Exploring a New Method for Classification with Local Time Dependence

Blakeley B. McShane[1]

Abstract

We have developed a sophisticated new statistical methodology which allows machine learning methods to model time series data. The methodology builds upon recent advances in machine learning, specifically random forests, combined with hidden Markov models to account for local time-dependence. We evaluate the properties of our model via a careful simulation study.

1 Introduction

Conventional machine learning methods such as AdaBoost (Freund and Schapire, 1996) have proven extremely successful at the task of classification on a wide variety of problems. These methods are so successful because they focus on minimizing classification error. As a consequence, these algorithms are very powerful, but they are also limited. For example, they assume *i.i.d* data which can be inappropriate if the states are correlated across time. Moreover, while often giving good class estimates, their conditional class probability estimates are often overfit, tending towards zero or one (Mease *et al.*, 2007). In addition, they focus on a uniform loss function over misclassifications. Finally, many of these algorithms were developed for the binary classification setting and do not naturally accommodate a multi-class setting.

While more recent methods such as random forests (Breiman, 2001) overcome some of these limitations, conventional methods still can fail when, for example, there is one state that is very rare and where there is temporal structure in the data. These unfortunate features are prevalent in the problem that served as the motivation for this research (automated scoring of sleep states into three states (REM, NREM, and WAKE) based on covariates culled from video data). In this paper, we focus on the time series issue and solve it by blending machine learning methods with a hidden Markov model. Our focus is on describing the method and investigating various properties of the model through a careful simulation study, though we provide a brief description of the motivation for the method.

[1]Department of Statistics, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104 mcshaneb@wharton.upenn.edu

2 Motivating Problem

The state-of-the-art, gold standard methodology for the study of sleep behavior is invasive, expensive, and time-consuming. First, a wire which captures E.E.G. and E.M.G. signals is implanted into a mouse's head. Then, researchers must wait ten to fourteen days during which the mouse recovers from the surgery. Next, they track the E.E.G. and E.M.G. signals for twenty-four hours. They then break up the day into 8,640 epochs, each ten seconds in length. Finally, these epochs are manually classified into three sleep stages: REM sleep, Non-REM sleep, and awake. An automated, algorithmic system therefore has the potential to be advantageous on many practical levels.

The method described in this paper has been used to predict the sleep state of mice based on video data[2]. We record the mice at ten frames per second and convert the video to numeric data by a computer program that tracks the mouse in each frame of the video. The tracking program calculate variouss summary statistics for each frame (velocity, aspect ratio, and size of the mouse) and we use the intra-epoch means and standard deviations of the three summary statistics over the 100 frames within each epoch as our covariates. We also include covariate which indicates whether light in the mouses cage was turned on (7:00AM - 7:00PM) or not since light has a substantial effect on a mouses sleep behavior.

Our method, described and investigated in detail below, performs sufficiently well that sleep researchers are now using it in lieu of manual scoring in several important applications. By using covariates culled from *video* data, we have saved sleep researchers vast sums of time and money by avoiding the difficult implantation procedure in addition to the obvious savings from replacing the manual scoring process with an automated one. An additional benefit of automation is that it allows high throughput, large sample studies which were previously prohibitive.

3 Methodology

3.1 Our Method

Suppose we observe a process as in Figure 1. In the first part, we observe (potentially with error) the response $\{Y_t\}$ as well as (potentially multivariate) covariates $\{X_t\}$. In the second part, we observe only the covariates. Let $\{Y_t\}$ take on values in $S = \{S_1, ..., S_N\}$ and furthermore assume $\{Y_t\}$ is a time-homogeneous first order Markov Chain. Also, assume there are N probability measures μ_i where $\mu_i(x) = \mathbb{P}(X_t = x | Y_t = S_i)$. Finally, assume an initial state distribution π where $\pi_i = \mathbb{P}(Y_1 = S_i)$.

This structure is very similar to a hidden Markov model (Rabiner, 1989). In this setting one

[2]For more details on the application, see McShane *et al.* (2009)

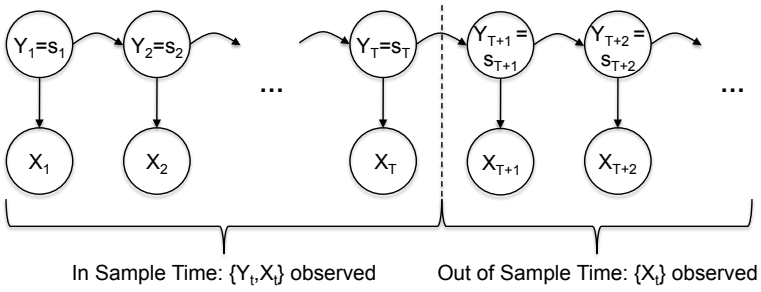

Figure 1: Sample Data Structure.

needs to estimate three things, the initial state distribution π, the transition probability matrix $A = a_{ij}$ where $a_{ij} = \mathbb{P}(Y_t = S_j | Y_{t-1} = S_i)$, and the probability measures μ governing the covariates.

While estimation of the first two is fairly straightforward (*e.g.*, use empirical frequencies), estimation of the N probability measures μ_i can be extremely difficult, particularly when X_t is high dimensional and the signal is complicated. Machine learning methods thrive in such settings but are meant for univariate, categorical responses. We can use Bayes' Theorem to transform the problem:

$$\begin{aligned}\mu_i(x) &= \mathbb{P}(X_t = x | Y_t = S_i) \\ &= \frac{\overbrace{\mathbb{P}(Y_t = S_i | X_t = x)}^{Machine\ Learning} \quad \overbrace{\mathbb{P}(X_t = x)}^{Normalizing\ Constant}}{\underbrace{\mathbb{P}(Y_t = S_i)}_{Empirical\ Frequency}}\end{aligned}$$

We can use machine learning methods to estimate the first term in the numerator and we can ignore the second term. The denominator can be estimated by empirical frequencies as we did for the initial state distribution and the transition probability matrix.

Thus, we have transformed the problem of estimating N multi-dimensional probability density functions into a problem of estimating an N-dimensional probability vector conditional on covariates and N marginal probabilities. That is, our task has shifted from estimating $(\hat{\pi}, \widehat{A}, \hat{\mu})$ to estimating $(\hat{\pi}, \widehat{A}, \hat{f}, \hat{m})$ where $f_i(x_t) = \mathbb{P}(Y_t = S_i | X_t = x_t)$ and $m_i = \mathbb{P}(Y_t = S_i)$[3]

A "naive" machine learners are limited in this setting because they can, at best, estimate f_i. We can do better if there is time series dependence because we can estimate $\gamma_t(i) \equiv \mathbb{P}(Y_t = S_i | X_1, X_2, ..., X_T)$.

[3] m_i is the marginal state probability of the first out of sample datapoint in contrast to π_i which is the marginal probability of any datapoint. In the case that out of sample forecast begins at a random point, the distribution m is equivalent to the distribution π and therefore one can set $\hat{m} = \hat{\pi}$. In the case that the out of sample forecast continues on from some known point, as is illustrated in Figure 1, the distribution m comes from the transition probability matrix A and therefore one can set $\hat{m}$ equal to the appropriate row of $\widehat{A}$.

The main idea, turning a generative HMM problem into a discriminative problem, has been considered before (Smyth, 1994). However, it has typically been applied in settings with a much higher signal to noise ratio and less frequent transitions, both of which make overfit estimates of the conditional class probability function less problematic. In these settings, combining machine learning methods with an HMM model for time dependency serves mostly to locally smooth out probability estimates. However, in a noisy setting such as our sleep application, overfit estimates of the conditional class probabilities by the base method can cause insensitivity to the remaining time series dependency in the data. In addition, in our setting, the Markovian parameters are unknown and therefore must be estimated from the data.

3.2 Estimation and Calculation

Estimating each component of $\widehat{M} = (\hat{\pi}, \widehat{A}, \hat{f}, \hat{m})$ is straightforward. However, $\gamma_t(i)$ is a very complicated function so calculating it conditional on $\widehat{M}$ seems more difficult. Fortunately, we can make use of a modification of the forward backward algorithm commonly used in hidden Markov models (Rabiner, 1989) to do so.

First, define $\alpha_t(i) = \mathbb{P}(X_1 = x_1, X_2 = x_2, ..., X_t = x_t, Y_t = S_i|\widehat{M})$. Then, we can solve for these inductively as follows:

1. **Intialization:** $\alpha_1(i) = \hat{\pi}_i \frac{\hat{f}_i(x_1)}{\hat{m}_i}, \quad 1 \leq i \leq N$
2. **Induction:** $\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i)\hat{a}_{ij}\right] \frac{\hat{f}_j(x_{t+1})}{\hat{m}_j}$,
   $1 \leq t \leq T-1; 1 \leq j \leq N$
3. **Termination:** $\mathbb{P}(X_1 = x_1, X_2 = x_2, ..., X_T = x_t|\widehat{M}) = \sum_{i=1}^N \alpha_T(i)$.

Next, define $\beta_t(i) = \mathbb{P}(X_{t+1} = x_{t+1}, ..., X_T = x_T|Y_t = S_i, \widehat{M})$. Again, we can solve for these inductively as follows:

1. **Intialization:** $\beta_T(i) = 1, \quad 1 \leq i \leq N$
2. **Induction:** $\beta_t(i) = \sum_{j=1}^N \hat{a}_{ij}\beta_{t+1}(j)\frac{\hat{f}_j(x_{t+1})}{\hat{m}_j}$,
   $t = T-1, T-2, ..., 1; 1 \leq i \leq N$

Finally, we can recover $\mathbb{P}(Y_t = S_i|X_1, X_2, ..., X_T, \widehat{M})$ as follows:

$$\begin{aligned}
\alpha_t(i)\beta_t(i) &= \mathbb{P}(X_1, X_2, ..., X_t, Y_t = S_i|\widehat{M}) \cdot \mathbb{P}(X_{t+1}, ..., X_T|Y_t = S_i, \widehat{M}) \\
&= \mathbb{P}(Y_t = S_i|\widehat{M}) \cdot \mathbb{P}(X_1, X_2, ..., X_t|Y_t = S_i, \widehat{M}) \cdot \mathbb{P}(X_{t+1}, ..., X_T|Y_t = S_i, \widehat{M}) \\
&= \mathbb{P}(Y_t = S_i|\widehat{M}) \cdot \mathbb{P}(X_1, X_2, ..., X_T|Y_t = S_i, \widehat{M}) \\
&= \mathbb{P}(Y_t = S_i|X_1, ..., X_T, \widehat{M}) \cdot \underbrace{\mathbb{P}(X_1, X_2, ..., X_T|\widehat{M})}_{Normalizing\ Constant}
\end{aligned}$$

Therefore,

$$\mathbb{P}(Y_t = S_i | X_1, ..., X_T, \widehat{M}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \equiv \gamma_t(i).$$

If we want to classify Y_t at times $t = 1, 2, ..., T$ given $X_1, X_2, ..., X_T$, we have two options:

- **Modal Sequence**: At each t, we choose $\widehat{Y}_t$ to be the state that has the maximum probability conditional on the full set of covariates: $\widehat{Y}_t = \arg\max_i \gamma_t(i)$. That is, we choose the most likely state on a time period by time period basis.

- **Viterbi Sequence**: We can use the Viterbi Algorithm to find the single best sequence, that is the $\{\widehat{Y}_1, ..., \widehat{Y}_T\}$ such that $\mathbb{P}(\widehat{Y}_1, ..., \widehat{Y}_T | X_1, ..., X_T)$ is maximized.

When classification error is the loss function, the Modal Sequence performs best. Under other loss functions, the Viterbi Sequence (or some other sequence) could be optimal. However, since our focus is on probability estimation, we often use the $\gamma_t(i)$ directly.

3.3 Summary

In the setting we have described, $\mathbb{P}(Y_t | Y_1, ..., Y_{t-1}, Y_{t+1}, ..., Y_T, X_1, ..., X_T)$ is equivalent to $\mathbb{P}(Y_t | Y_{t-1}, Y_{t+1}, X_t)$. If Y_{t-1} and Y_{t+1} were known, naive machine learning algorithms would work by augmenting the covariates: $\widetilde{X}_t = \{X_t, Y_{t-1}, Y_{t+1}\}$. Since the $\{Y_{t-1}\}$ and $\{Y_{t+1}\}$ are unknown, we must go with the next best thing: $\mathbb{P}(Y_t | X_1, ..., X_T)$. A naive machine learner would have to augment the covariates to be $\widetilde{X}_t = \{..., X_{t-1}, X_t, X_{t+1}, ...\}$. This is infeasible since X_t is already high dimensional. Our method allows the machine learner to calculate $\mathbb{P}(Y_t | X_1, ..., X_T)$ while only fitting on X_t.

In sum, our methodology takes a machine learning algorithm as an input and returns an enhanced version of that algorithm which accounts for time series dependence. That is, we fit on $Y_t | X_t$ but get $Y_t | X_1, ...X_T$. The alternatives to our model are not as good:

- Use $Y_t | X_t$ thereby ignoring the time series structure.

- Augment the covariates $\widetilde{X}_t = \{..., X_{t-1}, X_t, X_{t+1}, ...\}$ but face the curse of dimensionality when trying to estimate $Y_t | \widetilde{X}_t$.

- Try and estimate the measures $\{\mu_i\}_{i=1}^{N}$, an extremely difficult task for multidimensional X_t.

4 Simulations

We provide two simulations to show how our model compares to various competitors and compared to the truth. We repeat each simulation 1,000 times and average the results over these repetitions.

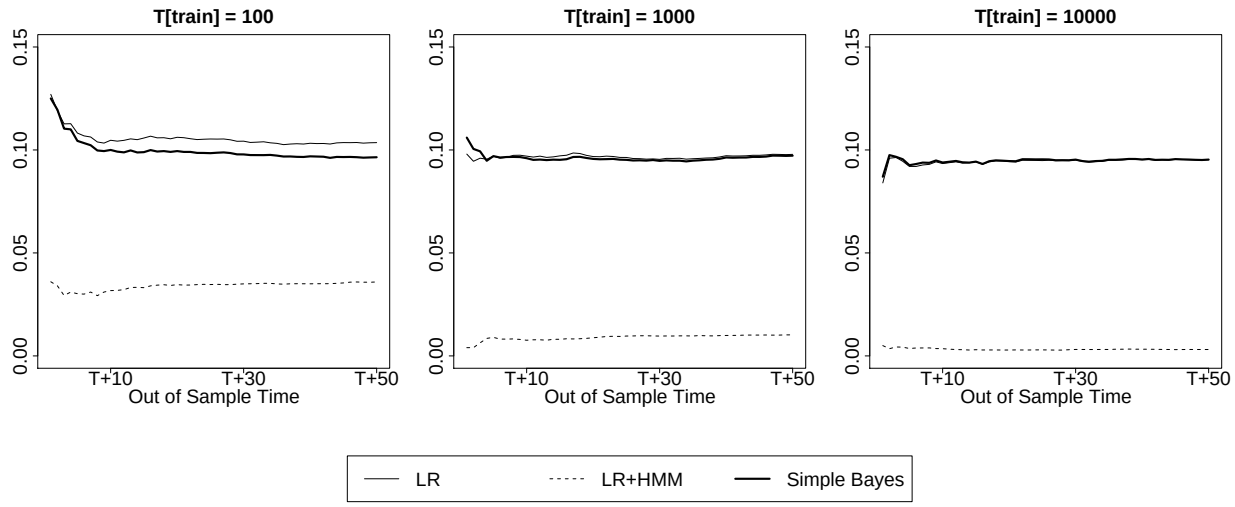

Figure 2: Classification Error Relative to Full Bayes Classifier

We are interested in how well the model estimates the true classes but also how well it recovers the true probabilities. In this setting, there are two Bayes rules to consider. The first one is the simple Bayes rule commonly used in machine learning, $\mathbb{P}(Y_t = T_i|X_t = x_t)$. Due to the time series dependence, this decision rule can be beat. The other Bayes Rule is the full Bayes rule, $\mathbb{P}(Y_t = T_i|X_1 = x_1, ..., X_T = x_T)$. This rule gives the true conditional probabilities and hence it cannot be beat.

4.1 Simulation I

The first simulation is extremely straightforward and is meant more as an illustration of the method than as a realistic example. In this case, $S = \{0, 1\}$, $\pi = \{.5, .5\}$ and $A = \begin{pmatrix} .8 & .2 \\ .3 & .7 \end{pmatrix}$. Finally, $\mathbb{P}(X|Y_t = i) \sim N(\mu = Y_t, \sigma = .5)$. We set the training sample size to 100, 1,000, and 10,000.

Since the covariate is univariate and the decision boundary is linear, we use logistic regression as our "machine learning method" since many machine learning methods behave strangely in one dimension (*e.g.*, AdaBoost becomes one-nearest neighbor).

As can be seen from examining Figure 2, logistic regression with the HMM adaptation replicates the full Bayes rule for classification (i.e., zero error). On the other hand, plain logistic regression is imitating the simple Bayes rule which can be improved upon in this setting. Since the time series structure resembles that of Figure 1, there is substantial information contained in the value of the last in-sample datapoint Y_T, data which is ignored by logistic regression and the simple Bayes rule which do not take it into account. Thus, they tend to perform worse initially and improve as

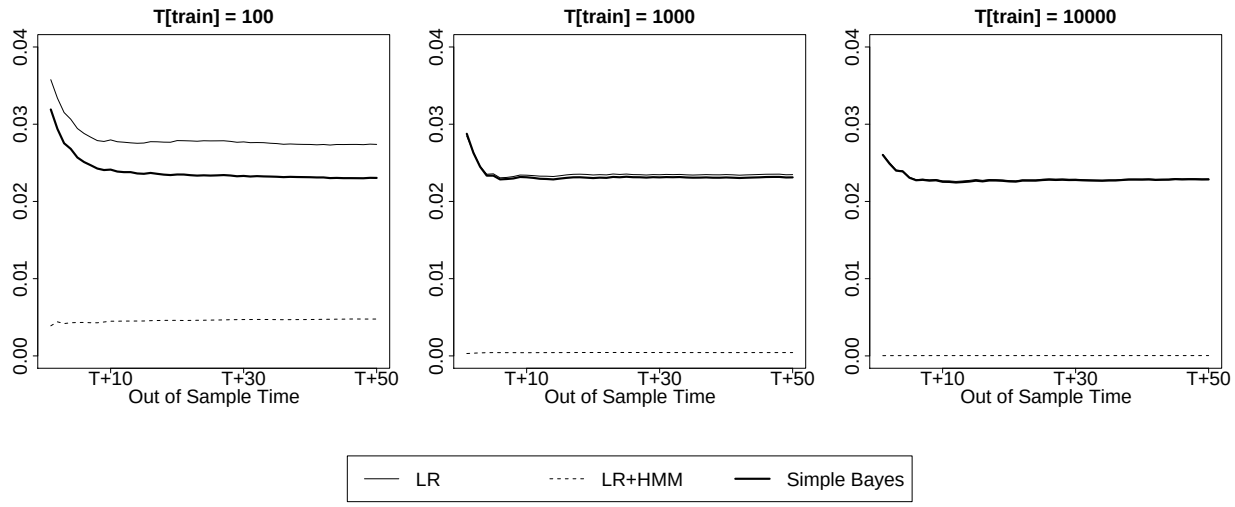

Figure 3: Squared Probability Error Relative to Full Bayes Probabilities.

the value of conditioning on Y_T diminishes over time.

Since we are interested in probability estimates as well as classifications, we plot in Figure 3 the squared difference in probabilities between the three models and the full Bayes rule[4]. With enough training points, logistic regression with the HMM adaptation replicates the probability estimates of the full Bayes rule. Likewise, plain logistic regression is imitating the simple Bayes rule.

In sum, the simple Bayes rule and logistic regression perform poorly since they do not take into account the time series nature of the data. On the other hand, our model–logistic regression augmented with the HMM–replicates both the class estimates and the probability estimates of the full Bayes rule. This fact, as will be seen next, is highly dependent on how well the machine learning method estimates $\mathbb{P}(Y_t = S_i | X_t = x_t)$ (*i.e.*, $\hat{f}(x)$). We will see in the next simulation that overfit estimates of $\hat{f}$ will cause the classifications produced by our model to be the same as the base model and cause the probability estimates to be poor.

4.2 Simulation II

In the second simulation, we again set $S = \{0, 1\}$, $\pi = \{.5, .5\}$ and $A = \begin{pmatrix} .8 & .2 \\ .3 & .7 \end{pmatrix}$. This time, however, the distributions μ_i are much more complicated. We let $\{X_t\}$ be a 10-dimensional vector. Eight of these dimensions are $Unif(0, 1)$ noise dimensions. The two dimensions with signal are distributed according to the patterns in Figures 4. We set the sample size progressively

[4]In addition to squared error loss, we also considered negative log likelihood loss, exponential loss, and Kullback-Leibler Divergence. Qualitatively, the same patterns held for all four measures.

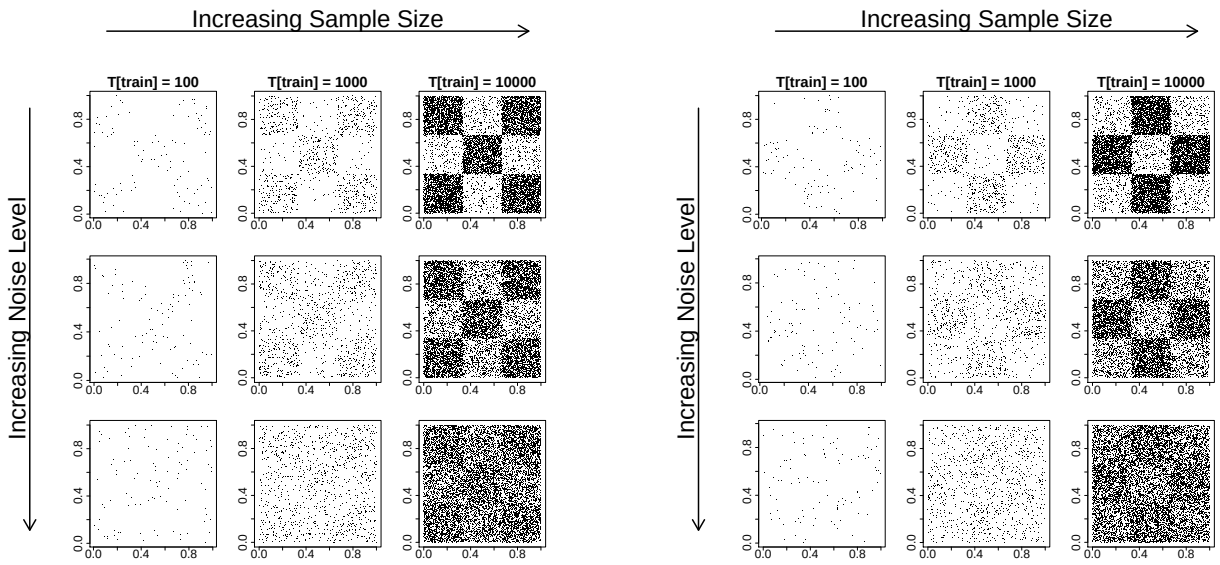

Figure 4: The left panel gives the distribution of $X^{(1,2)}$ given $Y = 0$ for the nine simulated scenarios. The right panel gives the distribution of $X^{(1,2)}$ given $Y = 1$.

to 100, 1,000, and 10,000 and consider three progressively more difficult settings. We also use three machine learning methods as our base algorithm: logistic regression, AdaBoost, and random forests.

Before showing the results of the simulation, we want touch on how the machine learning methods should perform and what should happen in each of the nine cells. First, we expect logistic regression to perform poorly (at both classification and probability estimation) since it is a linear method and the data is clearly non-linear. We expect AdaBoost to perform well on classification but poor on probability estimation since it tends to overfit on probabilities. Finally, we expect random forests to perform the best.

For the three cells of Figure 4 with training data size equal to 100, we really do not expect any algorithm to perform well. The structure is simply too complicated for effective learning in that environment. Furthermore, for the low noise setting, we expect the enhanced versions of the base algorithms will not perform much better than the base algorithms themselves. That is because in the low noise setting, the data pattern is so stark that incorporating time series effects does not add much. Similarly, though for an opposite reason, we do not expect to perform much better than the base models in the high noise setting. This is because there is so much noise that it is hard to beat a model which predicts the more frequent class every time.

Hence, the ”sweet spot” for our model is the medium noise middle row of each panel of Figure 4, particularly the second and third cells of the middle row where there is sufficient data for

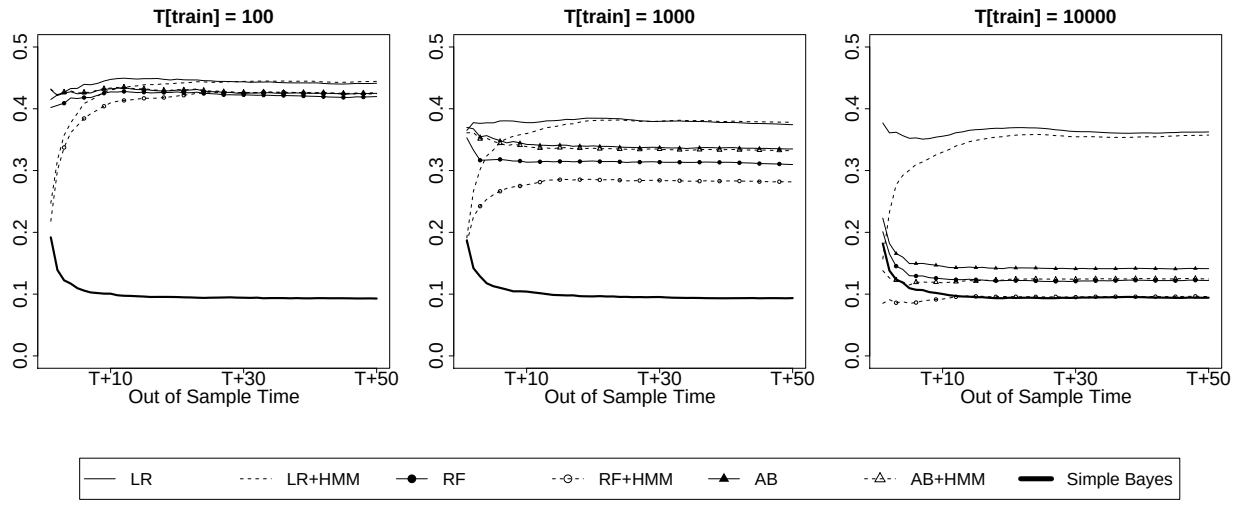

Figure 5: Classification Error Relative to Full Bayes Classifier For Medium Noise Setting.

estimation. As a consequence, we focus on the results for this row [5].

Figure 5 gives the classifications produced by our various models as compared to those produced by the full Bayes rule. Not surprisingly, all methods except the simple Bayes rule perform pretty poorly in the first plot where the training set is 100: there is simply not enough data for these methods to learn the complicated structure of Figure 4 as well as the time series structure. The simple Bayes rule, however, performs reasonably well which is not all that surprising when one considers that this rule knows the true structure of Figures 4, a complex structure which the other methods must learn.

As the training set size increases, we see that the classification error for AdaBoost and random forests begins to approach that of the simple Bayes rule. Moreover, since there is sufficient data to learn the time series structure, the versions of AdaBoost and random forests with the HMM perform better than the base models. Surprisingly, however, even with the HMM, these methods cannot beat the simple Bayes rule: complete knowledge of the complex conditional data structure of Figures 4 is simply very hard to beat.

Figure 6 gives the squared probability error for each method relative to the Full Bayes probabilities. Again, all methods with the exception of the simple Bayes rule are quite poor when the training set is of size 100 (AdaBoost is so poor at probability estimation that its squared error is off the charts for the first two plots). The simple Bayes rule performs well because knows the complicated structure and for our choice of simulation parameters $\mathbb{P}(Y_t = S_i|X_t = x_t)$ is not all that different from $\mathbb{P}(Y_t = S_i|X_1 = x_1, ..., X_T = x_T)$.

[5]The results for the other two rows can be found in the Appendix.

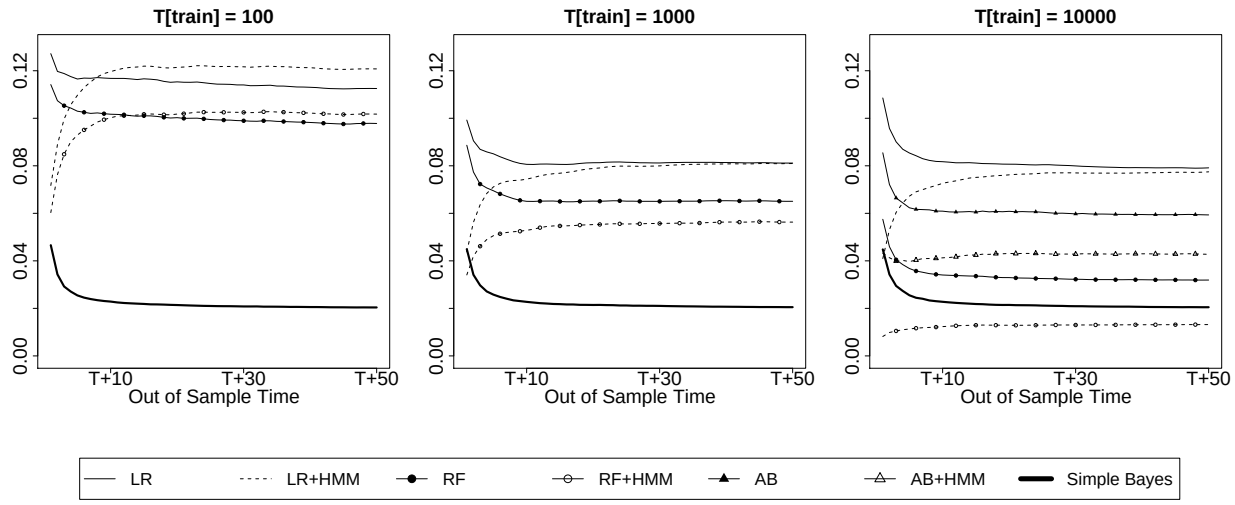

Figure 6: Squared Probability Error Relative to Full Bayes Probabilities For Medium Noise Setting. AdaBoost does not appear in some plots because it performs poorly relative to the other methods.

However, with increasing data, it is clear that adding the HMM does indeed improve the probability estimates of the base models (except logistic regression whose estimates are so poor that we cannot improve them much). In fact, the random forest performs so strongly with sufficient data that the HMM allows it to beat out even the simple Bayes rule, an impressive feat.

In conclusion, we see that logistic regression performs horribly because it cannot fit the function. The HMM helps slightly in the early portion of the hold-out sample, however. AdaBoost performs more or less the same regardless of whether or not the HMM is used. This is because AdaBoost produces probability estimates which tend to zero or one, causing insensitivity to the HMM. As a consequence, AdaBoost does well on the classification task but poorly on the probability estimation task. Random forests with the HMM performs very well.

The simple Bayes rule is hard to beat because the data structure is very complicated. In such settings, the true $\mathbb{P}(Y_t = S_i|X_t)$ can beat estimates $\widehat{\mathbb{P}}(Y_t = S_i|X_1, ..., X_T)$. However, we must keep in mind that the simple Bayes rule is not a realistic competitor because it can never be known in real settings. The fact that our model beats it in some settings and is comparable in many others is quite impressive.

5 Discussion

In conclusion, we have developed a methodology which takes a machine learning method as an input and returns a machine learning method which accounts for time series structure. This method-

ology performed extremely well in various simulations, even matching the full Bayes rule in some settings.

There are two apparent limitations to our model. In difficult settings, where it is hard to determine $\mathbb{P}(Y_t = S_i|X_t = x_t)$, our method performs less well even though it still beats all realistic competitors. This suggests our model is very sensitive to the quality of the input machine learning method which produces the estimates of $\mathbb{P}(Y_t = S_i|X_t = x_t)$. Further confirmation of this comes from the fact that AdaBoost, known for producing estimates of $\mathbb{P}(Y_t = S_i|X_t = x_t)$ that are close to zero or one, is insensitive to our method in some cases. Another limitation of our model is the first order Markov chain assumption. While first order Markov chains form a rich class and though they are able to approximate a much more general time series dependence structures, there is room for improvement. In particular, the Geometric holding time distribution implied by a first order Markov chain seem unrealistic in many situations. Thus, future research should proceed in two ways: (i) producing better machine learning probability estimates $\mathbb{P}(Y_t = S_i|X_t = x_t)$ and (ii) relaxing the first order Markov chain assumption.

As for the first, a theoretical improvement in classification technologies would be of general benefit to all machine learning algorithms and is not specific to our method. As mentioned above, most state-of-the-art classification tools are very good at classifying the most likely state but they are not so good at giving the probabilities of all the states. While random forests is noted for giving reasonably accurate probability estimates, the estimates seem to be unsatisfactory for some datasets. Namely, the probabilities are not appropriately calibrated: when the model says the mouse should be in a given state with probability p, the empirical frequency that it is in that state is not p. Improving the probability estimates would obviously improve our classifications. Moreover, they would make the hidden Markov models more competitive as those probability estimates enter directly into the Markov Model.

Regarding the second improvement suggested above, it is specific to our method: while many non-Markovian processes can be approximated by a Markov model, it would be useful to accommodate even more general dependence structures. Second, third, and higher order Markov Chains are a natural generalization; in fact, these can be implemented with minor variations to the current algorithm. Furthermore, we are currently implementing a generalized hidden Markov model to capture the time dependencies in the data. A GHMM is like an HMM except (i) each state has its own arbitrary holding time distribution (as opposed to the Geometric distribution implied by an HMM) and (ii) self-transitions are not allowed (since they are accounted for by the arbitrary holding time distribution). Another variant we are working on is a Variable Length hidden Markov model (a higher order Markov model with variable memory length, the length of which depends on the sequence of previous states).

In sum, our method is very general, can fit a wide class of problems, and will perform better

than conventional machine learning methods in the presence of time series dependence. When the conditional class probability function can be described by the base machine learning method and sufficient training data is provided, our method recovers the true probabilities and provides superior classifications when compared to other methods. This is critical given the prevalence of categorical time series in applications (e.g., the motivating example of sleep, signal failure detection, part of speech tagging, etc.). Ongoing research seeks to make further improvements to our sequential classification method through general improvements to machine learning algorithms and by accommodating even more general time dependence structures.

References

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

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

McShane, B. B., Jensen, S. T., and Wyner, A. J. (2009). Statistical learning methods for modeling sleep in mice. Tech. rep., Department of Statistics, University of Pennsylvania.

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

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

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.

A Results for Low Noise Setting of Simulation II

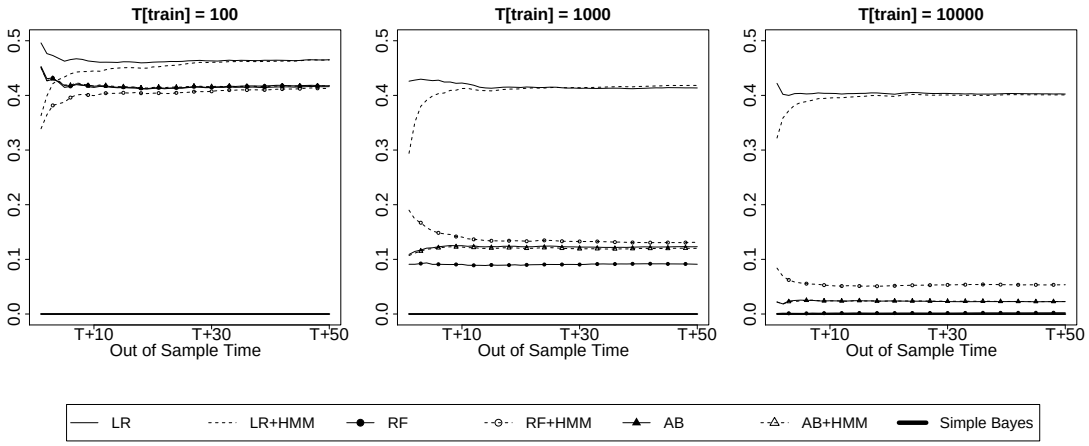

Figure 7: Classification Error Relative to Full Bayes Probabilities For Low Noise Setting.

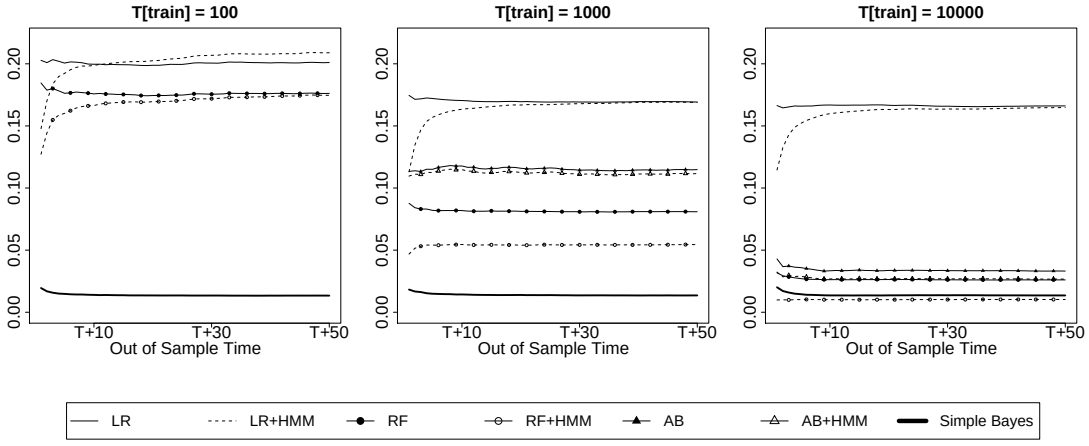

Figure 8: Squared Probability Error Relative to Full Bayes Probabilities For Low Noise Setting. AdaBoost does not appear in some plots because it performs poorly relative to the other methods.

B Results for High Noise Setting of Simulation II

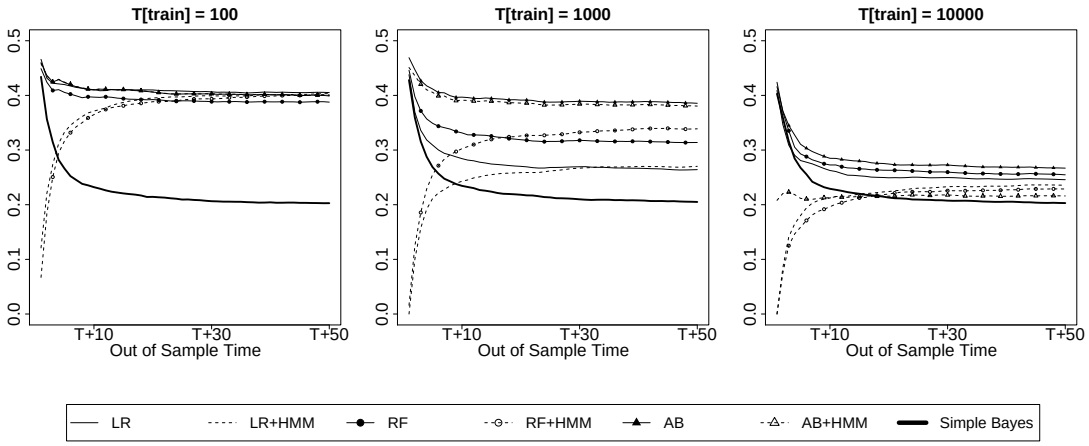

Figure 9: Classification Error Relative to Full Bayes Probabilities For High Noise Setting.

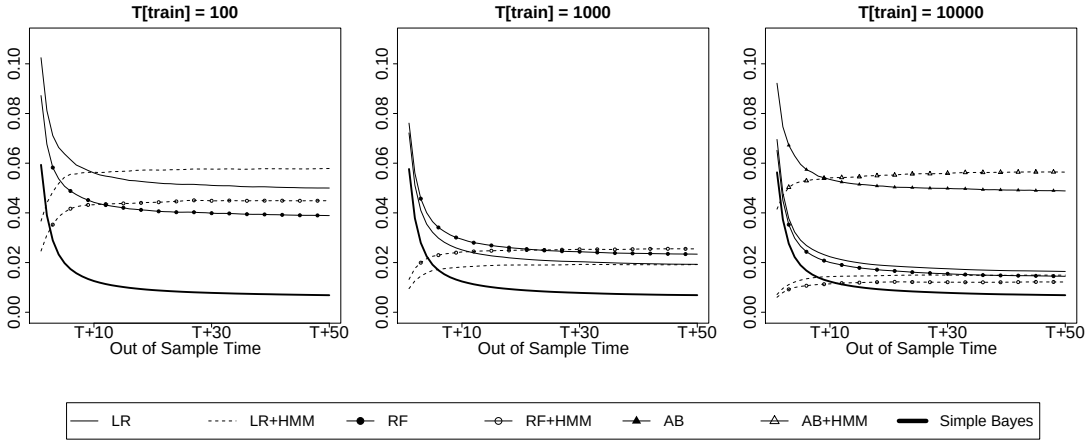

Figure 10: Squared Probability Error Relative to Full Bayes Probabilities For Low Noise Setting. AdaBoost does not appear in some plots because it performs poorly relative to the other methods.