University of California, Merced

Capstone Project

Monte Carlo Approaches to Hidden Markov Model State Estimation

Derek Sollberger

A technical report submitted
in partial fulfillment of the requirements for the degree of

Master of Science in Applied Mathematics

2011

University of California, Merced
Graduate Division

This is to certify that I have examined a copy of a technical report by

Derek Sollberger

and found it satisfactory in all respects, and that any and all revisions required by the examining committee have been made.

Research Advisor: _____
Harish Bhat

Reading Committee: _____
Roummel Marcia

Graduate Coordinator: _____
Boaz Ilan

_____
May 4, 2011

ABSTRACT OF THE CAPSTONE PROJECT

Monte Carlo Approaches to HMM State Estimation

by

Derek Sollberger

Master of Science in Applied Mathematics
2011

University of California, Merced

Prof. Harish Bhat, Advisor

Abstract

In this paper, we develop a Monte Carlo approach for hidden Markov model (HMM) order estimation–finding the underlying number of states in a hidden Markov model. We compare predictions and true observations using classification rates, correlations, and ROC curves as statistical estimators. Tests are run on both artificial data in a controlled experiment and on real-world data sets.

Contents

1 Introduction **6**
1.1 Markov Models . . . 6
1.2 Hidden Markov Models . . . 7
1.3 Bayes' Rule . . . 9

2 HMM Derivations **10**
2.1 Forward Algorithm . . . 10
2.2 Backward Algorithm . . . 13
2.3 Viterbi Algorithm . . . 16
2.4 Baum-Welch Algorithm . . . 18
2.5 Notes on Convergence . . . 22

3 HMM Experiments **23**
3.1 Tutorial for the PMTK HMM Code . . . 23
3.2 Monte Carlo HMM State Estimation . . . 24
3.3 Other Statistical Estimators . . . 25
3.4 HMMs on Real-World Data Sets . . . 27
3.5 Bagging . . . 29

4 Conclusion **31**

5 Appendices **32**
5.1 Descriptions of Data Sets . . . 32
5.2 Glossary . . . 33
5.3 MATLAB Code . . . 33
5.4 Acknowledgments . . . 40

List of Figures

1 Example Hidden Markov Model: The dashed boxes represent hidden states. The number in parentheses are transition probabilities for returning to the same state . . . 8
2 Forward Algorithm Schematic . . . 11
3 Backward Algorithm Schematic . . . 15
4 Viterbi Algorithm Schematic . . . 18
5 EM Algorithm Schematic . . . 20
6 Baum Welch Algorithm Schematic . . . 21
7 Classification Rate over Observation Sequence . . . 24
8 Log of pmf . . . 25
9 Box-and-Whisker Plot for Bootstrap Aggregation (Bagging) Results with Sample Size $\hat{T} = 123$ Observations and $B = 5$ . . . 29
10 Box-and-Whisker Plot for Bootstrap Aggregation (Bagging) Results with Sample Size $\hat{T} = 123$ Observations and $B = 50$ . . . 30
11 Box-and-Whisker Plot for Bootstrap Aggregation (Bagging) Results with Sample Size $\hat{T} = 100$ Energy Prices and $B = 5$ . . . 30
12 Box-and-Whisker Plot for Bootstrap Aggregation (Bagging) Results with Sample Size $\hat{T} = 100$ Energy Prices and $B = 50$ . . . 31

List of Tables

1 Example: Initial Distribution 7
2 Example: Transition Probabilities 8
3 Example: Emission Probabilities 8
4 Baum-Welch Reestimation Formulas 21
5 Trials on Experimental Data (with $U = 3$ hidden states) 26
6 Trials on Experimental Data (with $U = 5$ hidden states) 26
7 Trials on Experimental Data (with $U = 7$ hidden states) 27
8 Trials on Attendance and Corn Data. The true number of states for the baseball data is $V = 4$. For the corn data, we used $V = 9$. 27
9 Trials on Energy and Rainfall Data. The true number of states for the energy data is $V = 4$. For the rainfall data, we used $V = 5$. 28
10 Trials on Unemployment Data. This data was discretized into $V = 7$ symbols. 28
11 Comparison of Classification Rates Between Markov and Hidden Markov models . . 29

1 Introduction

A hidden Markov model (HMM) assumes that we can model a list of observations as the result of two processes. First, there is a Markov chain describing the state of the system at $t = 1, 2, ..., T$. The number of possible states is unknown, hence "hidden". Second, the observation at time t is governed by the corresponding state at time t. In this paper, we will model several data sets with HMMs. Each model needs an educated guess at the number of hidden states. With a model in place, we hope that it represents the observation sequence well enough to make predictions for future behavior. The order estimation problem for hidden Markov models is the search for the best number of hidden states, as a part of an HMM with the best prediction rate.

We hypothesize that we can use a brute force, hands-on approach—a Monte Carlo approach—to order estimation. We hope that this approach will be more accessible to hidden Markov model practitioners since little experience in statistics is necessary, while we still try to provide assurance about the correct number of hidden states. Other, more theoretical, methods for this type of model selection have been proposed to estimate the number of states for a hidden Markov model. Statistical inference for model selection includes the Bayesian information criterion (BIC) outlined by Schwarz [14]. We will see in later in Table 6 that we can have similar results for different amounts of hidden states. To penalize possible overfitting from using too many states, Leroux and Ryden employ a penalized maximum likelihood method [6][13]. Finesso put a theoretical, upper bound on Markov chain and HMM state estimation in his thesis[1]. More recent developments in order estimation include the minimum-distance method experiments by MacKay [7], while a penalized minimum density power divergence estimator (MDPDE) used by Lee [5]. Poskitt and Zhang utilize quasi-likelihoods rather than exact likelihoods when modeling with finite mixtures [11]. Shinozaki and Ostendorf expand on HMM use with aggregated EM training for Gaussian mixture models (GMM) [15].

This paper is meant to be an introduction to hidden Markov models, with both mathematical and computational expositions. In the rest of this section, the notions of Markov chains and the more complex, HMMs are explained. Plus, there is a quick review of Bayes' Rule since it appears frequently in the development of the HMM theory. Section 2 contains step-by-step derivations of the common algorithms that arise in the use of hidden Markov models: the Viterbi Algorithm and the Baum-Welch Algorithm. Section 3 is a journal of the design of our order estimation experiment. There, we explain how we made our Monte Carlo approach and the statistical estimators—classification rates, correlations, AUC for ROCs—that are used to analyze the results. Section 4 merges the Monte Carlo idea with other, contemporary work in HMMs, including machine learning theory.

Unless otherwise noted, the derivations put forth in this paper will follow the expositions of Fraser [4]. The computations utilize MATLAB and the Probabilistic Modeling Toolkit[1] (PMTK) that was compiled by Kevin Murphy, et al.

1.1 Markov Models

We will be focusing on discrete-time, stochastic models with the Markov property. In this paper, we will use x to denote the current state and x_{+1} to denote the next state. The probability for an upcoming state x_{+1} is conditional on the present state x of the system. Furthermore, that next state x_{+1} is conditionally independent from the past states $x(-t+1), x(-t+2), ..., x(t-2), x_{+1}$.

A *Markov chain* is a sequence of random variables, and we will write one Markov chain as a vector with T elements: x_1^T. A *stochastic matrix* describes the transition in a Markov chain x_1^T

[1] `http://code.google.com/p/pmtk3/`

ground offensive	aerial offensive	defensive
0.33	0.33	0.34

Table 1: Example: Initial Distribution

over a finite state space. Let U denote the number of states. If the probability of moving from state i to state j in one time step is $a_{ij} = P(j|i)$, the stochastic matrix A is given with a_{ij} in the i^{th} row and j^{th} column:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1U} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2U} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{iU} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{U1} & a_{U2} & \cdots & a_{Uj} & \cdots & a_{UU} \end{pmatrix}$$

Note that by the nature of how probabilities add up to 100 percent, each column of A adds up to one. In order to find the *transition matrix* of a *random walk* after k steps, we simply compute the matrix A^k.

1.2 Hidden Markov Models

Now, consider a discrete set of observations y_1^T relating to the Markov chain x_1^T. We will assume that there are U different states and V different observation symbols. For each state x and each possible observation y, there is an *emission* probability $P(y|x)$. To start, an HMM needs an initial distribution π of states; that is, what is the probability of starting the Markov chain in state $y(1)$? The initial distribution, along with the transition and emission probabilities form the *model parameters* Ψ:

$$\text{model parameters} = \left\{ \begin{array}{c} \text{initial distribution} \\ \text{transition probabilities} \\ \text{emission probabilities} \end{array} \right\}$$

$$\Psi = \{P(x), P(x_{+1}|x), P(y|x) \big| x \in U, y \in V\} \tag{1}$$

Sometimes, it is easier to understand HMMs through an example. Consider the video game Street Fighter II. The protagonist Ryu is a martial arts fighter whose task is to defeat an opponent. For simplicity, assume that Ryu has three maneuvers at his disposal: a rising punch, a projectile, and a spinning kick. The original, Japanese names for these maneuvers are "shoryuken", "hadoken", and "tatsumaki senpuukyaku" respectively. The English, colloquial names are the "dragon punch", "fire ball", and "hurricane kick" respectively. A veteran player of this video game will perform these moves in response to the state of the opponent–whether the opponent is doing a ground offense, an aerial offense, or being defensive–with some probabilities.

For the opponent, let us say that he will start a situation with the initial probabilities in Table 1 Next, the opponent might keep or change his state with the transition probabilities in Table 2. Finally, the protagonist Ryu can respond to his opponent's state with the emission probabilities in Table 3

That is, if we could only observe Ryu's actions (for instance, if we could only hear those maneuvers for Ryu, but not hear or see anything else in the match), we could try to analyze the match between Ryu and his opponent with a hidden Markov model. The model parameters in Figures 1, 2, and 3 are collected in Figure 1.

A	ground	aerial	defensive
ground	0.40	0.80	0.25
aerial	0.20	0.10	0.25
defensive	0.40	0.10	0.50

Table 2: Example: Transition Probabilities

E	ground	aerial	defensive
rising punch	0.15	0.80	0.10
fireball	0.25	0.10	0.70
spinning kick	0.60	0.10	0.20

Table 3: Example: Emission Probabilities

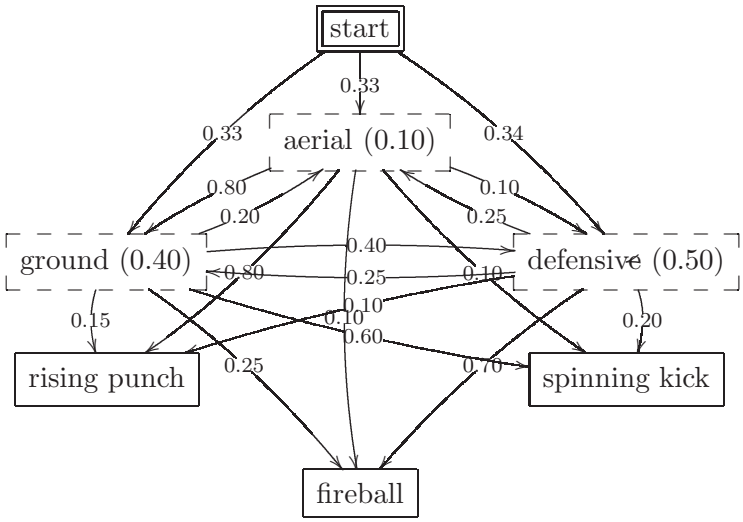

Figure 1: Example Hidden Markov Model: The dashed boxes represent hidden states. The number in parentheses are transition probabilities for returning to the same state

From here, we might take our data of Ryu's fighting maneuvers for different tasks. For example, given our data y_1^T, we might want to know how likely it was to observe that particular data (Forward Algorithm). From the Ryu data, we could also try to discern the opponent's behavior. Assuming that each one of Ryu's actions depends on his opponent's state, we can compute a corresponding, highly probable list of states for the opponent (Viterbi Algorithm). Those results could be semantically generalized to summarize if the opponent is offensive or defensive in nature. The probabilities shown in Figures 1, 2, and 3 are just educated guesses, so we might use the list of data to refine those probabilities (Baum-Welch Algorithm). The video game itself is more complex, so we might use about $U = 75$ hidden states for the opponent and about $V = 75$ maneuvers for Ryu, but the framework is still the same to use an HMM.

1.3 Bayes' Rule

The foray into conditional probability generalizes into *Bayes' Rule*. For starters, the conditional probability of event B given that A has already taken place is

$$P(A|B) = \frac{P(B,A)}{P(B)} \tag{BR1}$$

Rearranging the terms of BR1, we get another form of Bayes' Rule that may be easier to interpret. The Bayes' Rule for odds is intuitively set up as

$$\text{posterior odds} = \text{likelihoods} \times \text{prior odds}$$

and is mathematically represented as

$$P(A,B) = P(A|B)P(B) \tag{BR2}$$

That is, we can determine the posterior probability that A and B will happen together if we know the prior probability of having event B and the likelihood of including event A. [10]

We will want to consider conditional probabilities over the discrete set of states. If the states are listed as $x_1, x_2, ..., x_U$, then the *total probability* of event B can be written as

$$P(B) = \sum_{i=1}^{U} P(B, x_i) \tag{2}$$

From there, we can write a corollary of BR1 as

$$P(A|B) = \frac{P(B,A)}{P(B)} = \frac{P(BA)}{\sum_{i=1}^{U} P(B, x_i)}$$

Finally, if we want to know the posterior odds of witnessing events A and B given event C, we can use the following corollary of BR2:

$$P(A,B|C) = P(A|B,C)P(B|C)$$

These results will help us form the mathematical derivation of the hidden Markov model. [8]

2 HMM Derivations

Lawrence Rabiner is one the primary researchers in the use of HMMs for speech recognition. In his seminal paper [12], he states that we need to understand the following three problems so that we can apply HMMs to real-world applications:

1. If we have an observation sequence $\{y_1, y_2, ..., y_T\}$ and a model $\Psi = \{A, E, \pi\}$, then what is the probability that that model would have produced that sequence of events?

2. If we have an observation sequence $\{y_1, y_2, ..., y_T\}$ and a model $\Psi = \{A, E, \pi\}$, then what was the most probable sequence of states $\{x_1, x_2, ..., x_T\}$ that would have produced that observation sequence?

3. How can we refine the parameters in model Ψ?

These problems are known respectively as the Forward Algorithm, the Viterbi Algorithm, and the Baum-Welch Algorithm.

2.1 Forward Algorithm

Once we have an HMM in place with model parameters Ψ, one of the first tasks that appears is answering, "Given a list of observations, how likely was it for those events to take place?" In math terms, we want to compute $P(y_1^T|\Psi)$, where y_1^T is the whole list of observations. Ideally, we would want to go through each, possible sequence of states x_1^T, see what the chances were for that list of observations, and add those chances for the total probability:

$$P(y_1^T|\Psi) = \sum_{x_1^T} P(x_1^T, y_1^T|\Psi) \tag{3}$$

However, we can see in equation (3) that the sheer amount of calculations would scale exponentially with T, the length of the observations list. In practice, we will be using hundreds—possibly thousands—of data points, so such an approach would be quite unwieldy. Until the section about the Baum-Welch Algorithm and EM, we can assume that the model parameters Ψ are fixed and drop Ψ from the notation.

Fortunately, there is a paradigm shift that makes use of HMMs possible:

$$P(y_1^T) = P(y(1)) \prod_{t=2}^{T} P(y(t)|y_1^{t-1}) \tag{4}$$

This equation is telling us that we are going to ask "What is the probability of having the first item on our observation list?" Next, our goal will be to find the conditional probability of having this particular observation $y(t)$, given all of the items y_1^{t-1} that came before it. That forward movement sounds reasonable enough, but we need to delve back into the layer of hidden states. That is, while equation (4) is not a straight forward calculation, we might expect that the total number of calculations to be some multiple of T, so we can conjecture that the Forward Algorithm is $O(T)$. This approach is much more feasible than the exponential growth found in equation (3).

We will encounter the use of logarithms many times when working with HMMs. Recalling how a set of probabilities must add up to 100 percent, individual probabilities of a rather large list—for instance, the transition probability between states—might be very small. To avoid underflow,

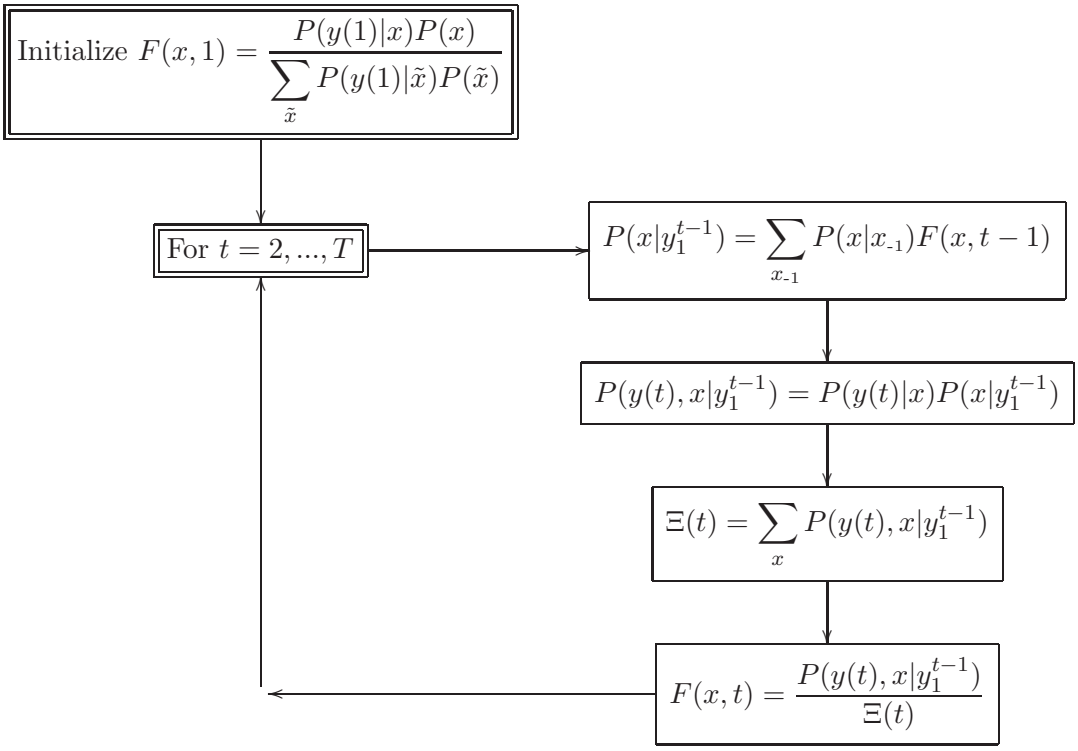

Figure 2: Forward Algorithm Schematic

we take the log of both sides of (4) and compute

$$\log P(y_1^T) = \log P(y(1)) + \sum_{t=2}^{T} \log P(y(t)|y_1^{t-1}) \tag{5}$$

Since those conditional probabilities are important to us, let us give them a name:

$$\Xi(t) \equiv P(y(t)|y_1^{t-1}) \tag{6}$$

Considering how we will have to alternate our focus between the layer of hidden states and the layer of observations, let us define[2]

$$F(x,t) \equiv P(x|y_1^t) \tag{7}$$

the probability of being in state x given the list of observations up to the current time t.

As outlined in Figure 2, we initialize the Forward Algorithm by computing $F(x,1)$, the

[2]A lot of literature on HMMs use α for this conditional probability.

distribution[3] of states at time $t = 1$. With liberal application of Bayes' Rule, we get $\forall \tilde{x} \in X$

$$\begin{aligned}
F(x,1) &= P(x|y(1)) \\
&= \frac{P(y(1),x)}{P(y(1))} && \text{by BR1} \\
&= \frac{P(y(1)|x)P(x)}{P(y(1))} && \text{by BR2 in the numerator} \\
&= \frac{P(y(1)|x)P(x)}{\sum_{\tilde{x}} P(y(1),\tilde{x})} && \text{finding total probability in the denominator} \\
&= \frac{P(y(1)|x)P(x)}{\sum_{\tilde{x}} P(y(1)|\tilde{x})P(\tilde{x})} && \text{by BR2 in the denominator}
\end{aligned}$$

Note that $P(y(1)|\tilde{x})$ is a model parameter that is available to us by assumption, and $P(\tilde{x})$ is an element of the prior probability matrix.

Later, we will need to calculate all the values of F for all x and t. Now, we need to find all the values of Ξ for all t to use equation (5). For times $t = 2, 3, ..., T$, we perform the following steps. First, we find the distribution of states based on the sequence of observations before the current time. That is, $\forall x_{\text{-}1} \in X$ in the previous time step, we have for each state x:

$$\begin{aligned}
P(x|y_1^{t-1}) &= \sum_{x_{\text{-}1}} P(x, x_{\text{-}1}|y_1^{t-1}) && \text{total probability over all states} \\
&= \sum_{x_{\text{-}1}} P(x|x_{\text{-}1}, y_1^{t-1})P(x_{\text{-}1}|y_1^{t-1}) && \text{BR2 on the items left of the bar} \\
&= \sum_{x_{\text{-}1}} P(x|x_{\text{-}1}, y_1^{t-1})F(x,t-1) && \text{by definition of } F \\
&= \sum_{x_{\text{-}1}} P(x|x_{\text{-}1})F(x,t-1)
\end{aligned}$$

where the last simplification was done by noting how state x, is conditionally independent from states $x(1), x(2), ..., x(t-2)$. We have derived a formula for $P(x|y_1^{t-1})$ as a combination of previously computed $F(x, t-1)$ and elements $P(x|x_{\text{-}1})$ of the state transition matrix. Next, we fill in the emission between the state space and the observation space. For this time step t and $\forall x \in X$, we obtain:

$$\begin{aligned}
P(y(t), x|y_1^{t-1}) &= P(y(t)|x, y_1^{t-1})P(x|y_1^{t-1}) && \text{by BR2} \\
&= P(y(t)|x)P(x|y_1^{t-1})
\end{aligned}$$

[3]While $F(x,t)$ refers to one particular state x, we will need to compute that quantity for every state x, so we are considering a whole distribution of states even though here F is a single-valued function.

where the simplification in the first factor $P(y(t)|x)$ was done by noting how $y(t)$ is conditionally independent from states $x(1), x(2), ..., x(t-2)$. The second factor $P(x|y_1^{t-1})$ was computed above. Now that we filled in those emission probabilities at time t, we can easily find the value of $\Xi(t)$:

$$\begin{aligned}
\Xi(t) &= P(y(t)|y_1^{t-1}) && \text{definition of } \Xi \\
&= \sum_x P(y(t), x|y_1^{t-1}) && \text{total probability over all states } x
\end{aligned}$$

Here it appears that we have the value for Ξ that can be used in equation (5). However, to perform the next iteration of the Forward Algorithm, and for later calculations in the Baum-Welch Algorithm, we need to calculate $F(x,t)$ for the current time step:

$$\begin{aligned}
F(x,t) &= P(x|y_1^t) && \text{definition of } F \\
&= P(x|y(t), y_1^{t-1}) && \text{partitioning the list of observations} \\
&= \frac{P(y(t), x|y_1^{t-1})}{P(y(t)|y_1^{t-1})} && \text{by BR1} \\
&= \frac{P(y(t), x|y_1^{t-1})}{\Xi(t)} && \text{definition of } \Xi
\end{aligned}$$

In other words, the probability of being in state x given the observation sequence up to this point $P(x|y_1^t)$ is the emission probability $P(y(t), x|y_1^{t-1})$ normalized by $\Xi(t)$.

Note that we find values for F for every time $t = 1, 2, ..., T$ and for each state x. The calculation for each state x is dependent on the calculation for each $x_{\text{-}1}$ in the previous time step, so we say that the Forward Algorithm is of order $O(U^2T)$. This is far better than the exponential growth in equation (3).

2.2 Backward Algorithm

Overall, we have been trying to take a list of observations, and finding the probabilities of being in the underlying, hidden states. Above, we used the Forward Algorithm to find $P(x|y_1^t)$ for *partial* sequences of observations y_1^t. We might need stronger calculations for these probabilities, say, for the Baum-Welch Algorithm. If we wish to use the *entire* observation sequence y_1^T, one surprising approach is to assume that we can simply multiply the probabilities $F(x,t)$ by some other probabilities[4] $B(x,t)$ to get the desired product:

$$P(x|y_1^T) = F(x,t)B(x,t) \tag{8}$$

[4]Most HMM literature uses β here.

The naive approach is to simply solve for B:

$$\begin{aligned} B(x,t) &= \frac{P(x|y_1^T)}{F(x,t)} \\ &= \frac{P(x|y_1^T)}{P(x|y_1^t)} \end{aligned}$$

Here we have solved for $B(x,t)$, but in terms of the $P(x|y_1^T)$ that we wish to find (which is circular logic). Keeping in mind that the Forward Algorithm found F with dependence up to time t, the goal now is to fill in the dependence on the observations between times $t+1$ and final time T. In the following derivation, we are going to assume that observations are conditionally independent from each other.

$$\begin{aligned} B(x,t) &= P(x|y_1^T)\frac{1}{P(x|y_1^t)} \\ &= \frac{P(y_1^T,x)}{P(y_1^T)} \cdot \frac{P(y_1^t)}{P(y_1^t,x)} && \text{BR1 on both } P(x|y_1^T) \text{ and } P(x|y_1^t) \\ &= \frac{P(y_1^T|x)P(x)}{P(y_1^T)} \cdot \frac{P(y_1^t)}{P(y_1^t|x)P(x)} && \text{BR2 on both } P(y_1^T,x) \text{ and } P(y_1^t,x) \\ &= \frac{P(y_1^T|x)}{P(y_1^t|x)} \cdot \frac{P(y_1^t)}{P(y_1^T)} && \text{cancellation of } P(x) \\ &= \frac{P(y_1^t|x)P(y_{t+1}^T|x)}{P(y_1^t|x)} \cdot \frac{P(y_1^t)}{P(y_1^T)} && \text{assumed independence: } P(y_1^T) = P(y_1^t)P(y_{t+1}^T) \\ &= P(y_{t+1}^T|x) \cdot \frac{P(y_1^t)}{P(y_1^T)} && \text{cancellation of } P(y_1^t|x) \\ &= P(y_{t+1}^T|x) \cdot \frac{1}{P(y_{t+1}^T|y_1^t)} && \text{BR1 on } P(y_{t+1}^T|y_1^t) \end{aligned}$$

Now we see that

$$B(x,t) = \frac{P(y_{t+1}^T|x)}{P(y_{t+1}^T|y_1^t)} \tag{9}$$

is intent on only finding probabilities between times $t+1$ and T.

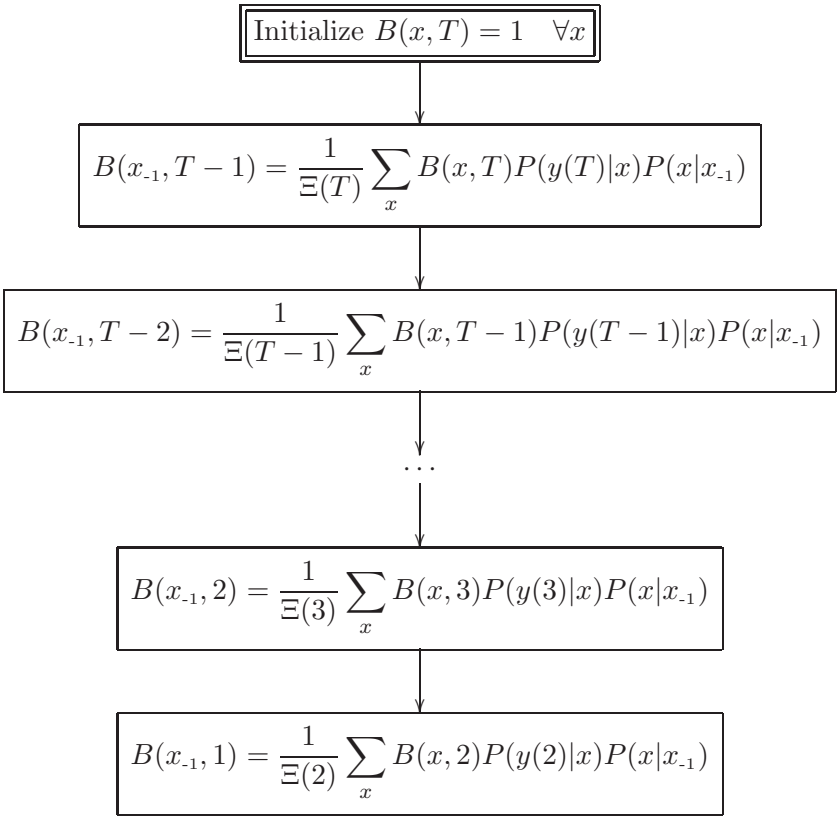

Figure 3: Backward Algorithm Schematic

Now we are headed into Figure 3, the heart of the Backward Algorithm. Starting with the end time T, equation (8) becomes

$$P(x|y_1^T) = F(x,T)B(x,T) = P(x|y_1^T)B(x,T)$$

which implies that we initialize $B(x,T)=1 \quad \forall x$. Now we fill in other values of B by going backward from time T, so for $t = T-1, T-2, ..., 1$, we form $B(x_{-1}, t-1)$ with equation (9) and iterate with

$$
\begin{aligned}
B(x_{-1}, t-1) &= \frac{P(y_t^T|x_{-1})}{P(y_t^T|y_1^{t-1})} \\
&= \frac{\sum_x P(y_t^T, x|x_{-1})}{P(y_t^T|y_1^{t-1})} && \text{total probability over all states } x \\
&= \sum_x \frac{P(y_{t+1}^T, y(t), x|x_{-1})}{P(y_t^T|y_1^{t-1})} && \text{partition of observation sequence} \\
&= \sum_x \frac{P(y_{t+1}^T|y(t), x, x_{-1})P(y(t), x|x_{-1})}{P(y_t^T|y_1^{t-1})} && \text{BR2 on } P(y_{t+1}^T, y(t), x|x_{-1}) \\
&= \sum_x \frac{P(y_{t+1}^T|y(t), x, x_{-1})P(y(t)|x, x_{-1})P(x|x_{-1})}{P(y_t^T|y_1^{t-1})} && \text{BR2 on } P(y(t), x|x_{-1}) \\
&= \sum_x \frac{P(y_{t+1}^T|x)P(y(t)|x)P(x|x_{-1})}{P(y_t^T|y_1^{t-1})} && \text{model assumptions} \\
&= \sum_x \frac{B(x,t)P(y_{t+1}^T|y_1^t)P(y(t)|x)P(x|x_{-1})}{P(y_t^T|y_1^{t-1})} && \text{by equation (9)} \\
&= \sum_x \frac{B(x,t)P(y_{t+1}^T|y_1^t)P(y(t)|x)P(x|x_{-1})}{P(y(t)|y_1^{t-1})P(y_{t+1}^T|y_1^{t-1})} && \text{independence: } P(y_t^T) = P(y(t))P(y_{t+1}^T) \\
&= \frac{1}{P(y(t)|y_1^{t-1})} \sum_x B(x,t)P(y(t)|x)P(x|x_{-1}) && \text{cancellation of } P(y_{t+1}^T|y_1^t) \\
&= \frac{1}{\Xi(t)} \sum_x B(x,t)P(y(t)|x)P(x|x_{-1}) && \text{definition of } \Xi
\end{aligned}
$$

Here, the iteration formula for $B(x_{-1}, t-1)$ is in terms of previously calculated values of B, Ξ from the Forward Algorithm, and model parameters. Note that we find values for B for every time $t = T, T-1, ..., 1$ and for each state x. The calculation for each state x_{-1} is dependent on the calculation for each x in the higher time step, so we say that the Backward Algorithm is of order $O(U^2T)$.

2.3 Viterbi Algorithm

The second of the main HMM tasks set out by Ferguson was, having a list of observations, to find the most probably sequence of states that would have yielded that observation sequence. Moore gives a good exposition about why a brute force approach would not be feasible [9]. In our notation, that might entail finding

$$P(y_1^T) = \sum_{x_1^T} P(y_1^T, x_1^T) = \sum_{x_1^T} P(y_1^T|x_1^T)P(x_1^T) \tag{10}$$

where once again we have used BR2. For each of those factors, the number of combinations of state sequences x_1^T is U^T. As seen before in the Forward Algorithm (3), we should avoid approaches that

grow exponentially with T, the length of the observation list.

Also similar to the Forward Algorithm, we try to proceed step by step through the observation list. Observe that with a couple of applications of BR2, we can rewrite that joint probability at time $t+1$ as

$$\begin{aligned} P(y_1^{t+1}, x_1^{t+1}) &= P(y(t+1), y_1^t, x(t+1), x_1^t) \\ &= P(y(t+1)|y_1^t, x(t+1), x_1^t) P(y_1^t, x(t+1), x_1^t) \\ &= P(y(t+1)|y_1^t, x(t+1), x_1^t) P(x(t+1)|y_1^t, x_1^t) P(y_1^t, x_1^t) \\ &= P(y(t+1)|x(t+1)) P(x(t+1)|x(t)) P(y_1^t, x_1^t) \end{aligned}$$

Here we have a recursive formula for $P(y_1^{t+1}, x_1^{t+1})$, where the other factors simplified to be model parameters. Furthermore, since some probabilities might be small, we can take logarithms to find

$$\log P(y_1^{t+1}, x_1^{t+1}) = \log P(y(t+1)|x(t+1)) + \log P(x(t+1)|x(t)) + \log P(y_1^t, x_1^t) \tag{11}$$

For the last term of equation (11), let

$$\chi(x,t) \equiv \max_{x_1^t : x(t)=x} \log P(y_1^t, x_1^t) \tag{12}$$

be the utility of the best sequence ending at a particular state x at time t. For example, we initialize the Viterbi Algorithm by finding

$$\chi(x,1) = \max_x \log P(y(1), x) \tag{13}$$

by searching the emission matrix for the maximal element in the column corresponding to $y(1)$. Focusing on the two state sequence elements in equation (11), let

$$\Lambda(x, x_{+1}, t) \equiv \log P(y(t+1)|x(t+1)) + \log P(x_{+1}|x) + \chi(x,t) \tag{14}$$

The first part of the Viterbi Algorithm consists of iterating through $t = 1, 2, ..., T-1$ to calculate

$$\chi(x_{+1}, t+1) \equiv \max_x \Lambda(x, x_{+1}, t) \tag{15}$$

Finally, the best state sequence $\hat{x}_1^T$ is found by starting at the end with

$$\hat{x}(T) = \arg\max_x \chi(x, T) \tag{16}$$

and iterate through $t = T-1, T-2, ..., 1$ with

$$\hat{x}(t) = \arg\max_x \Lambda(x, \hat{x}_{+1}, t+1) \tag{17}$$

That information is summarized in Figure 4. For every observation symbol, we are considering the combinations between two groups of states, so the order of the Viterbi Algorithm is also $O(U^2 T)$.

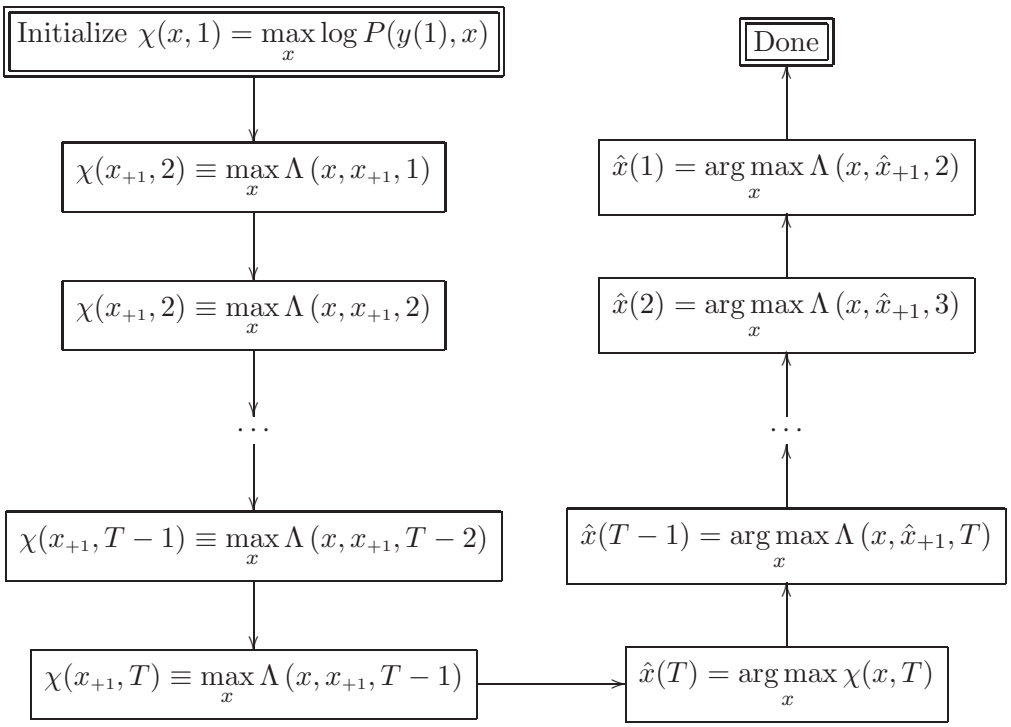

Figure 4: Viterbi Algorithm Schematic

2.4 Baum-Welch Algorithm

With the Forward and Backward Algorithms in place, we can use the Baum-Welch Algorithm to produce the model parameters Ψ. This process calculates the probabilities in π, the prior probability matrix; A, the transition matrix; and E, the emission matrix.

Similar to the Viterbi Algorithm, we will use a couple of intermediate variables to describe probabilities that appear frequently during the derivation. First, directly from equation (8), let one weight be defined by:

$$\omega(x,t) \equiv P(x|y_1^T) = F(x,t)B(x,t) \tag{18}$$

A more complicated task is to consider two levels of hidden states given the list of observations:

$$
\begin{aligned}
P(x_{+1}, x|y_1^T) &= P(x_{+1}, x|y_{t+2}^T, y_1^{t+1}) && \text{independence} \\
&= \frac{P(y_{t+2}^T, x_{+1}, x|y_1^{t+1})}{P(y_{t+2}^T|y_1^{t+1})} && \text{BR1} \\
&= \frac{P(y_{t+2}^T|x_{+1}, x, y_1^{t+1})}{P(y_{t+2}^T|y_1^{t+1})} P(x_{+1}, x|y_1^{t+1}) && \text{BR2 on the numerator} \\
&= \frac{P(y_{t+2}^T|x_{+1})}{P(y_{t+2}^T|y_1^{t+1})} P(x_{+1}, x|y_1^{t+1}) && \text{model assumption} \\
&= B(x_{+1}, t+1) P(x_{+1}, x|y_1^{t+1}) && \text{by equation (9)} \\
&= B(x_{+1}, t+1) \frac{P(y(t+1), x_{+1}, x|y_1^t)}{P(y(t+1)|y_1^t)} && \text{BR1 on } P(x_{+1}, x|y_1^{t+1}) \\
&= \frac{B(x_{+1}, t+1)}{\Xi(t+1)} P(y(t+1), x_{+1}, x|y_1^t) && \text{definition of } \Xi \\
&= \frac{B(x_{+1}, t+1)}{\Xi(t+1)} P(y(t+1)|x_{+1}, x, y_1^t) P(x_{+1}, x|y_1^t) && \text{BR1 on } P(y(t+1), x_{+1}, x|y_1^t) \\
&= \frac{B(x_{+1}, t+1)}{\Xi(t+1)} P(y(t+1)|x_{+1}, x, y_1^t) P(x_{+1}|x, y_1^t) P(x|y_1^t) && \text{BR1 on } P(x_{+1}, x|y_1^t) \\
&= \frac{B(x_{+1}, t+1)}{\Xi(t+1)} P(y(t+1)|x_{+1}) P(x_{+1}|x) P(x|y_1^t) && \text{model assumptions} \\
&= \frac{F(x,t) B(x_{+1}, t+1)}{\Xi(t+1)} P(y(t+1)|x_{+1}) P(x_{+1}|x) && \text{definition of } F
\end{aligned}
$$

With that derivation, let the next weight be:

$$W(x_{+1}, x, t) \equiv \frac{F(x,t) B(x_{+1}, t+1)}{\Xi(t+1)} P(y(t+1)|x_{+1}) P(x_{+1}|x) \tag{19}$$

an elegant amalgamation of the model parameters and what we have calculated in the Forward and Backward Algorithms.

The process to improve the model parameters Ψ is called the EM Algorithm. Historically, the name was some combination of estimation/expectation and maximization/modification, but the name "EM" survives. As outlined in Figure 5, the "E-Step" involves understanding the state space X, given the list of observations y_1^T and the current guess Ψ at the model parameters. Next, the "M-Step" optimizes an auxiliary function C over Ψ_{+1}, all possible sets of parameters. Finally, these calculations are repeated until the model parameters have converged in some sense.

The Baum-Welch Algorithm was a precursor to the "EM" umbrella that focuses on HMMs. We have already developed the tools that characterize the state space $P(X|y_1^T, \Psi)$ for discrete HMMs, namely the Forward and Backward Algorithms.

Here, we outline the M-Step. Let the auxiliary function be defined as the product of the

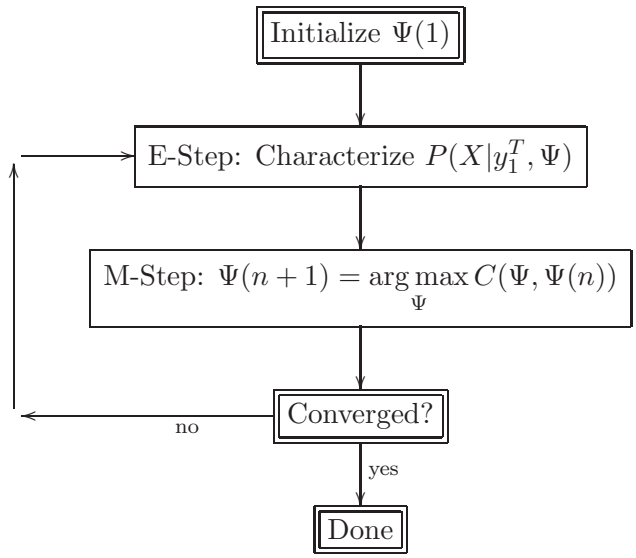

Figure 5: EM Algorithm Schematic

expectation of the state space and the log, joint probability of the states and observations

$$C(\Psi_{+1}, \Psi) = \mathbb{E}_{P(X|y_1^T, \Psi)} \cdot \log P(X, y_1^T|\Psi') \tag{20}$$

Similar to the paradigm presented in equation (4), we can defenestrate the exponential complexity and write joint probability can be expressed as

$$P(x_1^T, y_1^T|\Psi) = P(x(1)) \prod_{t=2}^{T} P(x(t)|x_{\text{-}1}) \prod_{t=1}^{T} P(y(t)|x(t))$$

That is, we consider the prior probability of being in the first state of the sequence $x(1)$, and then proceed through the transitions $P(x(t)|x_{\text{-}1})$ between states and the transmissions $P(y(t)|x(t))$ to the observations. To avoid underflow, we can take the logarithm of the above equation to get

$$\log P(x_1^T, y_1^T|\Psi) = \log P(x(1)) + \sum_{t=2}^{T} \log P(x(t)|x_{\text{-}1}) + \sum_{t=1}^{T} \log P(y(t)|x(t)) \tag{21}$$

Now we want to combine equations (20) and (21). In place of the expectation operator $\mathbb{E}$, we try to use the auxiliary function over all possible state sequences x_1^T. Thus, our goal is to optimize

$$C(\Psi', \Psi) = \sum_{x_1^T} P(x_1^T|y_1^T, \Psi) \left[\log P(x(1)|\Psi') + \sum_{t=2}^{T} \log P(x(t)|x_{\text{-}1}|\Psi') + \sum_{t=1}^{T} \log P(y(t)|x(t), \Psi') \right] \tag{22}$$

Following our goal to update the model parameters, Fraser's summary is shown in Table 4. A schematic for the Baum-Welch process is shown in Figure 6.

Item	Expression	Update Value
Prior Probability	$P(x\|\Psi(n+1))$	$\omega(x,1)$
Transition Probability	$P(x_{+1}\|x,\Psi(n+1))$	$\frac{\sum_{t=1}^{T-1} W(x_{+1},x,t)}{\sum_{\tilde{x}_{+1}\in X}\sum_{t=1}^{T-1} W(\tilde{x}_{+1},x,t)}$
Emission Probability	$P(y\|x,\Psi(n+1))$	$\frac{\sum_{t:y(t)=y} \omega(x,t)}{\sum_{t} \omega(x,t)}$

Table 4: Baum-Welch Reestimation Formulas

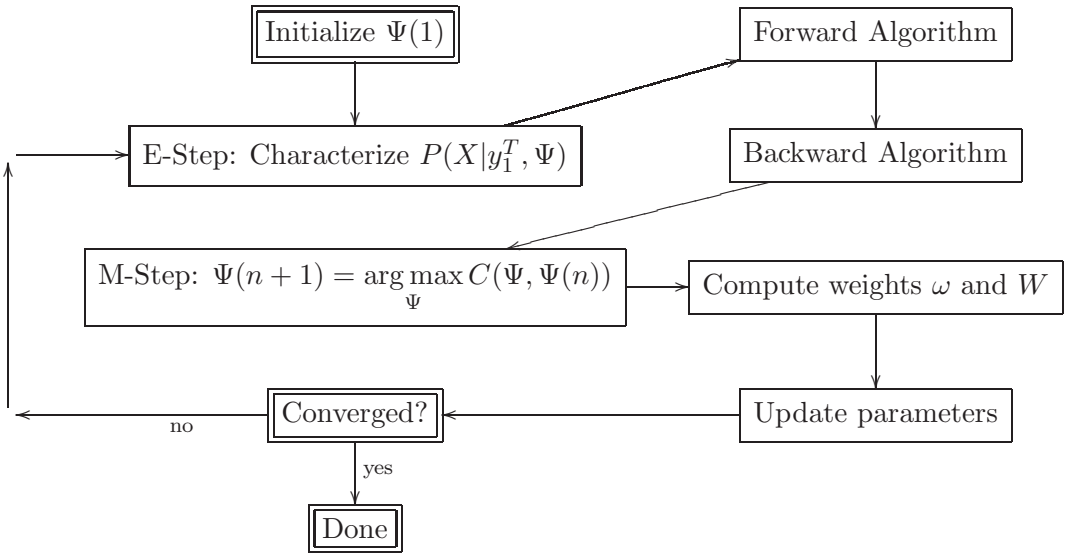

Figure 6: Baum Welch Algorithm Schematic

2.5 Notes on Convergence

Finally, now we want to claim that the Baum-Welch Algorithm converges to the best model parameters Ψ in some sense. The idea here is to note how the log likelihood of observing a sequence y_1^T given the model parameters

$$L(\Psi) \equiv \log P(y_1^T|\Psi) \tag{23}$$

is bounded above by zero, and then show that the likelihood is a nondecreasing function so that convergence of the likelihood function would follow.

Let the *cross-entropy* of two models with parameters Ψ and Ψ' be defined as

$$H(\Psi'||\Psi) \equiv -\mathbb{E}_{P(X|y_1^T,\Psi)} \cdot \log P(X|y_1^T, \Psi') \tag{24}$$

The cross-entropy is a measurement of information gain that we achieve by moving from using parameters Ψ to using parameters Ψ' [8].

We can write the Baum-Welch auxiliary function as

$$\begin{aligned}
C(\Psi', \Psi) &= \mathbb{E}_{P(X|y_1^T,\Psi)} \cdot \log P(X, y_1^T|\Psi') \\
&= \mathbb{E}_{P(X|y_1^T,\Psi)} \cdot \log \left[P(X|y_1^T, \Psi')P(y_1^T|\Psi')\right] && \text{BR2 on } P(X, y_1^T|\Psi') \\
&= \mathbb{E}_{P(X|y_1^T,\Psi)} \left[\log P(X|y_1^T, \Psi') + \log P(y_1^T|\Psi')\right] && \text{property of logarithms} \\
&= L(\Psi') - H(\Psi'||\Psi) && \text{by equations (23) and (24)}
\end{aligned}$$

That leaves us with the likelihood function

$$L(\Psi') = C(\Psi', \Psi) + H(\Psi'||\Psi) \tag{25}$$

which we wish to maximize.

Since hidden states are conditionally independent from the observations, we have $P(x|y_1^T, \Psi') = P(x|y_1^T, \Psi)$ for all states x and for model parameter sets Ψ and Ψ'. That allows us to invoke Gibbs' Inequality

$$P(x|y_1^T, \Psi') = P(x|y_1^T, \Psi) \ \forall x \quad \Rightarrow \quad H(\Psi'||\Psi) \geq H(\Psi||\Psi) \tag{26}$$

which is a consequence of Jensen's Inequality.

Recall that the M-step of the EM Algorithm is $\Psi(n+1) = \arg\max_{\Psi} C(\Psi, \Psi(n))$. Combining that formulation with $H(\Psi'||\Psi) \geq H(\Psi||\Psi)$ from Gibbs' Inequality (26), it follows that the likelihood function (25) is nondecreasing. Since $L(\Psi)$ is a nondecreasing function and bounded above, the Baum-Welch Algorithm converges.

Just like in calculus, up to here we have shown how to search for local maxima for the likelihood function L over the space of all possible parameters Ψ. One of the shortcomings of the Baum-Welch Algorithm is that we might approach different sets of model parameters, depending on how we initialize $\Psi(1)$. Fraser presents a discussion of Hessian calculations as an analogue of the second-derivative test, but those calculations show that we cannot guarantee a limit point—much less a global maximum.

3 HMM Experiments

With all of the above HMM derivations in hand, we wish to build an experiment that will allow us to make an educated guess about the number of hidden states in a discrete HMM. The hypothesis is that we can somehow use the likelihoods found in the HMM fitting calculations, and form a predictive model.

One way to approach this hypothesis is to build a hidden Markov model with a known number of states, and run the process T times to form the observation list y_1^T. Next, we pretend to not know the correct number of states, and fit various HMMs to the observation sequence. The test here is leave-one-out predictions. We cut off the observations at some value k, fit an HMM to the observations y_1^k, and try to predict observation $y(k+1)$. This is done by finding the MAP Viterbi path x_1^k, and then making a prediction $y(k+1)$ through a probability mass function (pmf) over the discrete set of possible observation symbols $y_1, y_2, ..., y_V$.

We also might want to know how the use of HMMs as predictive modeling compares to other stochastic models, such as regular Markov models. Some trials of our program were run on real-world data sets. We can then compare our classification rate from our HMMs (performed with various numbers of states) to a previous work on Markov models.

3.1 Tutorial for the PMTK HMM Code

The Probabilistic Modeling Toolkit (PMTK) contains functions that will run perform common, HMM actions such as

- computing log likelihoods
- fitting an HMM to a given list of observations (Baum-Welch Algorithm)
- ascertaining the MAP sequence of states (Viterbi Algorithm)

The MATLAB m-files for HMMs are contained in the 'hmm' directory (folder), which in turn is found in the 'Latent Variable Models' directory. Most of the functions will require the "model", which is a struct data type in the toolkit. As outlined the derivations above, this "model" struct consists of information such as the number of hidden states, the prior distribution of states π, the transition matrix A, and another struct T for the emission matrix. Both structs indicate a data type for the observation list; we will use 'discrete' data throughout this paper. As customary, more information about the functions can be found in the m-files themselves.

We can start out by quickly generating a list of T observations among a list of V symbols:

```
T = 100;
V = 5;
X = round((V-1)*rand(1,T) + 1); %produces a list with numbers 1,2,...,V
```

Next, we create a "model" with an educated guess of U hidden states by fitting model parameters to the list of observations:

```
[model, loglikHist] = hmmFit(X, U, 'discrete');
```

Again, the `model` contains the HMM and information about the data. The `loglikHist` is a monotonically increasing array of the log likelihood of the model based on the data that confirms the notion of convergence for the EM algorithm.

Individual log likelihoods can be computed for some other list of observation $\tilde{X}$, say for predictive modeling, with

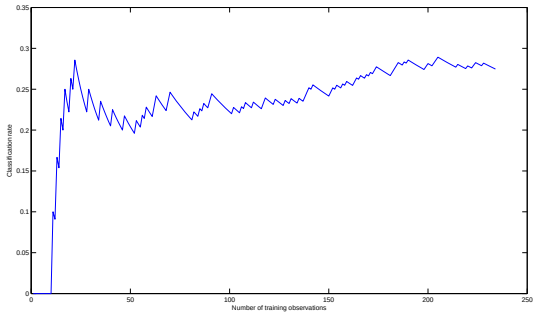

Figure 7: Classification Rate over Observation Sequence

```
logp = hmmLogprob(model, Xtilde);
```

Finally, given a list of observations, we can find the most likely, corresponding list of states–the MAP sequence of states–with the Viterbi Algorithm:

```
path = hmmMap(model, X);
```

Note that since `model` already includes the transition matrix A based on the number of hidden states, `path` will inherit that constraint.

3.2 Monte Carlo HMM State Estimation

In our trials, we made an observation list with an HMM set up for $V = 7$ symbols and $U = 5$ underlying states. We employ rejection sampling when creating transition and emission matrices so that their entries are far from uniform—that is, nontrivial matrices. An observation list y_1^T is created by running through the HMM. Once the list y_1^T was obtained, for each k in $\left\lfloor \frac{T}{2} \right\rfloor \leq k \leq T-1$, we perform leave-one-out testing:

- fit a new HMM onto the data y_1^k
- find the MAP, Viterbi path
- form a probability mass function (pmf) for the V symbols
- make a prediction via the pmf

After all those trials, we define the classification rate simply as the ratio of the number of correct predictions (compared to the true sequence) and the number of tests $(T-1)-2 = T-3$. Over many observations, the classification rate tends to converge, such as in Figure 7.

A possible fallacy with using classification rates is their binary nature. Up to now, we have been dividing predictions into "right" and "wrong" bins. This approach does not incorporate any sense of whether or not a prediction is "close" to the true value. Our predictive models might be more robust for interpretation if we can both make predictions and convey some confidence in those predictions. One place to start might be the log-likelihoods that we compute in the Baum-Welch Algorithm. We can fit HMMs with various numbers of hidden states, find the log-likelihood of the observation sequence, and form a quick distribution such as Figure 8.

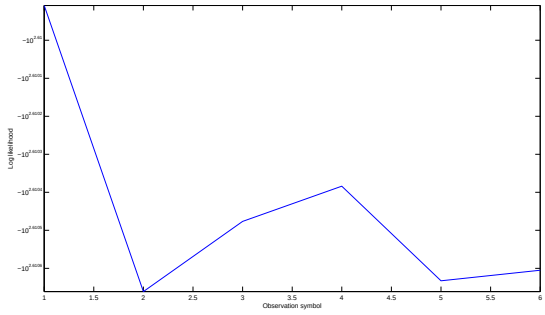

Figure 8: Log of pmf

3.3 Other Statistical Estimators

While a visual such as Figure 8 might be useful at the end of an experiment for a prediction–especially if the log-likelihoods show a skew or bias–we need to judge our program over many predictions. To condense the information between the predictions and the true observations, we need some statistical estimators.

The simplest summary might be Pearson's correlation coefficient. This number falls in the range $[-1, 1]$. If the predictions tend to be close to the observations, then the correlation should approach a value of one. The computation is as follows. Define the following:

y_m	sample mean of true observations
$\hat{y}_m$	sample mean of predictions
y_{SD}	sample standard deviation of true observations
$\hat{y}_{SD}$	sample standard deviation of predictions

then the *sample correlation* is computed as

$$r \equiv \frac{\sum_{i=1}^{T}(y_i - y_m)(\hat{y}_i - \hat{y}_m)}{(n-1)y_{SD}\hat{y}_{SD}} \tag{27}$$

A more modern approach to judging predictive models is the use of ROC curves (receiver operating characteristic). A ROC curve formed in the space $[0, 1] \times [0, 1]$, and is sketched as follows:

- Find the number of true predictions A ("true positives")
- Find the number of false predictions B ("false positives")
- Starting at the origin $(0, 0)$, for each of trials in the experiment
  - If the prediction was true, move up $1/A$ unit
  - If the prediction was false, move right $1/B$ unit

This process ends at the point $(1, 1)$. If a model is good at predictions, then the curve will generally be above the 45° line, where that diagonal line simply represents random guessing, or no confidence in the predictions. To summarize these pictures, we can compute the area under an ROC curve (AUC). The perfect classifier would have an AUC of one, while a truly random guesser would have an AUC of about 0.50. We seek values between 0.50 and 1 for our HMMs.

	Small Sample ($T = 123$)			Larger Sample ($T = 1234$)		
U	class rate	correlation	AUC	class rate	correlation	AUC
2	0.196721	0.196690	0.363333	0.165584	0.014626	0.531544
3	0.163934	0.016482	0.501923	0.163961	0.022680	0.600564
4	0.163934	0.059135	0.501923	0.154221	-0.005944	0.541500
5	0.147541	0.120900	0.601677	0.146104	0.010232	0.550748
6	0.163934	0.252252	0.550000	0.162338	0.058609	0.506673
7	0.098361	-0.170286	0.529762	0.170455	0.005472	0.530990
8	0.147541	-0.027539	0.643606	0.159091	0.012612	0.522256
16	0.180328	-0.027629	0.393939	0.139610	0.028819	0.567468
32	0.049180	-0.104958	0.293785	0.155844	-0.045158	0.550984

Table 5: Trials on Experimental Data (with $U = 3$ hidden states)

	Small Sample ($T = 123$)			Larger Sample ($T = 1234$)		
U	class rate	correlation	AUC	class rate	correlation	AUC
2	0.229508	0.105881	0.654762	0.141234	-0.084996	0.564346
3	0.147541	0.029961	0.599581	0.159091	-0.021499	0.569148
4	0.131148	-0.082064	0.699074	0.181818	0.037724	0.572065
5	0.180328	0.127917	0.648841	0.123377	-0.025676	0.612389
6	0.147541	0.014821	0.723270	0.165584	0.022616	0.576870
7	0.196721	0.074449	0.666667	0.155844	-0.010919	0.578615
8	0.196721	0.135354	0.668333	0.165584	-0.080123	0.554826
16	0.131148	-0.122543	0.680556	0.150974	-0.012586	0.611118
32	0.065574	-0.113381	0.637931	0.186688	0.042591	0.589538

Table 6: Trials on Experimental Data (with $U = 5$ hidden states)

For our experiment in order estimation, 3 observation sequences were made from hidden Markov models with 3,5, and 7 hidden states respectively. In each experiment, the Monte Carlo approach was used with $T = 1234$ observations. After each of those trials, the observation sequence was saved so that we could run the experiment again on a small sample of $T = 123$ observations. The results of those trials are shown in Tables 5, 6, and 7.

In each of those trials, we can see that the AUC becomes the favorable statistical estimator. We can seek out the maximum percentages in the classification rates, but in these experiments where we do know the true number of hidden states, those maxima do not correspond the true number U. All of the correlations are insignificant since they are values near zero. The correlation values are near zero due to two instances of randomization in the experiement—initialization for the Baum-Welch step, and prediction through the Viterbi path. Fortunately, these trials show promising results for the AUC. In Table 5, we see that the maximum AUC occurs when $U = 3$, which was the true number of hidden states for that trial. Tables 6 and 7 also show maximum values corresponding to the true number of states $U = 5$ and $U = 7$ respectively.

Unfortunately, when we ran the experiment on the smaller samples of observations, the maximal values the statistical estimators did not correspond to the true number of hidden states.

	Small Sample ($T = 123$)			Larger Sample ($T = 1234$)		
U	class rate	correlation	AUC	class rate	correlation	AUC
2	0.180328	-0.034394	0.386809	0.165584	0.006998	0.522673
3	0.147541	-0.035297	0.645702	0.150974	0.068284	0.542354
4	0.147541	-0.012143	0.561845	0.154221	0.039078	0.573704
5	0.163934	-0.076197	0.548077	0.133117	0.004966	0.532323
6	0.163934	0.139570	0.403846	0.175325	0.064977	0.539384
7	0.131148	0.016965	0.400463	0.150974	-0.027120	0.591028
8	0.098361	-0.043387	0.428571	0.123377	-0.010642	0.508318
16	0.163934	0.216625	0.551923	0.150974	0.003475	0.527723
32	0.114754	-0.113302	0.436364	0.152597	0.012155	0.584618

Table 7: Trials on Experimental Data (with $U = 7$ hidden states)

	Baseball Attendance ($T = 81$)			Corn Exports ($T = 1103$)		
U	class rate	correlation	AUC	class rate	correlation	AUC
2	0.346154	0.037718	0.592593	0.152727	0.329174	0.645741
3	0.243590	-0.090879	0.542982	0.181818	0.361274	0.673696
4	0.346154	-0.099596	0.515670	0.179091	0.302795	0.647955
5	0.282051	0.022481	0.472089	0.172727	0.269742	0.697140
6	0.243590	-0.055948	0.659649	0.167273	0.287733	0.684285
7	0.307692	-0.025488	0.596970	0.152727	0.297354	0.644522
8	0.217949	-0.086973	0.626186	0.147273	0.168738	0.580536
16	0.243590	-0.295286	0.405263	0.130909	0.020863	0.562863
32	0.179487	-0.226568	0.521978	0.137273	-0.013927	0.556075

Table 8: Trials on Attendance and Corn Data. The true number of states for the baseball data is $V = 4$. For the corn data, we used $V = 9$.

3.4 HMMs on Real-World Data Sets

Now we might want to consider how our order estimation program would work with real-world data. The goal is to see if a clear maximum arises from any of our statistical estimators. For example, we can see that in Table 8, there is virtually no correlation between the predictions and the true data for the baseball attendance data set. However, there is a virtual tie with the classification rate, as the trials to fit the data with $U = 2$ and $U = 4$ HMMs produced maximal results. Perhaps the best result comes with the AUC, for there is a clear maximum at $U = 6$ hidden states.

Also in Table 8, we see better results for the correlations. Those correlations are still low (compared to values nearing 100 percent), but what could be of note is how the correlations sharply decrease when we might be performing overfitting with $U = 16$ and $U = 32$ hidden states. The classification rate and the correlation columns both have maxima when $U = 3$, and the AUC is not far off with its maximum at $U = 5$.

In Table 9, we see similar results for the rainfall data set. That is, $U = 3$ lead to the maxima for the classification rate and correlation estimators, and $U = 5$ has the maximum AUC. The energy prices data set trial produced maximum classification rate and correlation at $U = 4$, and a maximum AUC at $U = 2$—hinting that the best HMM for that data set would have a low number of hidden states.

Finally, in Table 10, we see an example where the correlation and AUC estimators coincide

	Energy Prices ($T = 96$)			Snoqualmie Rainfall ($T = 365$)		
U	class rate	correlation	AUC	class rate	correlation	AUC
2	0.341430	0.264879	0.757002	0.425414	0.238412	0.814112
3	0.361673	0.266261	0.747150	0.425414	0.290088	0.832194
4	0.380567	0.273602	0.755419	0.414365	0.180289	0.798779
5	0.348178	0.208392	0.744354	0.381215	0.170137	0.837134
6	0.338731	0.215580	0.733936	0.392265	0.103755	0.812918
7	0.318489	0.192347	0.722541	0.389503	0.151424	0.786691
8	0.345479	0.242221	0.687122	0.356354	0.058738	0.756377
16	0.265857	0.084563	0.631183	0.339779	0.069868	0.724526
32	0.294197	0.065277	0.646316	0.281768	0.022077	0.750920

Table 9: Trials on Energy and Rainfall Data. The true number of states for the energy data is $V = 4$. For the rainfall data, we used $V = 5$.

	Unemployment Levels ($T = 561$)		
U	class rate	correlation	AUC
2	0.190476	0.353807	0.771429
3	0.251701	0.467324	0.772096
4	0.258503	0.448323	0.741866
5	0.244898	0.420855	0.756448
6	0.183673	0.060408	0.553719
7	0.183673	0.119410	0.618916
8	0.204082	-0.007852	0.504520
16	0.197279	0.133157	0.583309
32	0.210884	0.085538	0.518335

Table 10: Trials on Unemployment Data. This data was discretized into $V = 7$ symbols.

in their maxima.

At first, the HMMs seem to produce relatively low classification rates. However, we should keep in mind that the predictions are distributed over V observation symbols. Random guessing would likely produce a success rate around $\frac{1}{V}$, assuming an underlying, uniform distribution. Looking back at Table 8 with the baseball attendance and corn sales data, we can see that the classification rates were higher when $V = 4$ and lower when $V = 9$ respectively, so classification rates appear to be inversely proportional to V. A nonuniform pmf can be made from the emission matrix E, and when we compare our classification rates to the maximal value of E, we see that the prediction ability actually performs well compared to the best possible prediction rate.

These data sets were chosen so that we can compare the classification rates with the Markov model work presented in the Bhat/Kumar [2] paper. In Table 11, the HMMs only outperformed the Markov models for the energy prices data set. Reviewing the nature of hidden Markov models might explain this discrepancy. In addition to the prior probability matrix π and the transition matrix A that govern the Markov chain of states, the HMM adds on a layer of complexity with the $\mathbb{R}^{U \times V}$ emission matrix. Filling in the parameters in those matrices is a much larger task in the HMM parameter space. The set of Markov models is a subset of the HMM parameter space; we set the number of observation symbols V equal to the number of hidden states U, and use an identity matrix for the emission matrix to see this subset relationship. Some Markov models have a tractable MLE function, and a global maximum can be found. For an HMM, on the contrary,

data set	Markov models	Hidden Markov Models
baseball attendance	0.3951	0.346154
corn exports	0.1976	0.181818
energy prices	0.3542	0.380567
unemployment claims	0.5667	0.258503

Table 11: Comparison of Classification Rates Between Markov and Hidden Markov models

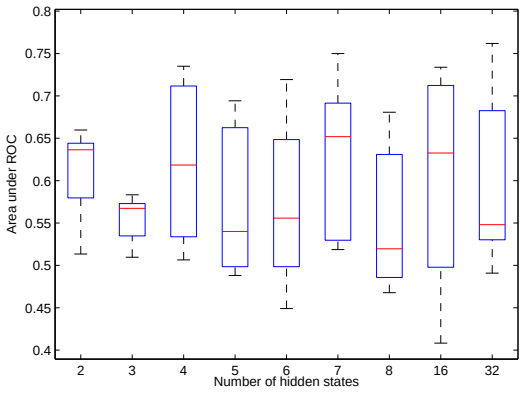

Figure 9: Box-and-Whisker Plot for Bootstrap Aggregation (Bagging) Results with Sample Size $\hat{T} = 123$ Observations and $B = 5$

we use the Baum-Welch algorithm to merely approach a local maximum, so we do not know if we reach a global maximum.

3.5 Bagging

The results of this Monte Carlo approach reveal variance. Running the experiment again can yield different classification rates, correlations, and AUC; in turn, our estimation for the number of states may be affected. Since the AUC appears to be the most promising statistical estimator in Tables 5-7, we will focus on that quantity.

Here we introduce Breiman's concept of bootstrap aggregation—"bagging" [3]. We will run the experiment B times, each time collecting the AUC for the various values of hidden states U. However, due to the processing time of our experiment, we might not want to use the entire list of observations. Instead, we perform bagging on a smaller sample size $\hat{T}$. This sampling also gives us the flexibility to run our Monte Carlo process on slightly different segments of the observation list. Different samples of size $\hat{T}$ are used, but keeping the idea that the observations are sequential, each sample uses $\hat{T}$ consecutive values.

To summarize the bootstrap aggregation results, we employ box-and-whisker plots. As seen in Figure 9, the red lines represent the median for each guess at the number of hidden states. The top and bottom of each "box" represent the 75th and 25th percentiles respectively of the AUC quantities. The top and bottom "whiskers" are approximately 2.7 standard deviations from the mean. The red plus signs beyond the whiskers are considered to be outliers. The choice to investigate the medians may come in handy since a median cannot be skewed by an outlier as much as a mean.

In Figure 9, we created an HMM with $U = 7$ hidden states, and we see that upon bagging

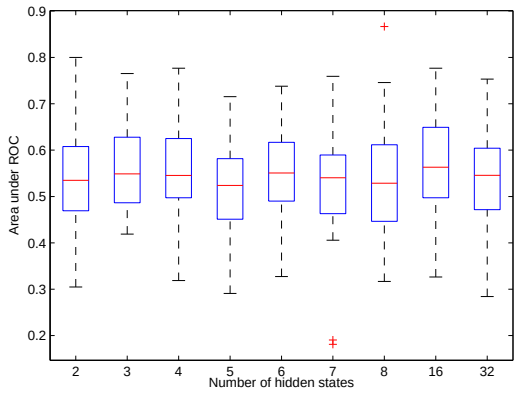

Figure 10: Box-and-Whisker Plot for Bootstrap Aggregation (Bagging) Results with Sample Size $\hat{T} = 123$ Observations and $B = 50$

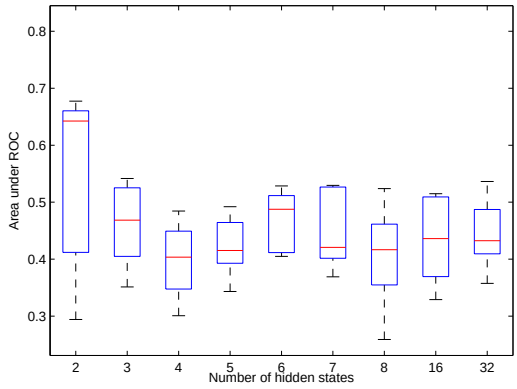

Figure 11: Box-and-Whisker Plot for Bootstrap Aggregation (Bagging) Results with Sample Size $\hat{T} = 100$ Energy Prices and $B = 5$

the procedure $B = 5$ times, the median AUC for $U = 7$ is slightly higher than the median for the testing under different numbers of hidden states. That is a promising result. However, in Figure 10, the results across various values of U become too similar to aid us in order estimation.

We see similar results with real-world data. For example, Figure 11 shows the result of our experiment on the energy prices data set. When we perform bagging $B = 5$ times, $U = 2$ clearly produces higher AUC values, so it appears that using an HMM with $U = 2$ hidden states —or at least with a small number of hidden states—would produce the best, predictive models. However, Figure 12 once again shows that bagging $B = 50$ times does not produce a clear indication for the number of hidden states.

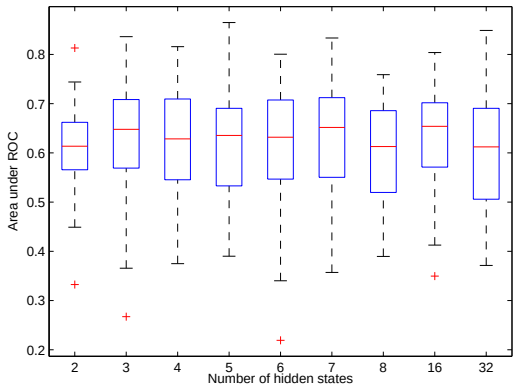

Figure 12: Box-and-Whisker Plot for Bootstrap Aggregation (Bagging) Results with Sample Size $\hat{T} = 100$ Energy Prices and $B = 50$

4 Conclusion

In many of our trials, such as such as Table 6, it was heuristically difficult to single out a clear choice for the order estimation in the hidden Markov model. Our experiment did not demonstrate any pattern concerning underfitting or overfitting the HMMs, as for the number of hidden states. Table 6 also gives a glance at overfitting/underfitting based on the number of observations; we run the program on $T = 123$ observations, then on a larger set with $T = 1234$ (and vice versa). The order estimation based on the AUC results differ. To penalize the results in those trials for various values of U hidden states, we could use a penalty function such as Lee's MDPDE [5] and have confidence in the possible number of hidden states using Finesso's upper bound [1]. Alternatively, MacKay shows how the minimum-distance method both performs order estimation and isolates local minima for the likelihood [7]. Shinozaki and Ostendorf use cross-validation and aggregated EM training for order estimation for Gaussian mixture models (GMMs). Their experiments show promising results for smaller data sets (around 50 data points) [15]. Machine learning theory might also provide assurance that we can use HMMs for smaller data sets.

Run time is the main obstacle in our Monte Carlo approach. For example, performing the leave-one-out predictions on an observation list with $T = 1234$ elements took about 9 hours on a Mac OS X with a 2.66 GHz dual core processor. In those experiments, we found results for 9 different guesses at the number of hidden states, and each use of the Baum-Welch algorithm was $O(U^2T)$. That run time motivated the bagging technique with smaller sample sizes.

After running trials of our experiment, results such as Table 6 revealed that using AUC might be able to suss out the inherent number of hidden states in an HMM. Unfortunately, running the Baum-Welch Algorithm on each observation, and for each attempt at the number of states, quickly compounds the run time for the Monte Carlo approach. On the other hand, some machine learning theory might allow researchers to perform exploratory fitting of HMMs on smaller samples of data. In this report, we have produced a parallelizable program that allows the user to fit a hidden Markov model to observed data with an educated guess for order estimation.

5 Appendices

5.1 Descriptions of Data Sets

Here are summaries of the real-world data sets used in this paper:

1. **Baseball Attendance:** home game data[5] for the 2010 season of the San Francisco Giants. The observations came from $T = 81$ regular-season games, and the list was discretized into $V = 4$ symbols.

2. **Corn Exports:** USDA[6] weekly data from 1-4-1990 to 10-21-2010 ($T = 1103$) discretized into $V = 9$ symbols. The data is in metric tons.

3. **Energy Prices:** California Independent System Operator[7] hourly data from 1-28-2009 to 1-31-2009 ($T = 96$) discretized into $V = 4$ symbols. The data is in dollars per megawatt-hour.

4. **Snoqualmie Rainfall:** daily rainfall data in 2009 ($T = 365$) for the town of Snoqualmie, Washington[8]. The data is discretized into $V = 5$ symbols.

5. **Unemployment Levels:** weekly, USDL[9] data from 11-17-2007 to 9-25-2010 ($T = 561$) discretized into $V = 7$ symbols.

[5] `http:/www.baseball-reference.com`
[6] `http://www.fas.usda.gov/export-sales/wkHistData.htm`
[7] `http://oasishis.caiso.com/`
[8] `http://www.snoqualmieweather.com/wxrainchart.php`
[9] `http://workforcesecurity.doleta.gov/unemploy/claims_arch.asp`

5.2 Glossary

A	Transition matrix with entries $P(x\|x_{\text{-}1})$
AUC	Area under a ROC curve
$B(x,t)$	Defined implicitly in equation (8): $P(x\|y_1^T) = F(x,t)B(x,t)$
BR	Bayes' Rule
$C(\Psi_{+1}, \Psi)$	Auxiliary function for the Baum-Welch Algorithm
$\chi(x,t)$	$\max_{x_1^t : x(t)=x} \log P(y_1^t, x_1^t)$
E	Emission matrix with entries $P(y(t)\|x)$
$F(x,t)$	$P(x\|y_1^t)$
GMM	Gaussian mixture model
$H(\Psi'\|\|\Psi)$	Cross entropy between models with parameters Ψ and Ψ'
HMM	Hidden Markov Model
$L(\Psi)$	Log likelihood $\log P(y_1^T\|\Psi)$
$\Lambda(x, x_{+1}, t)$	$\log P(y(t+1)\|x(t+1)) + \log P(x_{+1}\|x) + \chi(x,t)$
MAP	Maximum a posteriori ("most likely")
MLE	Maximum likelihood estimator
$\omega(x,t)$	Weight $P(x\|y_1^T) = F(x,t)B(x,t)$
PDFA	Probabilistic deterministic finite automata
pi	Prior probabilities for the hidden states
pmf	Probability mass function
Ψ	Model parameters: the set of $P(x), P(x\|x_{\text{-}1}), P(y(t)\|x)$
r	Correlation
SVM	Support vector machine
T	Length of the list of observations (and states)
U	Number of hidden states in the HMM
V	Number of possible symbols for the observations
$W(x_{+1}, x, t)$	Weight with state transition
X	State space with labels $\{1, 2, ..., U\}$
x	State
$x(t)$	Current state
$x_{\text{-}1}$	State from the previous time step
x_{+1}	State from the next time step
x_1^t	Partial list of states
x_1^T	Whole list of states
$\Xi(t)$	$P(y(t)\|y_1^{t-1})$
Y	Observation space with labels $\{1, 2, ..., V\}$
y	Observation
$y(t)$	Current observation
y_1^t	Partial list of observations
y_1^T	Whole list of observations

5.3 MATLAB Code

Besides the code in the PMTK, we have created and utilized the following codes for the experiments.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Derek Sollberger
```

```
%
% This m-file will try to indicate the number of states for a hidden Markov
% model.  That is, this program will form random data and train a hidden
% Markov model with a known number of sets.  Next, some sampling will be
% done.  Finally, through brute force attempts, HMMs with various numbers
% of states will try to fit the data.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear all
tic

% Prepare output file
time_stamp = datestr(now, 'yyyymmddTHHMMSS');
out_file = ['OE' time_stamp '.txt'];
fid_out = fopen(out_file, 'w+');

%%%% KEY PARAMETERS %%%%%
B = 50; %number of bagging iterations (use B=1 to forego bagging)
V = 7; %number of possibilities for the observations
U = 3; %number of hidden states (true value)
T = 1234; %length of observations list

%%%%% GENERATE LIST OF OBSERVATIONS %%%%%
A = mkColStocMat(U,U); %Markov process transition matrix
E = mkColStocMat(V,U); %HMM emission matrix
h = waitbar(0, 'Generating (fake) observations...');
x_prev = (1/U)*ones(U,1); %initial, (uniform) state vector
y = ones(T,1); %store list of observations

% Run HMM process
for j = 1:T
    % Transition between states
    x_curr = A*x_prev;

    % Emission to observation
    y_curr = E*x_curr;
    [temp, max_loc] = max(cumsum(y_curr) > rand);
    y(j) = max_loc;

    % Iterate
    x_prev = x_curr;
    waitbar(j/T);
end
close(h); %close wait bar

% load obs3
% T = 123;
```

```
% y = y(1:T);



% Try various amounts for the number of states
fprintf('\n');
U_vals = [2 3 4 5 6 7 8 16 32]; %the various number of states
J = length(U_vals);

classRate = zeros(J,B); %record average success rate
correlations = zeros(J,B); %record correlations
AUC = zeros(J,B); %record areas under ROC cuves

% margin12 = zeros(endTime - startTime + 1, 1);

%%%%% BAGGING %%%%%
y_True = y; %keep true list of observations
T_hat = 234; %sample size
startTime = floor(T_hat/2); endTime = T_hat-1;
pred_vals = zeros(endTime - startTime + 1, 1);


for b = 1:B %perform bagging
    % Grab a sample of the data
    picker = ceil((T_hat-B)*rand);
    y = y_True(picker:picker + T_hat);
    true_vals = y_True(startTime+picker+1 : endTime+picker+1);


    %%%%% FIT HMMs WITH VARIOUS NUMBERS OF STATES %%%%%
    fprintf('Using random, MATLAB generated data of %u observations', T_hat);
    fprintf('(with %u hidden states),\n', U);
    fprintf('and discretizing observations into %u symbols,\n', V);
    fprintf(fid_out, 'Using random, MATLAB generated data of %u observations', T_hat);
    fprintf(fid_out, '(with %u hidden states),\n', U);
    fprintf(fid_out, 'and discretizing observations into %u symbols,\n', V);


    marginW = zeros(endTime - startTime + 1, 1);

    fprintf('\n');
    fprintf(' U  class rate  correlation  AUC\n');
    for j = 1:J
        h = waitbar(0, 'Leave-one-out Testing...');

        % Discard model data from a different value of U
        clear model
        clear loglikHist
```

```
numSuccesses = 0;

% Leave-One-Out testing
for k = startTime : endTime
    k2 = k - startTime + 1; %shift

    % Train model on first k observations, 2 <= k <= T-1
    if k ~= startTime %use previous model parameters
        [model, loglikHist] = hmmFit(y(1:k)', U_vals(j), 'discrete', ...
            'pi0', pik, 'trans0', Ak, ...
            'convTol', 1e-12, 'maxIter', 100);
    else %initialize model parameters
        [model, loglikHist] = hmmFit(y(1:k)', U_vals(j), 'discrete', ...
            'convTol', 1e-12, 'maxIter', 100);
    end

    % Iterate model parameters
    pik = model.pi;
    Ak = model.A;
    Ek = model.emission;
    Tk = Ek.T;

    % Prediction via Viterbi path
    path = hmmMap(model,y(1:k)');
    HMMiter = Ak*Tk;
    spreadRow = HMMiter(path(end),:);
    [maxL, pred_vals(k2)] = max(cumsum(spreadRow) > rand);

    % Compare result to actual observation y(T)
    if pred_vals(k2) == true_vals(k2)
        numSuccesses = numSuccesses + 1;
    end

    % Margin function
    marginW(k2) = spreadRow( pred_vals(k2) );

    waitbar(k / (T_hat-1));
end

% Record statistics
classRate(j,b) = numSuccesses / (endTime - startTime);
correlations(j,b) = corr(true_vals, pred_vals);
[AUC(j,b), xplot, yplot] = ROC( [true_vals, pred_vals, marginW] );

% Spit out current results
close(h);
fprintf('%2u   %8f    %8f     %f\n', U_vals(j), ...
  classRate(j,b), correlations(j,b), AUC(j,b));
```

```
        fprintf(fid_out, '%2u   %8f    %8f     %f\n', U_vals(j), ...
          classRate(j,b), correlations(j,b), AUC(j,b));
    end
    fprintf('\n');
end

% Spit out results from baggin
fprintf('\n');
fprintf('After bagging the experiment %u times\n', B);
fprintf('using random samples of MATLAB generated data of %u observations\n', T);
fprintf('(with %u hidden states),\n', U);
fprintf('and discretizing observations into %u symbols,\n', V);
fprintf('the median values are:');
fprintf(fid_out, 'After bagging the experiment %u times\n', B);
fprintf(fid_out, 'using random samples of MATLAB generated data of %u observations\n', T);
fprintf(fid_out, '(with %u hidden states),\n', U);
fprintf(fid_out, 'and discretizing observations into %u symbols,\n', V);
fprintf(fid_out, 'the median values are:');
fprintf('\n');

fprintf('\n');
fprintf(' U  class rate  correlation  AUC\n');
for j = 1:J
    fprintf('%2u   %8f    %8f     %f\n', U_vals(j), ...
        median(classRate(j,:)), median(correlations(j,:)), median(AUC(j,:)));
    fprintf(fid_out, '%2u   %8f    %8f     %f\n', U_vals(j), ...
        median(classRate(j,:)), median(correlations(j,:)), median(AUC(j,:)));
end

% Visual result of bagging
boxplot(AUC');
  set(gca,'XTick',1:length(U_vals))
  set(gca,'XTickLabel',U_vals)
  xlabel('Number of hidden states')
  ylabel('Area under ROC')

% Wrap up output file
fprintf('\n');
time_stamp = datestr(now, 'dd-mmm-yyyy HH:MM:SS');
fprintf(fid_out, 'This file was saved on %s', time_stamp);
fclose(fid_out);
toc

function [ theMatrix ] = mkColStocMat( m,n )
% This function will produce a column stochastic matrix, where the columns
% a forced away from uniformity by rejection sampling.
e = (1/m)*ones(m,1); %vector of ones --> normalized
j = 1; %counter
```

```
h = waitbar(0, 'Making column stochastic matrix ...');
theMatrix = zeros(m,n);
tol = 1e-4; %tolerance

while j <= n
    thisCol = rand(m,1);
    thisCol = thisCol / sum(thisCol); %elements of this col. now equals one

    % Rejection sampling
    if norm(thisCol - e) > ( tol / sqrt(m) )
        theMatrix(:,j) = thisCol;
    end

    j = j + 1;
    waitbar(j/n);
end

close(h); %close wait bar
end

function [ AUC, xplot, yplot ] = ROC( M )
% This procedure will produce the a receiver operating charactertisic (ROC)
% graph and the associated area under the curve (AUC) assuming a two-class
% setup.  The algorithm is based on the work found at
% https://cours.etsmtl.ca/sys828/REFS/A1/Fawcett_PRL2006.pdf
% input: three-column matrix with true values (discrete), predicted values
% (discrete), and results from some scoring/margin function
% outputs: the AUC, and the x- and y-coordinates (for plotting purposes)

% Input check
[m n] = size(M);
if n ~= 3
    fprintf('Invalid data matrix for ROC formulation');
end

% Sort data by third column ("negative" column leads to descending order)
X = sortrows(M, -3);

% Find number of true and false positives
TP = sum(X(:,1) == X(:,2));
FP = m - TP;

% Prepare coordinates for plotting purposes
xloc = 0;
yloc = 0;
xplot = zeros(m+1,1); %going to leave the origin as a starting point
yplot = zeros(m+1,1);
if FP ~= 0
```

```
    xstep = 1 / FP;
else
    xstep = 1;
end
if TP ~= 0
    ystep = 1 / TP;
else
    ystep = 1;
end

% Scan through given results
AUC = 0;
for j = 1:m
    if X(j,1) == X(j,2) % true classification
        yloc = yloc + ystep;
        xplot(j+1) = xloc;
        yplot(j+1) = yloc;
    else
        xloc = xloc + xstep;
        xplot(j+1) = xloc;
        yplot(j+1) = yloc;
        AUC = AUC + xstep*yloc;
    end
end

% Here's some code that will test this function with NT data points
% NT = 123;
% M = [round(rand(NT,1)) round(rand(NT,1)) rand(NT,1)];
end
```

5.4 Acknowledgments

I wish to thank my adviser Professor Harish Bhat for suggesting a foray into hidden Markov models. He guided me through the statistical background, the experiment design, and the computational aspects for this capstone project. I also want to thank the reading committee–Professors Roummel Marcia and Boaz Ilan–for reviewing this report.

Earlier in my time as a graduate student, Professors David Noelle and Ming-Hsuan Yang were patient with me as I attempted preliminary projects about HMMs in their EECS courses.

Finally, I want to state how grateful I am for my parents and their support during my graduate school years and all of preceding times. This endeavor would not have been possible without them.

References

[1] J. S. Baras and L. Finesso, Consistent Estimation of the Order of Hidden Markov-Chains, Lecture Notes in Control and Information Sciences, 184 (1992), pp. 26–39.

[2] H. Bhat and N. Kumar, Comparing Exact Bayesian and BIC Markov Order Classifiers. `http://nscs00.ucmerced.edu/~nkumar4/BhatKumar2011.pdf`.

[3] L. Breiman, Bagging Predictors, Machine Learning, 24 (1996), pp. 123–140.

[4] A. M. Fraser, Hidden Markov Models and Dynamical Systems, Society for Industrial Mathematics, 1 edition ed., 2009.

[5] S. Lee and T. Lee, Robust Estimation for Order of Hidden Markov Models based on Density Power Divergences, Journal of Statistical Computation and Simulation, 80 (2010), pp. 503–512.

[6] B. G. Leroux and M. L. Puterman, Maximum-Penalized-Likelihood Estimation for Independent and Markov-Dependent Mixture-Models, Biometrics, 48 (1992), pp. 545–558.

[7] R. J. MacKay, Estimating the Order of a Hidden Markov Model, Canadian Journal of Statistics-Revue Canadienne De Statistique, 30 (2002), pp. 573–589.

[8] T. M. Mitchell, Machine Learning, The McGraw-Hill Companies, Inc., 1997.

[9] A. W. Moore, Hidden Markov Models. `http://www.autonlab.org/tutorials/hmm.html`.

[10] J. Pitman, Probability, Springer-Verlag, 1993.

[11] D. S. Poskitt and J. Zhang, Estimating Components in Finite Mixtures and Hidden Markov Models, Australian and New Zealand Journal of Statistics, 47 (2005), pp. 269–286.

[12] L. R. Rabiner, A Tutorial on Hidden Markov-Models and Selected Applications in Speech Recognition, Proceedings of the Ieee, 77 (1989), pp. 257–286.

[13] T. Ryden, Estimating the Order of Hidden Markov-Models, Statistics, 26 (1995), pp. 345–354.

[14] G. Schwarz, Estimating Dimension of a Model, Annals of Statistics, 6 (1978), pp. 461–464.

[15] T. Shinozaki and M. Ostendorf, Cross-Validation and Aggregated EM Training for Robust Parameter Estimation, Computer Speech and Language, 22 (2008), pp. 185–195.