**ECE593 Introduction to Machine Learning Homework 7**

**Hidden Markov Model**

Bo Lin

Introduction

A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be presented as the simplest dynamic Bayesian network. The mathematics behind the HMM were developed by L. E. Baum and coworkers. It is closely related to an earlier work on the optimal nonlinear filtering problem by Ruslan L. Stratonovich, who was the first to describe the forward-backward procedure.

In simpler Markov models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but the output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states. The adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; the model is still referred to as a 'hidden' Markov model even if these parameters are known exactly.

A hidden Markov model can be considered a generalization of a mixture model where the hidden variables (or latent variables), which control the mixture component to be selected for each observation, are related through a Markov process rather than independent of each other. Recently, hidden Markov models have been generalized to pairwise Markov models and triplet Markov models which allow consideration of more complex data and the modelling of nonstationary data.

Evaluation of Hidden Markov Models

Given an observation sequence *O* = {*O*1*O*2 · · ·*OT* } and a state sequence *Q* = {*q*1*q*2 · · · *qT* }, the probability of observing *O* given the state sequence *Q* is simply



which we cannot calculate because we do not know the state sequence.

There is an efficient forward-backward procedure to calculate P(O|λ), which is called the forward-backward procedure (see Fig.1). It is based on the idea of dividing the observation sequence into two parts: the first one starting from time 1 until time t, and the second one from time t + 1 until T.

We define the forward variable αt(i) as the probability of observing the partial sequence {O1 · · ·Ot} until time t and being in Si at time t, given the model λ:



It can be calculated recursively by accumulating results on the way.

Initialization:



Recursion



αt (i) explains the first t observations and ends in state Si . We multiply this by the probability aij to move to state Sj , and because there are N possible previous states, we need to sum up over all such possible previous Si . bj(Ot+1) then is the probability we generate the (t + 1)st observation while in state Sj at time t + 1.

The backward variable, which is the βt(i), which is the probability of

being in Si at time t and observing the partial sequence Ot+1 · · ·OT can be similarly defined as



Then the probability of the observation sequence is



The probability of the state sequence is



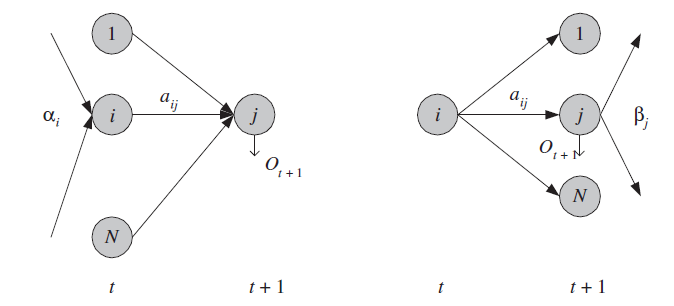


Figure 1

Learning Hidden Markov Model

The approach to learn an HMM from data is maximum likelihood, and we would like to calculate λ∗ that maximizes the likelihood of the sample of training sequences,  , namely, P(X|λ). We start by defining a new variable that will become handy later on.

We define ξt (i, j) as the probability of being in Si at time t and in Sj at time t + 1, given the whole observation O and λ:



Similarly here we have the Baum-Welch algorithm, which is an EM pro cedure. At each iteration, first in the E-step, we compute ξt (i, j) and γt (i) values given the current λ = (A, B,Π), and then in the M-step, we recalculate λ given ξt (i, j) and γt (i). These two steps are alternated until

convergence during which, it has been shown, P(O|λ) never decreases.

Assume indicator variables  as



and



These are 0/1 in the case of an observable Markov model and are hidden random variables in the case of an HMM. In this latter case, we estimate them in the E-step as



In the M-step, we calculate the parameters given these estimated values.



HW7.1. Learning HMM

In this homework, in the dataset ‘112\_A\_hmm.mat’, the first two matrices (data1 and data2) represent output of two Markov processes (lets call them Process 1 and Process 2) with 3 discrete outputs and unknown number of hidden states. Each matrix represents 20 samples of 200 consequtive observations. These data are to be used for learning Hidden Markov Models.

First, we divide the data1 and data2 into v subsets of equal size in order to perform a v-fold cross-validation . Sequentially one subset is tested using the classifier trained on the remaining v − 1 subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. In simulation, v is chosen as 4.

Secondly, the hmm model is trained using “dhmm\_em”(discrete hmm expectation maximization) function given random initial prior, transition and observation matrices. log likelihood of each validation subset data is calculated using “dhmm\_logprob” function and the mean of all log likelihood of test data is collected in “meanloglik”.

Next, the procedure is repeated using different number of hidden state from 2 to 30 to find the best possible number of hidden states.

Finally, the result is plotted shown in Fig.2 and Fig.3. We can see that starting at small hidden state numbers the likelihood is comparatively lower due to the rash of modeling. After we increase the number of hidden states, the result will tend to stable. In order to find the best parameter for later classification, for process 1, “state\_num” is set as 13, for process 2, “state\_num” is set as 25.



Figure 2



Figure 3

HW7.2. Process Classification

From HW7.1, we have the best possible number of hidden states. The hidden state numbers are then set as fixed. For process classification of vectors X1 to X6, the log-likelihood value of both processes is calculated using each vector. Then for classification, the vector will be classified in to the process that has a larger log-likehood.

In addition, I also tested the classification under different number of hidden states; the results always remain the same. The result is shown in Fig.4.



Figure 4

HW7.3 Bonus HMM classifier for Physical activity

For this bonus problem we want to create an activity classifier that will classify two activity states: walking and running.

We will construct activity classifier using a Gaussian mixture model, represented as a simple graphical model and constructed within the BNT framework. We will train this model using real research data. For the purposes of this tutorial it doesn't really matter what this data is or where it comes from. However, this is not a toy problem -- we will be working in a 31 dimensional feature space, and the results will be a generative model suitable for use in a real-time activity classification system.

By following the steps in the tutorial, we have the new trained Bayes net and inference engine. By comparing the synthetic data and training data in Fig. 5, we can visually compare the figures and see whether the model is reasonable. The two plots look similar.



Figure 5

After we have the model, we will do the classification. In order to test the performance of the model as a classifier, we will see how well it classifies our testing data.

We can see from Fig. 6 that the marginal for each class are at or near 1.0 in the appropriate regions of the data, and at or near zero elsewhere. Our Bayes net performs well as both a generative and discriminative model.



Figure 6