving averages of order ten of each of the six continuous covariates. These features should be helpful particularly to the non-sequential methods because they will allow them to capture some of the

Table 7.5: Error Rates for Various Methods Using Augmented Covariates

Method	**Overall Error**	**REM False Positive**	**REM False Negative**
RF	0.131	0.641	0.869
RF + MM Mode	0.260	0.833	0.320
RF + TDGMM Mode	0.235	0.819	0.328
Logistic Regression	0.126	0.542	0.782
LogitBoost	0.172	0.736	0.784
AdaBoost	0.135	0.626	0.826
Bagged Classification Trees	0.126	0.723	0.943
Bagged Probability Trees	0.126	0.709	0.948
Gaussian Markov Model	0.218	0.764	0.590
Mixture Markov Model	0.234	0.841	0.472
TreeCRF: Bins=5, Leaves=32	0.140	0.741	0.862
TreeCRF: Bins=5, Leaves=64	0.138	0.741	0.876
TreeCRF: Bins=10, Leaves=32	0.139	0.702	0.893
TreeCRF: Bins=10, Leaves=64	0.140	0.729	0.905
MALLET CRF	NA	NA	NA
MALLET CRF Order 2	NA	NA	NA
MALLET CRF Order 3	NA	NA	NA
40 Second Rule	0.146	NA	NA
Gold Standard EEG	$\approx$0.080	NA	NA

time series structure in the Y_t via nearby X_t (note, MALLET would not run on this higher-dimensional dataset).

The three error rates for the various methods trained on the augmented covariate space are presented in Table 7.5 and Figure 7.16. All methods demonstrate improvement to some degree.

Our RF + TDGMM is particularly interesting. It has an approximately equivalent overall error rate compared to when it was fit using only the standard covariates. However, the REM false negative rate has dramatically improved with even a small

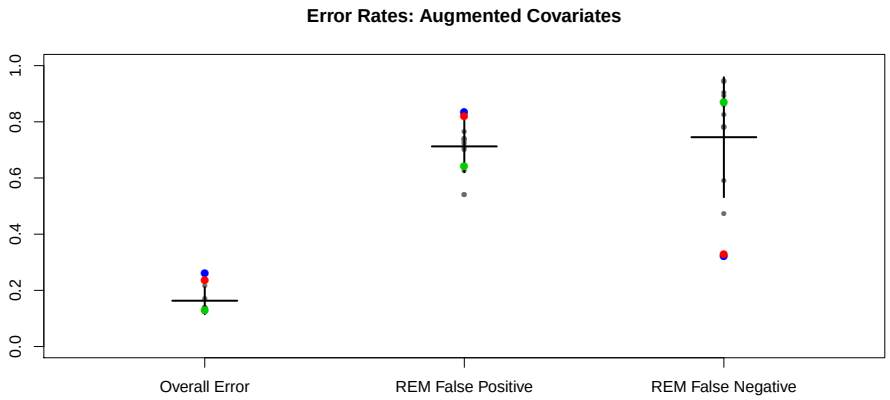

Figure 7.16: Error Rates for Various Methods Using Augmented Covariates. The black horizontal lines indicate the mean across all methods and the black vertical lines denote ±1 standard error. Random forests error rates are given in green, Random forests with Markov model in blue, and Random forests with TDGMM in red.

decrease in REM false positive rate. It is rare for a method to yield improvements in both the false positive and negative rates for a class. Interestingly, with the augmented covariates, the TDGMM and first order Markov model perform similarly on the two REM rates suggesting that, whatever longer-term information is provided by the TDGMM's beta negative binomial distribution, it can be more or less captured via local covariates and a simple first order Markov model geometric distribution, at least for REM; the better performance of the TDGMM on the overall error rate demonstrates that this is not necessarily the case for the other two states. This makes good sense since REM is by far the shortest duration state.

Also, with the augmented covariates, several methods beat the 40-second rule on overall error rates, even coming fairly close to the EEG error rates. While this

Table 7.6: χ^2 Goodness of Fit Statistics for Various Methods Using Augmented Covariates

Method	**NREM-> REM**	**WAKE-> REM**	**REM-> NREM**	**WAKE-> NREM**	**NREM-> WAKE**	**REM-> WAKE**
RF	189.9	2	54.7	104.4	37.7	10.7
RF + MM	271.6	23.1	31.8	152.9	287	191.1
RF + TDGMM	299.2	5	32.1	143.9	259.6	96.5
Logistic Regression	226.7	3.4	28.6	280.6	31.1	90.2
LogitBoost	486.9	3.9	84.9	473	66.3	57.8
AdaBoost	311.6	2.1	61.2	186.9	32.1	18
Bagged Classification Trees	104.1	2.3	44.1	148.7	26.4	29.6
Bagged Probability Trees	99.6	2.2	42.3	155.5	26.2	25.7
Gaussian Markov Model	72.5	2.4	39.2	94.7	165.4	76.9
Mixture Markov Model	150.7	14.3	40.4	80.4	121.9	79.4
TreeCRF: Bins=5, Leaves=32	35.1	5.9	24.5	165.1	74.5	29.9
TreeCRF: Bins=5, Leaves=64	28.7	6.6	21.5	182.4	76.5	35.8
TreeCRF: Bins=10, Leaves=32	32.7	5.6	22.3	187.5	64.5	43.1
TreeCRF: Bins=10, Leaves=64	50.5	3.7	23	170.9	60.7	40.7
MALLET CRF	NA	NA	NA	NA	NA	NA
MALLET CRF Order 2	NA	NA	NA	NA	NA	NA
MALLET CRF Order 3	NA	NA	NA	NA	NA	NA
40 Second Rule	NA	NA	NA	125.7	1408.4	NA
Gold Standard EEG	NA	NA	NA	NA	NA	NA

is promising, it is not necessarily helpful for sleep scientists: these methods, like the 40-second rule, are able to perform so well on the overall error rate largely by ignoring the REM state and hence demonstrate extremely high REM false negative rates.

Finally, in Table 7.6 and Figure 7.17, we examine the performance of the various methods at fitting the full duration distributions. These results are rather less promising. First, as with the error rates, the first order Markov and TDGMM variants of the PrAGMaTiSt perform quite similarly. Second, it seems augmenting the covariate space has actually degraded the predictions of our two methods, at least in terms of relative performance: rather than performing better than average across each of the six states, they tend to perform considerably worse than average.

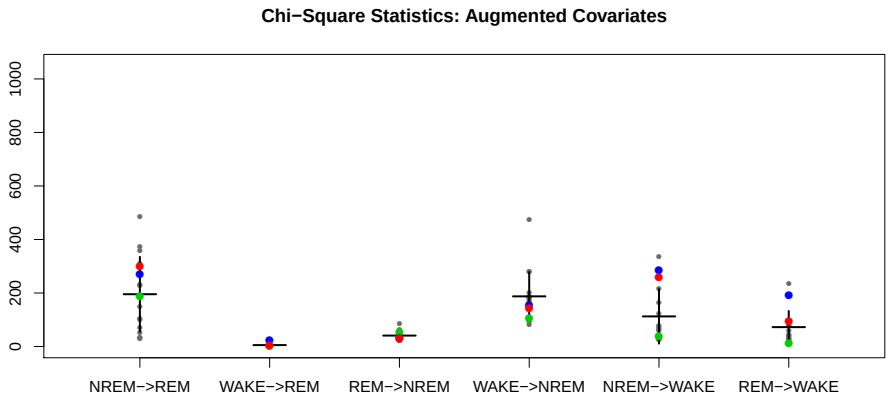

Figure 7.17: χ^2 Statistics for Various Methods Using Augmented Covariates. The black horizontal lines indicate the mean across all methods and the black vertical lines denote ± 1 standard error. Random forests is given in green, Random forests with Markov model in blue, and Random forests with TDGMM in red.

In fact, it seems the degradation on the distributional fits is not only relative but also absolute. For many of the states, the χ^2 statistics are worse than those in Table 7.4.

On the other hand, the plain Random forest demonstrates very strong performance, among the best methods for almost every state. In other words, local X_t seem to serve as a strong proxy for the dependence structure of the Y_t, and, therefore, using standard sequential methods on an augmented covariate space is a reasonable strategy for this endeavor. Nevertheless, this should be done with caution as such an approach will tend to ignore the REM state.

7.7 Variation by Fit Mouse

The error rates and χ^2 statistics presented in the previous section were aggregated across all out-of-sample fit mouse / test mouse combinations. Namely, we trained on one mouse and fit on all other seven, repeating this procedure over all eight mice and averaging.

It is interesting to examine, however, how these rates vary by the fit mouse / test mouse pair. We do this for the RF + TDGMM trained on the original covariates in Figure 7.18. There are several features of note. First, the in-sample fits are in general very good. This is a feature of the Random forest base classifier. Second, when considering overall error rate, some mice seem difficult to predict (e.g., mouse four) no matter which mouse is used to train the algorithm. Likewise, some mice are consistently well-predicted (e.g., mice six and nine). Third, when training using some mice (e.g., mouse four), the error rates for the other mice tend to be on average lower *and* less variable.

Similar variation also applies to REM false positive rates. Mouse three appears difficult to predict and mouse five appears easy predict. However, in general, the variation for REM false positive is much less dramatic than for overall error or REM false negative. Furthermore, it does not appear that one fit mouse tends to dominate the others in terms of average error on the other mice.

For REM false negative rates, there is some systematic difference between easy and hard to predict mice, but it is much less prominent than for the other two error

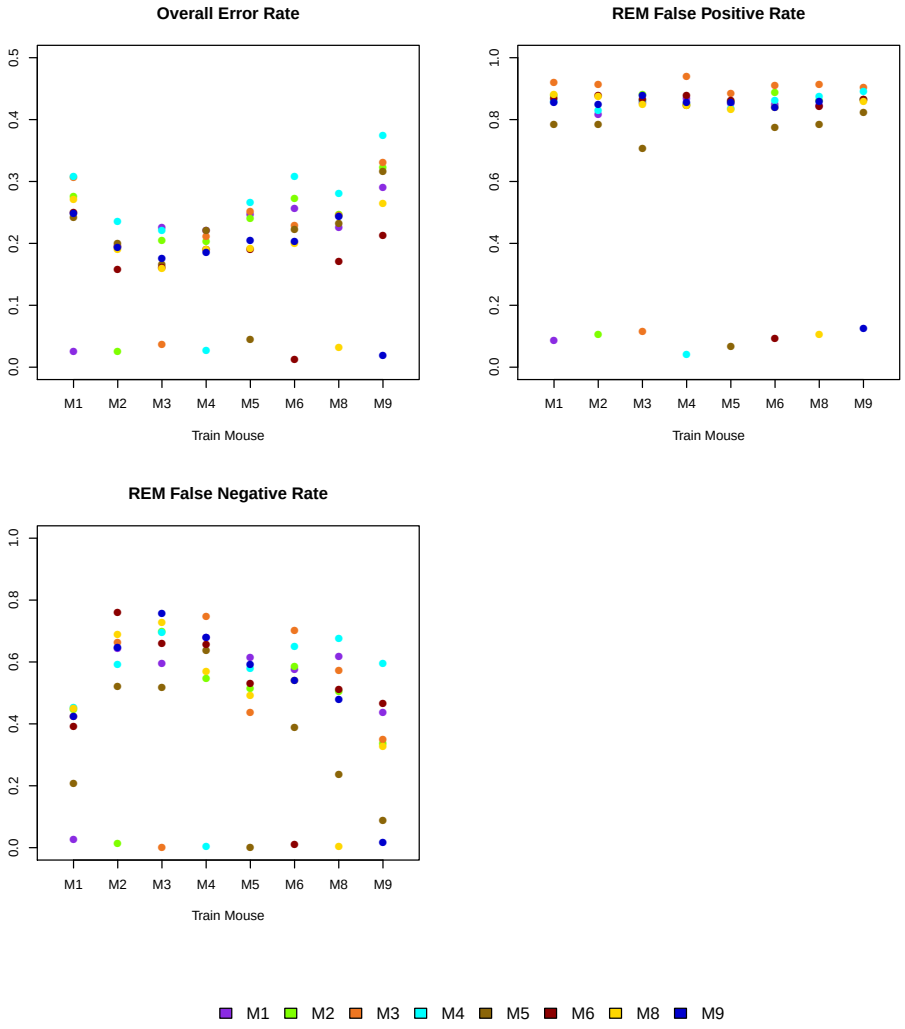

Figure 7.18: Error Rates by Fit Mouse. The first plot gives the overall error rate, the second the REM false positive rate, and the third the REM false negative rate. The mouse used to train the RF+TDGMM is given on the x-axis and the appropriate error when applied to all other mice is plotted on the y-axis.

rates. The most prominent feature of this plot is the wide variability among test mice for a given fit mouse. Furthermore, there appears to be a general trend in the average false positive rate on test mice for a given fit mouse *and* this general trend seems to be inversely related to that for the overall error rate. For instance, fitting with mouse four gives among the lower overall error rates yet among the highest overall REM false negative rates. Fitting with mouse nine provides the worst overall error rates but comparably good REM false negative rates.

Similar patterns hold when estimating the RF + TDGMM using the augmented covariates and are presented in Figure 7.19. A couple of contrasts stand out, however. The overall error rate and REM false positive rate plots appear to have a similar level to those using the standard covariates. However, the variability across test mice for a give fit mouse seems to have increased a bit. On the other hand, the REM false negative rates have dropped considerably for all mice and there appears to be starker differences as the fit mouse is varied.

These results suggest that combining models fit on different mice in a hierarchical fashion might be a useful strategy for future research. If one can adaptively tune the model to provide fits based on a given training mouse which is "most like" a given test mouse, substantial improvements in accuracy could be obtained. Furthermore, the right training mouse might depend on the goal: for example, for a given test mouse, it might be optimal to use one training mouse to minimize overall error and another training mouse to minimize REM false negative rates.

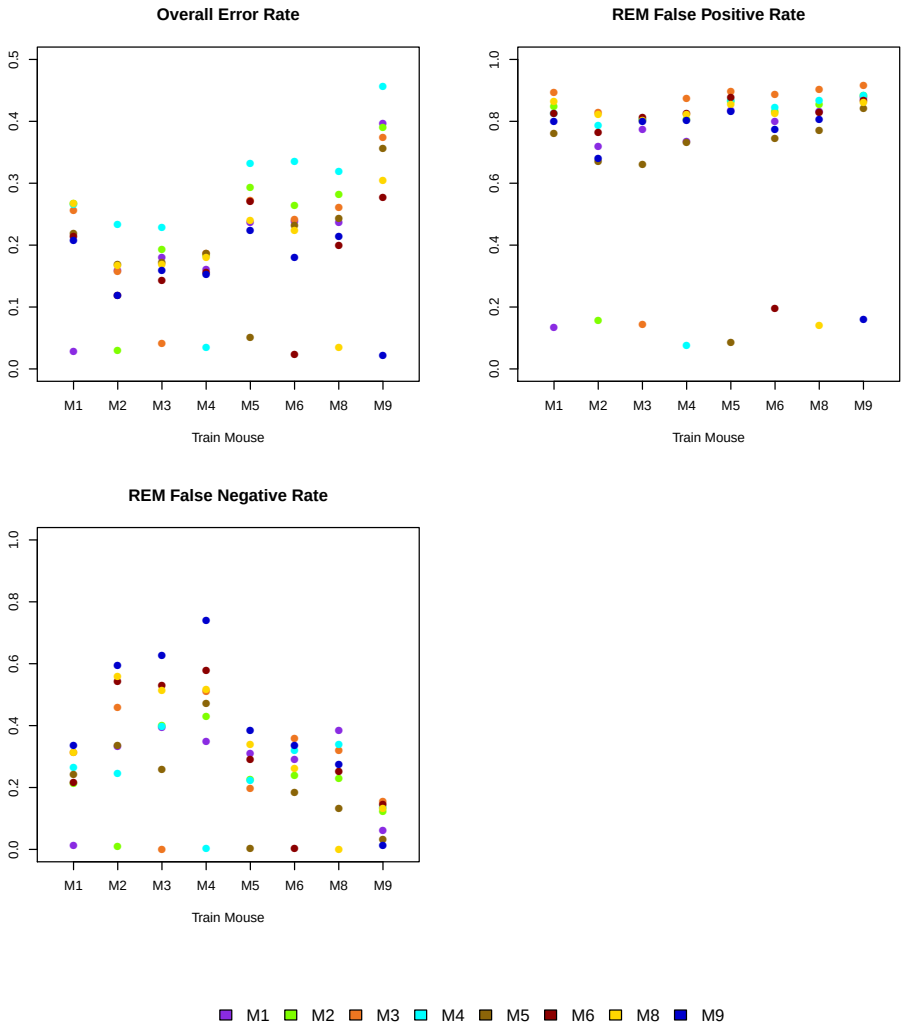

Figure 7.19: Error Rates by Fit Mouse Using Augmented Covariates. The first plot gives the overall error rate, the second the REM false positive rate, and the third the REM false negative rate. The mouse used to train the RF+TDGMM is given on the x-axis and the appropriate error when applied to all other mice is plotted on the y-axis.

7.8 Conclusion

In conclusion, the automated classification of sleep states based on video data is a difficult task: the Y_t are labeled in a noisy fashion, there are complicated dependence structures in the Y_t, the X_t are weak predictors of the Y_t for some states, and the important REM state is not only rare but also similar in terms of X_t to NREM. The PrAGMaTiSt was able to make dramatic improvements in the REM false negative rate relative to other methods while retaining reasonable REM false positive and overall error rates. It also provided among the best fits to the empirical sleep state duration distributions as evidenced by the χ^2 statistics.

However, when an augmented covariate space was considered, the results were somewhat less promising, particularly on a relative basis. The "local" evaluation by error rates remained strong: the REM false negative rate improved substantially with both a small decrease in the REM false positive rate and no concomitant rise in the overall error rate. However, the "global" distributional fits were weaker in both an absolute and a relative sense.

We also saw that there is considerable variability in the error rates among various fit mouse / test mouse pairs. While this should not necessarily be surprising as some mice are more likely to resemble one another than others, the degree of difference seemed quite large. Furthermore, it seems surprising that some mice were just in general easier or more difficult to classify regardless of the mouse used to train the algorithm.

As a final point, it is not surprising that non-sequential methods have the most to gain from augmenting the covariate space. However, the relative improvements of these other methods raises the question of whether more would be gained (i) by further augmenting the covariate space (either by considering functions of the current covariates or by obtaining additional ones via higher-resolution cameras, eye goggles for the mice, or piezo-electric recordings) or (ii) by focusing on modeling improvements and which of these two broad strategies is more efficient or effective. Such efforts are currently underway and are proceeding in tandem.

Chapter 8

Conclusion

8.1 Summation

We have discussed the statistical learning task of classification and its extension to sequential classification. Standard machine learning methods were considered as well as strategies for adapting them to the sequential case. Also, some popular sequential methods were presented. Finally, various difficulties involving class estimation, conditional class probability estimation, optimizing for arbitrary loss functions, variable selection, augmented covariates, and computational complexity have also been presented.

Most notably, we presented the PrAGMaTiSt, our strategy for sequential learning. Our method adapts non-sequential learning algorithms to account for local time series dependence. We do this by training a Markov model in a discrimina-

tive fashion using these base algorithms as an input. This strategy seems effective despite the fact that many machine learning classification tools are not very good at giving well-calibrated estimates of the probabilities for each state. Continuing research into improving the calibration of these probability estimates would provide further improvement to our methodology.

This methodology overcomes some of the principal disadvantages of standard first order Markov models. Using our discriminative training formulation, we no longer have to estimate k p-variate probability distributions. It also allows us to introduce sliding window and long-term features into the covariate space X_t. Furthermore, it allows us to use *any* standard classification algorithm to train the Markov model. Finally, by considering various generalizations of the first order Markov structure, we can introduce dependence directly into the Y_t sequence itself, no longer requiring the relationship between Y_t and Y_{t+k} to be indirect via $Y_{t+1}, ..., Y_{t+k-1}$.

Our Markovian model for the local time dependence is relatively simple and easy to estimate relative to alternative strategies for modeling time series dependence. The naive sliding window approach augments the covariate space for each time period with covariates from the surrounding time periods and then uses conventional machine learning methods on the augmented set of covariates. However, relative to our simple Markovian structure, this direct modeling can be extremely difficult in terms of the number of parameters to estimate, especially since X_t alone will be

high dimensional in many applications. Furthermore, this strategy can be ineffective when the dependence among the Y_t is not well-captured by neighboring X_t.

By embedding higher order, generalized, and variable length Markov structures into a first order Markov model, we can directly model very complicated Y_t dependence structures in a way that first order Markov models and sequential methods such as MEMMs and CRFs cannot. Moreover, the embedding allows use of the forward-backward and Viterbi algorithms. This is absolutely critical to our endeavor because, otherwise, our model could be specified and estimated but we would be unable to apply it to test data.

A thorough simulation study revealed some very interesting facts about the performance of various methods, most pertaining to how they perform as the noise level in $Y_t|X_t$ varies. In particular, for very high noise settings, the covariates become rather useless for prediction and the base rates of the classes yield good classifications and probability estimates, even in the presence of time series structure. This has a further implication: since all methods will perform more or less the same in high noise settings, simple methods will tend to dominate with finite data because they can provide better estimates. On the contrary, in very low noise settings, all methods also perform the same regardless of the time series structure because $Y_t|X_t$ puts almost all of its mass on one class. Hence, at the extremes, simple methods, even ones which lack time series structure, seem best.

The A and B simulations showed that we do not lose much with a moderate

amount of data if we fit GMMs and TDGMMs when the truth has either no time series structure or is first order Markov. However, the C and D simulations showed a contrary result. With sufficient noise, complexity of $Y_t|X_t$, or complexity of the time series structure, the standard Markov model can perform as well as or even better than the GMM or TDGMM even when the latter are true. Hence, there is a delicate tradeoff and sometimes one will want to fit a model which one knows to be incorrect.

When applied to the classification of the sleep states in mice, our procedure provides more accurate differentiation of the NREM and REM sleep states compared to any previous method in the field. The improvements in REM classification are especially beneficial, as the dynamics of REM sleep are of special interest to sleep scientists. Furthermore, our procedure provides substantial improvements in capturing the sleep state bout duration distributions relative to other methods.

However, the improvements were not uniform across all metrics nor were other methods completely uncompetitive. In particular, when a variant of a sliding window approach using forward and backward moving averages of the covariates was used, the PrAGMaTiSt was still quite strong on the "local" error rate task but suffered both absolutely and relative to other methods on the task of estimating sleep state bout duration distributions.

8.2 Future Research

Future research is proceeding on several fronts, with three main focuses: further exploration of the methodology as it stands, generalization of the methodology, and advances specific to sleep science. In terms of exploring the current methodology, a more elaborate series of simulations that follow an experimental design framework would be useful. We have started to get at this by varying the noise level, but there are other aspects of the simulations which would also be worth varying such as the number of the noise dimensions, the complexity of the underlying conditional covariate distributions, and the complexity of the time series model. In particular, if these latter two could be varied in a more "continuous" fashion, we could conduct an analysis of variance of our errors in order to determine which aspects of the simulation were having the largest effects. This may also provide some insight into whether there are summary statistics which could be calculated from the data in order to guide whether more complex variants of the model such as the TDGMM or whether the simpler variants such as the first order one should be used in practice.

Relatedly, it would be useful to understand exactly why the combined true and estimated models in Simulation 1D performed so poorly, particularly as the noise level increased. This may provide substantial intuition about the model which would also be useful for selecting between simpler and more complicated structures.

More advanced studies of computation times particularly relative to other methods are also worth considering. Anecdotally, all of the computations for our most

complicated method (i.e., RF + TDGMM) took well under a day when applied to the mice data. On the other hand, MALLET and TreeCRF, the competitors which are most similar in the sense that they are sequential methods, each took several days. Thus, the PrAGMaTiSt provides substantial computational savings as well as performance enhancements.

There are a number of extensions of the model we are considering. For instance, a hybrid VLMM/TDGMM approach that uses a VLMM in the "spike" followed by a TDGMM in the "slab" is currently underway. Also, extending our model from a one-dimensional lattice to two or more dimensions (i.e., from time to space) is another project under consideration.

We are also interested in shrinkage and hierarchical extensions of our model. For instance, modeling of the transition probabilities, duration distributions, and conditional class probability functions could be done in a way that pools, for example, mice together in some hierarchical fashion. This might provide better estimates at least of the variance of our fits and perhaps of the fit itself.

It would be also be interesting to consider some variations in the Y_t and X_t. For example, it would be useful to extend to cases where the X_t are either measured with error or, more saliently, when they are set strategically to some levels. Also, the case where the state space S of the Y_t is at least partially unknown would be interesting to consider (this is, of course, different than the fully unsupervised case).

We are also working on methodology to improve conditional class probability

estimates (i.e., the non-sequential case); since these estimates feed directly into our model, improvements in these non-sequential estimates would also improve our sequential estimates.

Continuing efforts on the sleep application are advancing on two fronts. First, various devices are being utilized to provide additional covariates which more accurately capture differences among NREM and REM (e.g., heart rates, respiratory rates, eye movements, etc.). Second, more accurate modeling of the Y_t dependence structure (e.g., VLMM in the "spike" followed by TDGMM in the "slab") may indeed provide further benefits.

Nonetheless, the PrAGMaTiSt provides a real advance for sequential classification. It allows (i) quick and efficient estimation of sequential models that incorporate (ii) complex Y_t dependencies as well as the benefit of using (iii) any base classifier. By tying these three features together in a way that preserves their fundamental independence, our method can directly benefit from advances on any individual front in a way that standard sequential methods such as CRFs cannot. As we have seen with both simulated data and sleep data, these advances can yield tremendous benefit in practice.

Bibliography

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control* **19**, 6, 716–723.

Anderson, E. (1935). The irises of the gaspe peninsula. *Bulletin of the American Iris Society* **59**, 2–5.

Bayes, T. (1764). Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London* .

Bickel, P. J., Ritov, Y., and Zakai, A. (2006). Some theory for generalized boosting algorithms. *Journal of Machine Learning Research* **7**, 705–732.

Bostrom, H. (2007). Estimating class probabilities in random forests. *Proceedings of the Sixth International Conference on Machine Learning and Applications* 211–216.

Bostrom, H. (2008). Calibrating random forests. *Proceedings of the Seventh International Conference on Machine Learning and Applications* .

Bottou, L. (1991). *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. Ph.D. thesis, Université de Paris XI, Orsay, France.

Breiman, L. (1996). Bagging predictors. *Machine Learning* **24**, 123–140.

Breiman, L. (1998). Arcing classifiers. *Annals of Statistics* **26**, 801–849.

Breiman, L. (2000). Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics* **28**, 374–377.

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

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth, New York.

Brillinger, D. R., Morettin, P. A., Irizarry, R. A., and Chiann, C. (2000). Some wavelet-based analyzes of markov chain data. *Signal Processing* **80**, 1607–1627.

Buhlmann, P. and Wyner, A. J. (1999). Variable length markov chains. *Annals of Statistics* **27**, 2, 480–513.

Buja, A. (2000). Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics* **28**, 387–291.

Buja, A. and Stuetzle, W. (2006). Observations on bagging. *Statistica Sinica* **16**, 2, 323–352.

Buja, A., Stuetzle, W., and Shen, Y. (2005). Loss functions for binary probability estimation and classification: Structure and applications. *http://www-stat.wharton.upenn.edu/ buja/PAPERS/paper-proper-scoring.pdf (Under review)* .

Candes, E. and Tao, T. (2007). The dantzig selector: Statistical estimation when p is much larger than n the dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics* **35**, 6, 2313–2351.

Chan, P. and Stolfo, S. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* 164–168.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* **16**, 321–357.

Chawla, N. V., Lazarevic, A., Hall, L. O., and Bowyer, K. W. (2003). Smoteboost: Improving prediction of the minority class in boosting. *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases* 107–119.

Cohen, W. W. and Singer, Y. (1999). A simple, fast, ad effective rule learner.

Proceedings of the 16th National Conference on Artificial Intelligence (AAAI) 335–342.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, B* **39**, 1–38.

Dettling, M. and Buhlmann, P. (2003). Boosting for tumor classification with gene expression data. *Bioinformatics* **19**, 1061–1069.

Dietterich, T. G. (2002). Machine learning for sequential data: A review. In T. Caelli, ed., *Lecture Notes in Computer Science*. Springer-Verlag.

Dietterich, T. G., Ashenfelter, A., and Bulatov, Y. (2004). Training conditional random fields via gradient tree boosting. *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)* .

Djuric, P. M. and Chun, J.-H. (2002). An mcmc sampling approach to estimation of nonstational hidden markov models. *IEEE Transactions on Signal Processing* **50**, 5, 1113–1123.

Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation via wavelet shrinkage. *Biometrika* **81**, 425–455.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. John Wiley and Sons, 2nd edn.

Efron, B. (1975). The efficiency of logistic regression compared to normal discriminant analysis. *Journal of the American Statistical Association* **80**, 892–898.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics* **7**, 1, 1–26.

Efron, B. and Tibshirani, R. (1994). *An Introduction to the Bootstrap*. Chapman and Hall (CRC).

Elkan, C. (2001). The foundations of cost-sensitive learning. *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)* 973–978.

Estabrooks, A., Jo, T., and Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced datasets. *Computational Intelligence* **20**, 18–36.

Fan, W., Stolfo, S., Zhang, J., and Chan, P. (1999). Adacost: Misclassification cost-sensitive boosting. *Proceedings of the 16th International Conference on Machine Learning* 97–105.

Ferguson, J. D. (1980). Variable duration models for speech. *Proceedings of Symposium on the Application of Hidden Markov Models to Text and Speech* 143–179.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics* **7**, II, 179–188.

Foster, D. P. and George, E. I. (1994). The risk inflation criterion for multiple regression. *Annals of Statistics* **22**, 4, 1947–1975.

Foster, D. P. and Stine, R. A. (1997). An information theoretic comparison of model selection criteria. Tech. Rep. 1180, Center for Mathematical Studies in Economics and Management Science, Northwestern University.

Foster, D. P. and Stine, R. A. (2004). Variable selection in data mining: Building a predictive model for bankruptcy. *Journal of the American Statistical Association* **99**, 303–313.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation* **121**, 2, 256–285.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, 148–156.

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

Freund, Y. and Schapire, R. E. (2000). Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics* **28**, 391–393.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics* **29**, 5, 1189–1232.

Friedman, J. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis* **38**, 4, 367–378.

Friedman, J., Hastie, T., and Tibshirani, R. (2000a). Additive logistic regression: a statistical view of boosting. *Annals of Statistics* **28**, 337–374.

Friedman, J., Hastie, T., and Tibshirani, R. (2000b). Rejoinder: Additive logistic regression: a statistical view of boosting. *Annals of Statistics* **28**, 400–407.

Fujikia, N., Chenga, T., Yoshinoa, F., and Nishino, S. (2009). Specificity of direct transition from wake to rem sleep in orexin/ataxin-3 transgenic narcoleptic mice. *Experimental Neurology Experimental Neurology Experimental Neurology Experimental Neurology* **217**, 1, 46–54.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transaction on Pattern Analysis and Machine Intelligence* **6**, 721–741.

George, E. I. and Foster, D. P. (2000). Calibration and empirical bayes variable selection. *Biometrika* **87**, 4, 731–747.

George, E. I. and McCulloch, R. E. (1997). Approaches for bayesian variable selection. *Statistica Sinica* **7**, 339–373.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edn.

Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika* **57**, 97–109.

Hoerl, A. E. (1962). Application of ridge analysis to regression problems. *Chemical Engineering Progress* **58**, 54–59.

Hothorn, T. and Buhlmann, P. (2002). Model-based boosting in high dimensions. *Bioinformatics* **22**, 22, 2828–2829.

Jiang, W. (2004). Process consistency for adaboost. *Annals of Statistics* **32**, 13–29.

Joshi, M., Kumar, V., and Agarwal, R. (2001). Evaluating bossting algorithms to classify rare classes: Comparison and improvements. *Proceedings of the First IEEE International Conference on Data Mining (ICDM)* 257–264.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)* .

Levinson, S. E. (1981). Continuously variable duration hidden markov models for automatic speech recognition. *Computer Speech and Language* **1**, 1, 29–45.

Lugosi, G. and Vayatis, N. (2004). On the bayes-risk consistency of regularized boosting methods. *Annals of Statistics* **32**, 30–55.

Mallows, C. L. (1973). Some comments on cp. *Technometrics* **15**, 661–675.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence* 403–410.

McCallum, A., Freitag, D., and Pereira, F. C. (2000). Maximum entropy markov models for information extraction and segmentation. *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)* 591–598.

McShane, B. B. (2007). More evidence contrary to the statistical view of boosting. Tech. rep., Department of Statistics, The Wharton School, University of Pennsylvania.

McShane, B. B., Galante, R. J., Jensen, S. T., Naidoo, N., Pack, A. I., and Wyner, A. (2010). Characterization of the bout durations of sleep and wakefulness. *Under Review* .

Mease, D. and Wyner, A. (2008). Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research* **9**, 131–156.

Mease, D., Wyner, A., and Buja, A. (2007). Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research* **8**, 409–439.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953).

Equation of state calculations by fast computing machines. *Journal of Chemical Physics* **21**, 1087–1092.

Pack, A. I., Galante, R. J., Maislin, G., Cater, J., Metaxas, D., Lu, S., Zhang, L., Smith, R. V., Kay, T., Lian, J., Svenson, K., and Peters, L. L. (2007). Novel method for high-throughput phenotyping of sleep in mice. *Physiological Genomics* **28**, 232–238.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, CA.

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

Ridgeway, G. (2000). Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics* **28**, 393–400.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica* **14**, 5, 465–471.

Savage, L. J. (1971). Elicitation of personal probabilities and expecations. *Journal of the American Statistical Association* **66**, 336, 783–801.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning* **5**, 197–227.

Schapire, R. E. and Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* .

Schwarz, E. G. (1978). Estimating the dimension of a model. *Annals of Statistics* **6**, 2, 461–464.

Sin, B. and Kim, J. H. (1995). Nonstationary hidden markov model. *Signal Processing* **46**, 31–46.

Smyth, P. (1994). Markov monitoring with unknown states. *IEEE Journal of Selected Areas in Communications, Special Issue on Intelligent Signal Processing for Communications* **12**, 1600–1612.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and van der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **64**, 4, 583–639.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B* **58**, 267–288.

Ting, K. M. (2000). A comparative study of cost-sensitive boosting algorithms.

Proceedings of the 17th International Conference on Machine Learning (ICML 2000) 983–990.

Vaseghi, S. V. (1995). State duration modeling in hidden markov models. *Signal Processing* **41**, 31–41.

Zadrozny, B. and Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. *Proceedings of the Eighteenth International Conference on Machine Learning* 609–616.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, B* **67**, 301–320.