Applications of Hidden Markov Models

Andrew Floren

December 12, 2014

Introduction

Hidden Markov Models represent a system where there is a hidden or unobservable random variable that is linked to an observable variable. They allows us to calculate the probability distribution of the hidden state at some point in time given a sequence of observed variables. The model only requires two conditions on the hidden and observed variables. First, the dynamics of the hidden states obey the Markov property that the distribution of a hidden state at a particular point in time is conditionally independent given the value of the state at the previous point in time. Second, the observed variables must follow a similar property in that the distribution of an observed variable at a particular point in time must be conditionally independent given the value of the hidden state at that time. This model is particularly relevant for many engineering problems where we wish to estimate the value of some unknown variable given imperfect measurements but the dynamics of the variable are well known and can be modeled. There are a number of useful conditional probabilities to consider in this model and we will be examining three of them: the probability distribution of a state given all previous observations, the probability of a state given all previous and future observations, and the probability of the entire sequence of states given the entire sequence of observed variables. We will also be introducing algorithms for calculating these probability distributions.

Hidden Markov Model

A Hidden Markov Model can be formally defined as the following. Let $X_1, X_2, \cdots, X_t$ be a Markov chain on the discrete domain Ω with transition probability matrix $P \in \mathbb{R}^{|\Omega|\times|\Omega|}$ where $P_{i,j} = P(X_{t+1} = j | X_t = i)$. Let $Y_1, Y_2, \cdots, Y_t$ be a series of random variables on the discrete domain Φ with observation probability matrix $O \in \mathbb{R}^{|\Omega|\times|\Phi|}$ where $O_{i,j} = P(Y_t = j | X_t = i)$ and each Y_t is conditionally independent given X_t.

1 Filtering

Filtering is the problem of finding the probability distribution of a hidden state given all previous observations. Formally, we can define this distribution as

$$P(X_t|Y_1, Y_2, \cdots, Y_t). \tag{1}$$

However, let us first consider how to calculate the full joint probability:

$$P(X_t, Y_1, Y_2, \cdots, Y_t) \tag{2}$$

$$\begin{aligned}
P(X_t, Y_1, Y_2, \cdots, Y_t) &= P(Y_t|X_t, Y_1, \cdots, Y_{t-1})P(X_t, Y_1, \cdots, Y_{t-1}) && (3)\\
&= P(Y_t|X_t)P(X_t, Y_1, \cdots, Y_{t-1}) && (4)\\
&= P(Y_t|X_t)\sum_{X_{t-1}\in\Omega} P(X_t, X_{t-1}, Y_1, \cdots, Y_{t-1}) && (5)\\
&= P(Y_t|X_t)\sum_{X_{t-1}\in\Omega} P(X_t|X_{t-1}, Y_1, \cdots, Y_{t-1})P(X_{t-1}, Y_1, \cdots, Y_{t-1}) && (6)\\
&= P(Y_t|X_t)\sum_{X_{t-1}\in\Omega} P(X_t|X_{t-1})P(X_{t-1}, Y_1, \cdots, Y_{t-1}) && (7)
\end{aligned}$$

Note that $P(Y_t|X_t)$ and $P(X_t|X_{t-1})$ are defined directly by our model, and $P(X_{t-1}, Y_1, \cdots, Y_{t-1})$ is the same as equation 2 at time $t-1$. This implies some recursive algorithm exists for calculating the joint probability. To recover the conditional probability we calculate

$$P(X_t|Y_1, Y_2, \cdots, Y_t) = \frac{P(X_t, Y_1, Y_2, \cdots, Y_t)}{\sum_{X_t\in\Omega} P(X_t, Y_1, Y_2, \cdots, Y_t)}. \tag{8}$$

1.1 Forward Algorithm

The forward algorithm is an efficient solution to the filtering problem making use the recursion identified in the previous equations. Given observations $y_1, y_2, \cdots, y_t$, a probability transition matrix P, an observation probability matrix O, an initial state distribution π, and $\alpha_i(x) = P(X_i = x, Y_1 = y_1, \cdots, Y_i = y_i)$, the filtered distribution of state $X_t \in \Omega$ can be determined using Algorithm 1.

2 Smoothing

Smoothing is the problem of finding the probability distribution of a hidden state given all observations including those in the future. Unlike filtering, smoothing

Algorithm 1 Forward Algorithm

$alpha_0 \leftarrow \pi$
for $i \leftarrow 1$ **to** t **do**
  $\alpha_i \leftarrow \text{diag}(O_{*,y_t}) \cdot P \cdot \alpha_{i-1}$
end for
return $\frac{\alpha_t}{\|\alpha_t\|_1}$

can not be performed in real-time as it requires future observations. Formally we can define this distribution as

$$P(X_k|Y_1, Y_2, \cdots, Y_t), \tag{9}$$

where $0 < k < t$. Similar to filtering, let us begin by considering the full joint probability

$$P(X_k, Y_1, Y_2, \cdots, Y_t). \tag{10}$$

$$P(X_k, Y_1, \cdots, Y_t) = P(Y_{k+1}, \cdots, Y_t|X_k, Y_1, \cdots, Y_k)P(X_k, Y_1, \cdots, Y_k) \tag{11}$$
$$= P(Y_{k+1}, \cdots, Y_t|X_k)P(X_k, Y_1, \cdots, Y_k) \tag{12}$$

Note that $P(X_k, Y_1, \cdots, Y_k)$ is the unnormalized output of the forward algorithm up to time k. Therefore, we only need to solve

$$P(Y_{k+1}, \cdots, Y_t|X_k). \tag{13}$$

$$P(Y_{k+1}, \cdots, Y_t|X_k) = \sum_{X_{k+1} \in \Omega} P(Y_{k+1}, \cdots, Y_t, X_{k+1}|X_k) \tag{14}$$
$$= \sum_{X_{k+1} \in \Omega} P(Y_{k+1}, \cdots, Y_t|X_{k+1}, X_k)P(X_{k+1}|X_k) \tag{15}$$
$$= \sum_{X_{k+1} \in \Omega} P(Y_{k+1}, \cdots, Y_t|X_{k+1})P(X_{k+1}|X_k) \tag{16}$$
$$= \sum_{X_{k+1} \in \Omega} P(Y_{k+1}|X_{k+1})P(Y_{k+2}, \cdots, Y_t|X_{k+1})P(X_{k+1}|X_k) \tag{17}$$

$P(Y_{k+1}|X_{k+1})$ and $P(X_{k+1}|X_k)$ are defined directly by our model, and $P(Y_{k+2}, \cdots, Y_t|X_{k+1})$ is the same as equation 13 at time $k+1$. Again, this implies some recursive algorithm exists for calculating the joint probability. To recover the conditional probability we calculate

$$P(X_k|Y_1, Y_2, \cdots, Y_t) = \frac{P(X_k, Y_1, Y_2, \cdots, Y_t)}{\sum_{X_k \in \Omega} P(X_k, Y_1, Y_2, \cdots, Y_t)} \tag{18}$$

2.1 Forward-Backward Algorithm

The forward-backward algorithm is an efficient solution to the smoothing problem making use of both the forward algorithm and the recursion identified in the previous equations. Given observation $y_1, y_2, \cdots, y_t$, transition probability matrix P, observation probability matrix O, and initial state probability distribution π, $\alpha_i(x) = P(X_i = x, Y_1 = y_1, \cdots, Y_i = y_i)$, and $\beta_i(x) = P(Y_{i+1} = y_{i+1}, \cdots, Y_i = y_i | X_i = x)$, the smoothed distribution of state $X_k \in \Omega$ can be determined using Algorithm 2. Note that this algorithm can easily be modified

Algorithm 2 Forward-Backward Algorithm

$\alpha_0 \leftarrow \pi$
for $i \leftarrow 1$ **to** k **do**
  $\alpha_i \leftarrow \text{diag}(O_{*,y_i}) \cdot P \cdot \alpha_{i-1}$
end for
$\beta_t \leftarrow \mathbb{1}$
for $i \leftarrow t-1$ **to** k **do**
  $\beta_i \leftarrow \text{diag}(O_{*,y_{i+1}}) \cdot P \cdot \beta_{i+1}$
end for
return $\frac{\alpha_k \cdot \beta_k}{\|\alpha_k \cdot \beta_k\|_1}$

to efficiently calculate smoothed values for the entire sequence of hidden variables. Instead of performing the forward and backward recursive calculations to a specific time point, we can simply perform both operations on the entire sequence and store the results of each iteration in memory. Then to find the smoothed value for any particular time point, we multiply and normalize the results of the forward and backward procedure for that particular time.

3 Maximum Likelihood Sequence

Although smoothing calculates the probability distributions for all hidden variables given every observation, it cannot tell you what the most likely sequence of hidden variables was. Taking the sequence with the largest probability at each time step according to the smoothing algorithm will not yield this sequence as these probabilities were not conditioned on the previous state. Therefore, we want to find

$$\underset{x_1, x_2, \cdots, x_t \in \Omega}{\arg\max} P(X_1 = x_1, X_2 = x_2, \cdots, X_t = x_t | Y_1, Y_2, \cdots, Y_t). \tag{19}$$

Consider

$$\max_{x_1, \cdots, x_t} P(X_1 = x_1, \cdots, X_t = x_t, X_{t+1}, Y_1, \cdots, Y_t, Y_{t+1}) = \tag{20}$$

$$P(Y_{t+1}|X_{t+1}) \max_{X_t \in \Omega} ($$
$$P(X_{t+1}|X_t) \max_{x_1,\cdots,x_{t-1} \in \Omega} P(X_1 = x_1, \cdots, X_{t-1} = x_{t-1}, X_t,$$
$$Y_1, \cdots, Y_{t-1}, Y_t)). \quad (21)$$

Note the strong similarity to the filtering problem except we are taking the max across states rather than the sum.

3.1 Viterbi Algorithm

The Viterbi Algorithm is an efficient solution to the problem of finding the maximum likelihood sequence that was proposed by Andrew Viterbi in 1967 as a decoding algorithm for convolutional codes and has since been used in a variety of applications (Viterbi (1967)). Given observations $y_1, y_2, \cdots, y_t \in \Phi$, probability transition matrix P, observation probability matrix O, initial state distribution π, and $\alpha_t(x) = \max_{x_1,\cdots,x_{t-1} \in \Omega} P(X_1 = x_1, \cdots, X_{t-1} = x_{t-1}, X_t = x, Y_1 = y_1, \cdots, Y_{t-1} = y_{t-1}, Y_t = y_t)$, we can determine the maximum probability sequence $x_1, x_2, \cdots, x_t \in \Omega$ using Algorithm 3.

Algorithm 3 The Viterbi Algorithm

```
alpha_0 ← π
for i ← 1 to t do
  for all x such that x ∈ Ω do
    α_i(x) ← O_{x,y_i} · max(diag(P_{x,*}) · α_{i−1})
  end for
  x_i ← arg max_{x∈Ω} α_i(x)
end for
return (x_1, ··· , x_t)
```

4 Decoding fMRI

In my research, I am attempting to decode brain states from functional magnetic resonance imaging (fMRI) data. The level of neural activation in the brain is approximately captured by the blood-oxygen-level dependent (BOLD) signal. However, this signal is extremely noisy; the signal-to-noise ratio is generally on the order of 1. Therefore, we must integrate repeated measures to achieve good accuracy. Traditionally, this is done by averaging across blocks of time where the stimulus, and presumably the brain state, is constant. By leveraging a hidden Markov model, we can relax the constraints on the temporal structure of the stimulus and hidden brain states.

4.1 Hidden Brain States

The true brain state is a massively high dimensional variable that is dependent on the state of all of the neurons and other signaling pathways in the brain. This state space is approximately bounded by pn^2 where p is the number of signaling pathways and n is the number of neurons in the brain. The average adult human has around 86 billion neurons which makes working with the true state space intractable. Instead, we consider a much smaller state space defined by an indicator function on the true state. In our experiments, subjects are asked to make a choice or complete some task. Presumably, the choice they make or the way they complete the task is a function of the true brain state. Each choice then can be considered an indicator function on the true state space that is 1 when the brain's state is such that it would choose that option. In this way, we can construct an arbitrary number of indicator functions to explore numerous neural hypotheses.

4.2 Observable Signal

The BOLD signal is captured by the MRI machine every 2.5 seconds across the entire brain with a resolution of 2mm^3. The signal is produced by shifts in the level of oxygen concentration in the capillaries feeding neurons. As neurons spike more frequently, they consume more glucose and and oxygen. To compensate for this increased metabolism, the brain increases blood flow specifically to the area of high activation. The increased blood flow results in increased oxygen concentration that can be measured by the MRI machine. The signal is correlated with neural activity but it is smeared out in time by the hemodynamic response function (HRF). This function is a measure of how the blood flow responds to activation. Unfortunately, this signal is generally quite weak compared to the noise in the system. Noise includes thermal and electronic noise as well as noise introduced by heart rate and respiration rate which also influence the oxygen concentration.

4.3 Model

To accurately model the state transition probabilities we found that we needed to use a discrete number of previous states. The set of brain states we are interested in decoding is ω, where $|\omega| = 6$ for our stimulus. The domain of the hidden states $X_1, \cdots, X_t$ is $\Omega = \omega^k$, where $k = 6$ is the number of memory states we use. This results in an extremely large but sparse state transition matrix P. The domain of the observable sequence $Y_1, \cdots, Y_t$ is $\Phi = \mathbb{R}^n$ where n is the number of voxels we are measuring. We use a trained feed-forward neural network to approximate $P(X_t|Y_t)$ from the observed fMRI data (Richard and Lippmann (1991)). We can't generate an observation probability matrix because our observable domain is continuous, but by fitting a distribution to $P(Y_t)$ we can calculate $P(Y_t|X_t)$ using Bayes' rule. Fortunately, prior fMRI studies have shown that the distribution of fMRI data can be reasonably approximated by a

multivariate Gaussian after whitening (Worsley (2001)).

4.4 Modified Forward Algorithm

Based on this model, we propose the following modified forward algorithm for fMRI data. Given observations $y_1, y_2, \cdots, y_t \in \Phi$, a probability transition matrix P, an observation probability distribution $P_y(x)$ $\mathbb{N}(\mu, \sigma)$, a function $N : \Phi \to \Omega$ that approximates $P(X_t|Y_t = y_t)$, an initial state distribution π, and $\alpha_i(x) = P(X_i = x, Y_1 = y_1, \cdots, Y_i = y_i)$, the filtered distribution for the state $X_t \in \Omega$ can be determined using Algorithm 4. The performance of this algo-

Algorithm 4 Modified Forward Algorithm

$alpha_0 \leftarrow \pi$
for $i \leftarrow 1$ **to** t **do**
  $\alpha_i(x) \leftarrow \frac{N(y_t)P_y(y_t)}{\pi} \cdot P \cdot \alpha_{i-1}$
end for
return $\frac{\alpha_t}{\|\alpha_t\|_1}$

rithm should be estimated using a cross-validation procedure (Kohavi (1995)) because the distribution P_y and the function N need to be constructed on a training set.

5 Methods

To test the modified forward algorithm, we will be comparing it's performance to simple block averaging on previously collected fMRI data.

5.1 Stimulus

The stimulus is a realistic virtual environment that the subject views while inside the scanner. The subject's view movies through a virtual town and periodically stops while some number of human characters move onto the screen. The characters stay on the screen for 15 seconds before the view moves on. The subject is not instructed to perform any task and simply views the scene passively.

5.2 Subjects

Data was collected on 5 male subjects aged 25 to 57. For each subject, 2 sessions were collected.

5.3 State Space

The information we attempted to decode from the fMRI data was the number of characters visible on the screen. During the stimulus, the number of characters

presented varied from 1 to 6 so we have 6 hidden states. However, we also considered the 5 previous states in our model which led to a total state space of 6^6 states. The observable was the fMRI data which was the BOLD signal on a 80 x 80 x 40 grid of voxels sampled every 2.5 seconds. Each 2.5 second period is called a frame and after removing the periods of moving through town we are left with 288 frames for each session.

5.4 Baseline

For baseline comparison, a neural network was trained to estimate the number of characters presented in each frame. We measured the decoding accuracy of the network, or the percent of correctly estimated states. The estimated state is taken to be the state with the maximum posterior probability. The accuracy of the classifier was estimated using cross-validation to avoid any bias.

5.5 Block Averaging

We also performed simple block averaging for comparison. During each 15 second period that characters are presented, the number of characters does not change. Therefore, we can average the fMRI data across these 15 second blocks to provide a better measurement. A second neural network was trained to estimate the number of characters presented in each block from the time-averaged fMRI data. The performance of the classifier was estimated using cross-validation.

5.6 Modified Forward Algorithm

The probability transition matrix was constructed such that the state would stay the same for 6 frames and then switch to another state with equal probability. Here, we are utilizing our knowledge that the number of characters stays the same for 15 seconds. This results in an extremely sparse transition matrix which we implemented using a sparse matrix class instead of forming a functional mapping. We estimated $P(Y_t)$ as a multivariate Gaussian which we fit to the fMRI data using a least squares approach. Similarly, we estimated $P(X_t|Y_t)$ using a trained neural network exactly as we did for the baseline. The modified forward algorithm was then used to generate the distribution $P(X_t|Y_1, Y_2, \cdots, Y_t)$ for each frame. The state with the maximum posterior probability was selected as the estimated state for that frame. The accuracy of the model was estimated using cross-validation where both the training of the neural network and the estimation of the multivariate Gaussian were performed within each fold.

6 Results

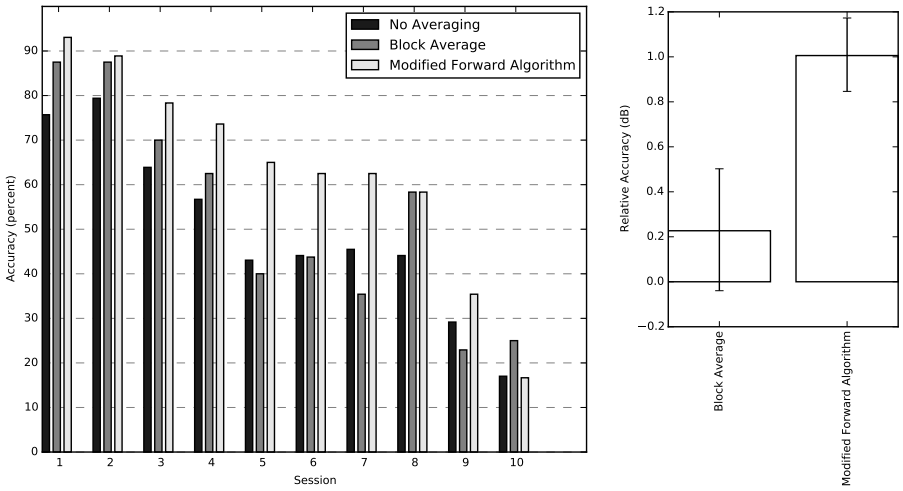

Figure 1. The results of the modified forward algorithm and basic block averaging on decoding presented character count from fMRI data in 10 different sessions. Results are reported in accuracy which is the percent of correctly decoded frames. Sessions are ordered by their baseline accuracy for easier readability and are not indicative of any kind of temporal trend. There is a significant variation in baseline performance which can be attributed to a variety of factors from scanner temperature to subject attentiveness. Therefore, to highlight the effect of the modified algorithm we also present relative accuracy (in dB) for basic block averaging and the modified forward algorithm averaged across frames. Error bars represent 68% confidence intervals produced by bootstrapping.

7 Conclusion

The results show that our modified forward algorithm is highly effective on an admittedly limited problem because the stimulus was still constructed using a rigid block structure. However, we believe these results show that the algorithm could be used to achieve good decoding performance even on a stimulus without a block structure. This opens the door for more naturally constructed stimuli which are important for therapy and training applications of fMRI. It should also be possible to construct modified forward-backward and Viterbi algorithms which could further improve decoding performance.

References

A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260–269, 1967.

M. D. Richard and R. P. Lippmann, "Neural Network Classifiers Estimate Bayesian a posteriori Probabilities," *Neural Computation*, vol. 3, no. 4, pp. 461–483, Dec. 1991. [Online]. Available: http://www.mitpressjournals.org/doi/abs/10.1162/neco.1991.3.4.461

K. J. Worsley, "Statistical analysis of activation images," in *Functional MRI: An Introduction to Methods*, 2001, p. Ch. 14.

R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *International Joint Conference on Artificial Intelligence*, 1995, pp. 1137–1145.

S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.