# Implementation of Hidden Markov Model

Gal Levy-Fix[*]

[*]Department of Biomedical Informatics, Columbia University

## Model Setup

A Hidden Markov Model (HMM) is implemented on a single simulated sequence with two normally distributed features. The model further assumes two hidden states. Two common HMM problems were evaluated: 1) learning and 2) decoding. The two tasks are described below, followed by a description of the data simulation and results.

## Learning

The first task implemented was to learn the model parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\phi}\}$ that best explain the observed data ($\boldsymbol{O}$). To do this, the model parameters were optimized using maximum likelihood via the expectation maximization (EM) algorithm. The EM algorithm provides an efficient way to maximize the likelihood function while avoiding direct maximization. In the E step of the algorithm the posterior distribution of the latent variable is found with initially selected model parameters: $p(\boldsymbol{Z}|\boldsymbol{O}, \boldsymbol{\theta^{old}})$. It is then used to evaluate the expectation of the complete- data log-likelihood function $Q(\boldsymbol{\theta}, \boldsymbol{\theta^{old}})$ presented below.

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta^{old}}) &= \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{O}, \boldsymbol{\theta^{old}}) \log p(\boldsymbol{O}, \boldsymbol{Z}|\boldsymbol{\theta}) \\ &= \sum_{i=0}^{N-1} \gamma_0(i) \log \pi_i + \sum_{t=1}^{T-1} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} \gamma_t(i,j) \log a_{i,j} + \sum_{t=0}^{T-1} \sum_{j=0}^{N-1} \gamma_t(j) \log b_j(O_t) \end{aligned}$$

Both $\gamma_t(j)$ and $\gamma_t(i,j)$, are evaluated in the E step of the algorithm and are defined in the following way:

$$\begin{aligned} \gamma_t(j) &= p(\text{state } j \text{ at time } t \mid O\ ,\theta^{old}) \\ \gamma_t(i,j) &= p(\text{state } i \text{ at } t-1 \text{ and state } j \text{ at } t \mid O,\ \theta^{old}) \end{aligned}$$

The notation of the model is as follows:

$N \equiv$ number of hidden states
$T \equiv$ number of time periods
$\pi \equiv$ initial probabilities for the hidden states with cell $\pi_i$, represents
the probability of being in state $i$ at time $t_0$
$A \equiv$ transition probability matrix with cell $a_{i,j}$ represents
probability of transitioning from state $i$ at time $t$ to state $j$ at time $t+1$
$\phi \equiv$ Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, where $\mu_i$ and $\Sigma_i$ are associated with the hidden state $i$
$b_j(O_t) \equiv p(O_t|\phi_j)$where $O_t$ is the observation at time t

To efficiently evaluate $\gamma_t(i)$ and $\gamma_t(i,j)$ the forward-backward algorithm (aka the Baim-Welch algorithm) is implemented. The forward algorithm allows for the calculation of $\alpha_t(i)$ for all (i) by starting at time $t = 0$ and propagating forward. Where $\alpha_t(i)$ for all (i) can be interpreted as the joint probability of observing all of the observations until time t and the hidden state (i) at time t. While the backward algorithm produces $\beta_t(i)$ for all (i) by starting at time $t = T$ and propagating backwards. Where $\beta_t(i)$ is the conditional probability of all future data from t+1 given the value of the hidden state (i). The forward-backward algorithm was implemented in log-scale to avoid underflow issues. The log-sum-exp trick is used when there is a log of a sum. The algorithm resulted in the following terms:

$$\alpha_t(j) = \begin{cases} \pi_j b_j(O_t), & \text{for } t = 0 \\ b_j(O_t) \sum_{i=0}^{N-1} \alpha_{t-1}(i) a_{ij}, & \text{for } t = 1,..,T-1 \end{cases}$$

In log space $\alpha_t(i)$ is:

$$\log \alpha_t(j) = \begin{cases} \log \pi_j + \log b_j(O_t), & \text{for } t = 0 \\ \log b_j(O_t) + \lambda_m + \log \sum_{i=0}^{N-1} e^{(\lambda_i - \lambda_m)}, & \text{for } t = 1,..,T-1 \end{cases}$$

Where $\lambda_i = \log \alpha_{t-1}(i) + \log a_{ij}$ and $\lambda_m = max\,\{\lambda_0, ..., \lambda_{T-1}\}$

The backward algorithm results in:

$$\beta_t(j) = \begin{cases} 1 & \text{for } t = T \\ \sum_{f=1}^{N} a_{jf} b_f(O_{t+1}) \beta_{t+1}(f), & \text{for } t = 0,..,T-2 \end{cases}$$

In log space $\beta_t(f)$ is:

$$\log \beta_t(j) = \begin{cases} \log(1), & \text{for } t = T \\ \lambda_m + \log \sum_{f=0}^{N-2} e^{(\lambda_f - \lambda_m)}, & \text{for } t = 1,..,T-1 \end{cases}$$

Where $\lambda_f = \log a_{jf} + \log b_f(O_{t+1}) + \log \beta_{t+1}(f)$ and $\lambda_m = max\,\{\lambda_0, ..., \lambda_{N-1}\}$

Then using the terms from the forward-backward algorithm $\gamma_t(i,j)$ can be expressed as:

$$\gamma_t(i,j) = \frac{\alpha_{t-1}(i) b_j(O_t) a_{ij} \beta_t(j)}{P(O|\lambda)}$$

Where, the likelihood function is obtained by $p(O|\lambda) = \sum_{i=0}^{N-1} \alpha_T(i)$

Moreover, $\gamma_t(i)$ can be obtained by summing over all states at time t:

$$\gamma_t(i) = \sum_{j=0}^{N-1} \gamma_t(i,j)$$

In log-space this expression is converted to:

$$\log \gamma_t(i,j) = \log \alpha_{t-1}(i) b_j(O_t) a_{ij} \beta_t(j) - \lambda_m - \log \sum_{i=0}^{N-1} e^{(\lambda_i - \lambda_m)}$$

where $\lambda_i = \log \alpha_{T-1}(i)$ and $\lambda_m = max \{\lambda_0, ..., \lambda_{N-1}\}$

In the M-step of the algorithm $Q(\theta, \theta^{old})$ is maximized with respect to the model parameters. This results in the following functional form:

The initial state probabilities are set to be: $\pi_i = \gamma_0(i)$

The transition probabilities are: $a_{ij} = \frac{\sum_{t=0}^{T-2} \gamma_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)}$

In logarithm form: $\log a_{ij} = \lambda_m + \log \sum_{t=0}^{T-2} e^{\lambda_t - \lambda_m} - \lambda_m^{'} - \log \sum_{t=0}^{T-2} e^{\lambda_t^{'} - \lambda_m^{'}}$

Where $\lambda_t = \log \gamma_t(i,j)$, $\lambda_t^{'} = \gamma_t(i)$, $\lambda_m = max \{\lambda_0, ..., \lambda_{T-2}\}$, and $\lambda_m^{'} = max \left\{\lambda_0^{'}, ..., \lambda_{T-2}^{'}\right\}$

The emission probabilities are obtained by updating the Gaussian parameters $\mu_i$ and $\Sigma_i$ defined below.

$$\boldsymbol{\mu}_i = \frac{\sum_{t=0}^{T-1} \gamma_t(i) \boldsymbol{x}_n}{\sum_{t=0}^{T-1} \gamma_t(i)}$$
$$\boldsymbol{\Sigma}_i = \frac{\sum_{t=0}^{T-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_i)(\boldsymbol{x}_n - \boldsymbol{\mu}_i)^T}{\sum_{t=0}^{T-1} \gamma_t(i)}$$

The parameters of the models are all iteratively updated until the number of maximum iterations is reached or if the log likelihood does not increase over the previous iteration.

## Decoding

Next, the Virterbi algorithm is utilized to obtain the most likely state sequence given the data and the model parameters. This problem is also called decoding and uses recursion (seen in the equations below) to efficiently searches for the most probably state path rather than extensively computing all possible paths. The probability of the most likely state sequence is recorded at each time point ($V_{j,t}$) for which (j) is its final state. The previous state ($i$) at time $t-1$ that produces the maximum probability for each final state ($j$) at time period ($t$) is also recorded. Once $V_{j,t}$ is calculated for the last time period the most probably state sequence is identified by tracing back the states leading to the last most probable state.

$$V_{j,t} = \begin{cases} max\left\{\pi_j b_t(j)\right\}, & \text{for } t = 0 \\ max\left\{V_{i,t-1} a_{i,j} b_t(j)\right\}, & \text{for } t = 1, .., T-1 \end{cases}$$

$$PrevState_{j,t} = \text{argmax}_{i=1,..,N-1}\ V_{j,t}(i)$$

The most probable state sequence is then used to classify observations over time to the different identified states. Using the classification summary statistics and data visualizations of the values associated with each state can be obtained to learn about the feature-profile of each state.

## Data Simulation

A single sequence of simulated data was generated following the generative process hypothesized by the model. The simulated data was composed of two features and 50 time period. To account for multiple data streams the model learning would iterate through multiple sets of data. To generate the data true parameters are selected $\theta^{true} = \{\pi^{true}, A^{true}, \phi^{true}\}$. For the sake of simplicity two states are assumed with two very distinct feature values with the following $\phi$ values:

$$\mu_1^{true} = [50, 55] \text{ and } \Sigma_1^{true} = [[10, 0], [0, 10]]$$
$$\mu_2^{true} = [-50, -55] \text{ and } \Sigma_2^{true} = [[10, 0], [0, 12]]$$

The probabilities of starting at each state and the transition probabilities were selected to be the following:

$$\pi^{true} = [0.95, 0.05]$$
$$A^{true} = [[0.95, 0.05], [0.05, 0.95]]$$

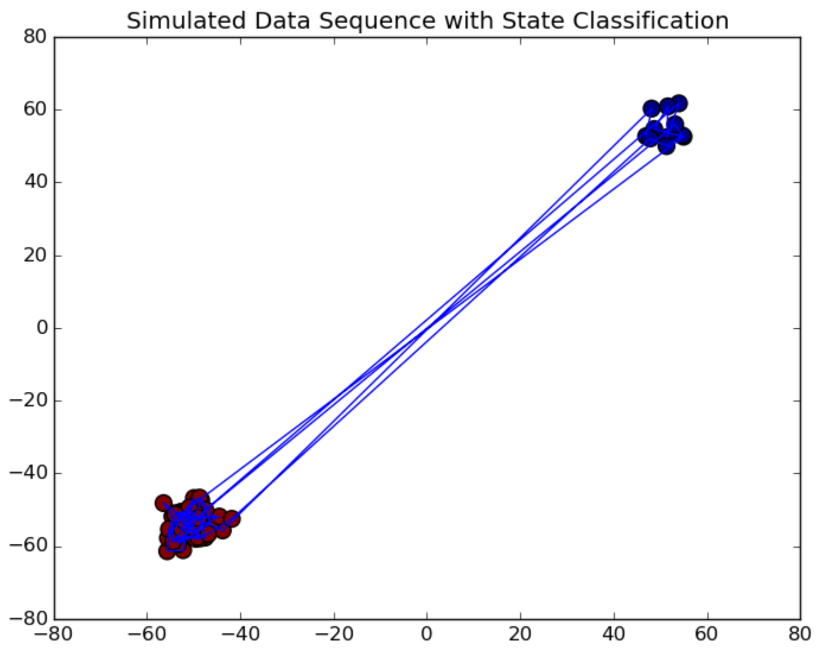
For the first period ($t_0$) a state ($z_i$) is selected with probability $\pi_i$. States $z_1, ..., z_{N-1}$ are selected according to the state transition matrix. Then according to the selected state, a data point is drawn from the multinomial normal $\mathcal{N}(\pi_i, \Sigma_i)$.

## Results

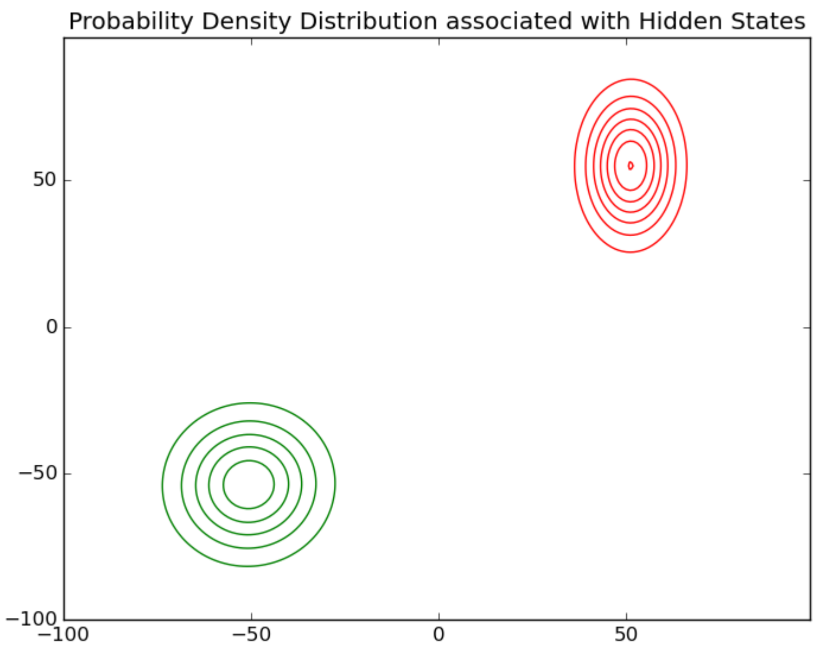
The data simulation described above resulted in the true state sequence presented below. As expected, the sequence starts with state 0 as the $\pi_0$ was set to be 0.95. Although the transition probabilities from one state to the other is only 5 percent, there are 5 transitions during the 50 time periods.

```
 [1] 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1
[39] 0 0 0 0 1 1 1 1 1 1 1 1
```

The model perfectly classified the simulated data into two states after only two iterations of the model. This is not surprising as the data was intentionally created to be very different in the two states. The state classification and the data sequence are visualized in the graph below:



The model parameters $\pi$ and $A$ learned by the model are $\pi^* = [1.0, 0.0]$ and $A^* = [[0.73, 0.27], [0.05, 0.95]]$. The next figure showcases the $\phi$ learned by the model and the associated probability density distributions. The contour map visualization of $\phi$ illustrates what feature values are associated with the hidden states.



## References

[1] Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Stump, Mark. *A Revealing Introduction to Hidden Markov Models*. 2004.