CS 698P: Markov Chains and their applications Hidden Markov Models and Kalman filters

Gopichand Kotana 14249 Shubham Pandey 14679

Introduction

A Markov model is used to model random processes. Real world processes produce observations which could be continuous or discrete. If we assume that there is a latent state s_t for each of these observations x_t, we obtain a hidden Markov model or a state space model depending on whether the latent state is discrete or a continuous vector. If the latent states take discrete values then it's called a HMM. If latent states are continuous vectors then it's called a state space model.
$s_n|s_{n-1} \sim P(s_n|s_{n-1})$, i.e, s_n satisfies the markovian property
$x_n|s_n \sim P(x_n|s_n)$, i.e, the observation x_n is dependent only on the hidden state s_n

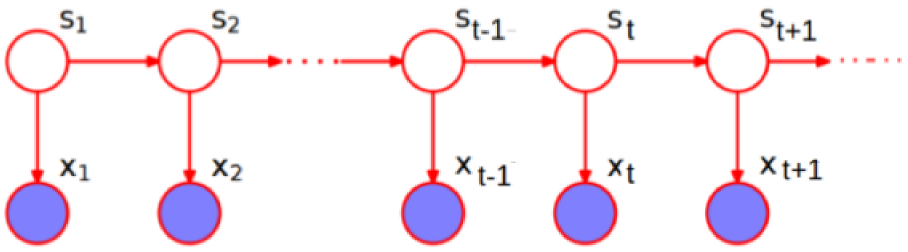


Figure 1: [4]

Hidden Markov Models

A hidden Markov model (HMM) allows us to infer the hidden events, which we believe are generating the observed events. We will see through an example the kind of problems we are interested in when we model a problem using a HMM. Consider a casino in which a dealer switches between two coins, a fair one and a biased one with a fixed unknown probability.

Let's assume that the probability of switching and, the probability of getting a head for each of the coin is known. After having observed for some time, how do we determine what is the most probable sequence of coin types used? What is the probability of the given sequence of observations to occur? Also, How do we learn the probability of coin switching and the probability of getting a head on the coins from the observations alone? These are the problems we give solutions to in the following sections.

Important problems in HMM

We formalize the problems we are dealing with, in this section.

- Given an observation symbol $O = O_1, O_2......O_T$ and a model $\lambda = \{A, B, \pi\}$. Calculate the probability of O efficiently.
- Given an observation symbol $O = O_1, O_2......O_T$ and a model $\lambda = \{A, B, \pi\}$. How do we choose state sequence $Q = \{q_1, q_2, ..., q_t\}$ such that probability of O is maximum.
- How do learn $\lambda = \{A, B, \pi\}$, such that $P(O/\lambda)$ is maximized.

Forward Algorithm

This is an efficient algorithm to calculate the probability of observations given the model parameters.

$$P(O) = \sum_Q P(O, Q) = \sum_Q P(O|Q)P(Q)$$

A trivial algorithm is to enumerate all possible T length hidden state sequences. If each state can take N different values then there are N^T possible sequences. We calculate the probability of the observations by summing over all the possible hidden state sequences. N^T is usually a very large number and hence it isn't practical to sum over all the possible hidden state sequences. The forward algorithm we describe below let's us do this in $O(N^2T)$

Forward Algorithm: Here we consider j length paths, and then compute probability of these j length paths(observation sequences) from 1 to t recursively. Define $\alpha_t(j)$ to be the probability of observations seen till t and the hidden state at t being j.

$$\alpha_t(j) = \{o_1, o_2....o_t, q_t = j|\lambda\}$$

We can write this recursively as,

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)a_{ij}b_j(i)$$

If q_F is the absorbing state, then, add probabilities corresponding to each state, after T iteration, to get P(O)

$$P(O|\lambda) = \alpha_t(q_F) = \sum_{i=1}^{N} \alpha_t(i) a_{iF}$$

Complexity $= O(N^2T)$

The Viterbi Algorithm

Here, we try to solve the second problem we have defined earlier. Given an observation symbol $O = O_1, O_2......O_T$ and a model $\lambda = \{A, B, \pi\}$. How do we choose state sequence $Q = \{q_1, q_2, ..., q_t\}$ such that probability of O is maximum.

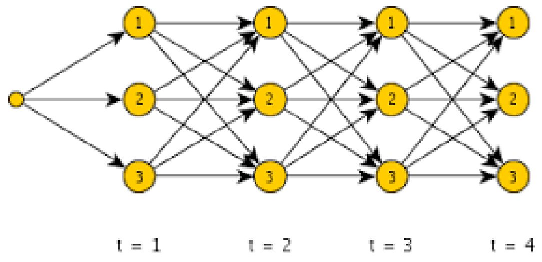


Figure 2: HMM for parameter estimation

One simple approach would be to consider all possible paths and choose path with highest probability. We need a more efficient algorithm which doesn't require us to enumerate all the possible paths. We use a dynamic programming based approach to achieve this.

We define $v_t(j)$ as the probability of the HMM to be in hidden state j at time t given the observations till t and the most probable sequence of hidden states till $t-1$.

$$v_t(j) = max_{q_0,q_1...,q_{t-1}} P(q_0, q_1..., q_{t-1}, o_1, o_2..., o_t, q_t = j|\lambda)$$

We calculate this quantity by recursively defining $v_t(j)$

$$v_t(j) = max_{i=1}{}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

At every time step we compute the quantity v_j for each state from the previous states using this recurrence. At time T, return the maximum of the $v_T(i)$ over all the states of the markov chain of the hidden states.

Parameter Estimation for HMM

This is considered to be the most important problem in HMM modeling. In general we don't know about the state transition probabilities and the emission probabilities of a given model. To understand the given problem properly we have to estimates it's latent parameters.

Let's considered our model has parameters $\lambda = \{A, B\}$. We are going to discuss two algorithms for estimation. One with a intuitive heuristic and the other is based on expectation maximization technique.

Given some observation sequence(O) of output we have to find $\lambda = \{A, B\}$ such that it maximizes the probability of observation sequence.

$$argmax_{\lambda} \quad P(O = o_1, o_2, ..., o_N|\lambda)$$

Maximizing Likelihood: The algorithm consist of two steps, first one is assigning some values to parameters, and second is estimating path from parameters. Following is the procedure followed.

Initialize model parameters($\lambda = \{A, B\}$) with some guess, and then calculate most prob-

able state path using Viterbi algorithm.

Until $\lambda = \{A, B\}$ converges, **repeat**:

1. Determine $\lambda = \{A, B\}$ using most probable state path.
2. Determine most probable path using $\lambda = \{A, B\}$ and Viterbi algorithm.

In parameter assignment step, we have both the information O,Q. we will use count based parameter estimation technique.

A_{kl} = Number of times each k to l hidden state transition has taken in the training sequences.

$E_{k(b)}$ = Number of times b is emitted from state k in the training sequences.

Compute a_{kl} and $e_{k(b)}$ as maximum likelihood estimators:

$$a_{kl} = \frac{A_{kl}}{\sum_m A_{km}}$$

$$e_{k(b)} = \frac{E_{k(b)}}{\sum_{b'} E_{k(b')}}$$

EM for HMMs: In this technique we will compute probability distribution of hidden state given HMM parameters and from that probability information we will again reestimate our parameters.

Similar to forward algorithm we are defining backward algorithm.

Backward Algorithm: Here we consider length T-j paths, and then increase j from 1 to t.

$$\beta_t(j) = \{o_{t+1}, ...o_T, q_t = j|\lambda\}$$

Where $o_{t+1}, ...o_T$ is all possible sequences of length T-j-1, occurring after o_t

$$\beta_t(j) = \sum_{i=1}^{N} \beta_{t+1}(i) a_{ij} b_j(o_t)$$

Now let's define a variable a'_{ij} which is approximation of a_{ij}.

$$a'_{ij} = \frac{\text{Expected number of transition from i to j}}{\text{Expected number of transition from i to any k}}$$

Now define another variable.

$$\epsilon_t(i,j) = P(q_t = i, q_{t+1} = j|\lambda, O)$$

represents probability of transition from i to j state given our observed sequence and parameters at given time t. To calculate a'_{ij} we can sum this variable over all possible time. To estimate ϵ we will define another term as ϵ' which is defined little differently than original one.

$$\epsilon'_t(i,j) = P(q_t = i, q_{t+1} = j, \mathbf{O}|\lambda)$$

we can break O in two part upto t and after t and combine it with q_t, q_{t+1} to get α_t, β_{t+1} respectively. hence-

$$\epsilon'_t(i,j) = \alpha_t \beta_{t+1} a_{i,j} b_j(o_{t+1})$$

$$P(O) = \alpha_T(q_F) = \beta_0(q_0)$$

Now consider following probability theory rule-

$$P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y)}$$

hence-

$$\epsilon_t(i,j) = \frac{\alpha_t \beta_{t+1} a_{i,j} b_j(o_{t+1})}{\alpha_T(q_F)}$$

And we can redefine a'_{ij} with help of above.

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \epsilon_t(i,j)}{\sum_{k=1}^{N} \sum_{t=1}^{T-1} \epsilon_t(i,k)}$$

Now for estimation of B, we will define another helper variables as stated below.

$$b'_j(v) = \frac{\text{Expected number of time observing v from state j}}{\text{Expected number of time reaching state j}}$$

$$\gamma_t(j) = P(q_t = j|O, \lambda)$$

$$\gamma_t(j) = \frac{(q_t = j, O|\lambda)}{P(O)}$$

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O)}$$

now from above this we can estimate b' variables using following technique. in numerator add γ_t only if observed data is same as corresponding variable of b'.

$$b'_j(v) = \frac{\sum_{t=1, s.t. o_t = v}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

With the help of above parameters we will define **Baum-Welch algorithm** as following-

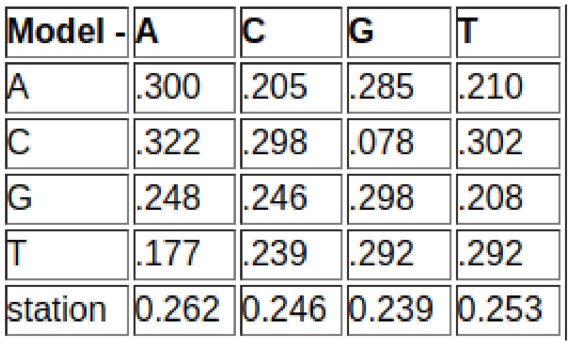
Initialize model parameters($\lambda = \{A, B\}$) with some guess, and then calculate most probable state path using Viterbi algorithm.

Until $\lambda = \{A, B\}$ converges, **repeat**:

1. E-step: we compute the expected state occupancy count γ and the expected state transition count ϵ from the earlier A and B probabilities
2. M step: we use γ and ϵ to recompute new A and B probabilities.

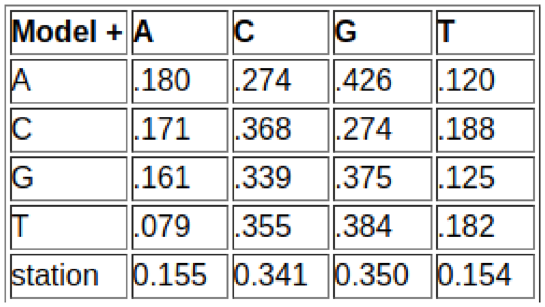
An application: Cpg Islands

- DNA sequences have four bases {C,G,A,T}.
- Human genome is 3billion base pair long but we have very few gene region out of that.
- Recognizing a gene region is very important problem.
- Each of the nucleotide is nearly equally likely to occur in DNA.



Model -	A	C	G	T
A	.300	.205	.285	.210
C	.322	.298	.078	.302
G	.248	.246	.298	.208
T	.177	.239	.292	.292
station	0.262	0.246	0.239	0.253

Figure 3: Frequencies away from CpG Island



Model +	A	C	G	T
A	.180	.274	.426	.120
C	.171	.368	.274	.188
G	.161	.339	.375	.125
T	.079	.355	.384	.182
station	0.155	0.341	0.350	0.154

Figure 4: Frequencies near CpG Island

Now we will lead toward our problem statement.

- Cosider Di-nucleotide, they are not equally likely, specially CG.
- CG is the least frequent dinucleotide because C in CG is easily methylated and has the tendency to change into T afterwards.
- However, the methylation is suppressed around genes in a genome.
- Hence identifying CpG Island is very useful.
- now consider + as region of CpG Island and - as other region.
  define: ${a_{s,t}}^{+} = \frac{{c_{s,t}}^{+}}{\sum_k {c_{s,k}}^{+}}$, where ${a_{s,t}}^{+}$ is probability of seeing t just after s and c is corresponding count.

From 3 and 4 we can observe the difference between probabilities of di-nucleotide CG. We will break our parameter estimation problem in two parts for better understanding.

Problem 1: Given a sequence find if it came from CpG Island.

Solution: Define two HMMs: one for CpG '+' and one for CpG '-' the architecture of

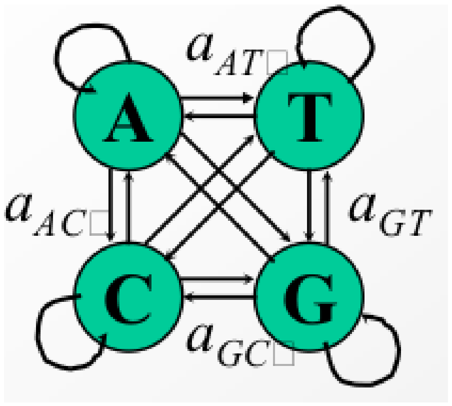


Figure 5: HMM for DNA bases

HMM is shown in 5.
Calculate probability of sequence considering each HMM using forward algorithm and output one with higher probability.

Now extending the problem statement for general purpose.

Problem 2: Given sequences identify regions corresponding to CpG Islands.
Solution: Build a single model that combines both the HMMs as shown in 7.
Now use Expectation Maximization algorithms to estimate parameters.
Then using Viterbi algorithm calculate most probable state sequences. And from state information calculate CpG Island.

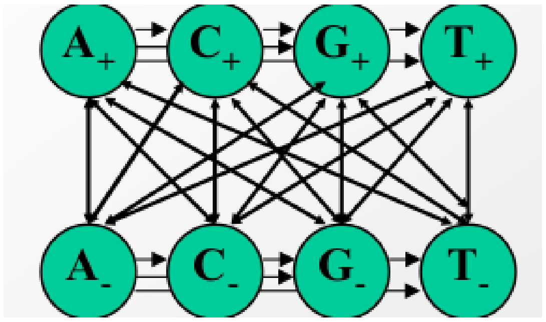


Figure 6: HMM for parameter estimation

Kalman Filters

In figure 1, if states s_n are continuous vectors, we get a state space model (SSM). x_n can either be continuous or discrete. We see through an example where such a model is used. Let $x_t \in R^2$ be the coordinates of a object and, $s_t \in R^6$ denotes the state vector of a body $s_t = [pos_1, vel_1, accel_1, pos_2, vel_2, accel_2]$, then from the kinematics equations we can write the following,

$$\mathbf{s}_t = \overset{\mathbf{A}_t}{\boxed{\begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & e^{-\alpha \Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & e^{-\alpha \Delta t} \end{bmatrix}}} \mathbf{s}_{t-1} + \epsilon_t$$

$$\mathbf{x}_t = \overset{\mathbf{B}_t}{\boxed{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}} \mathbf{s}_t + \delta_t$$

Figure 7: Example of a SSM [4]

The above equations can be used to model the location of an object. We are able to figure out the parameters of the above model by our domain knowledge. This is not always possible. Note that if we consider the noise ϵ_t, and δ_t to be Gaussian, then the above system mentioned is a linear Gaussian SSM, .

There are two ways in which inference of the hidden states is usually done

- Infer $p(s_t|x_1, x_2, ..., x_t)$, i.e, infer s_t using the observations we made till t. This is known as, filtering problem.
- Infer $p(s_t|x_1, x_2, ..., x_T)$, i.e, infer s_t using the all the observations we made. This is known as, smoothing problem.

We now look at the filtering problem.[4]
Consider a linear SSM, i.e, $s_t|s_{t-1} = A_t s_{t-1} + \epsilon_t$ and $x_t|s_t = B_t s_{t-1} + \delta_t$. If the noises are Gaussian with a zero mean, i.e, $\epsilon_t = \mathcal{N}(0, Q_t)$ and $\delta_t = \mathcal{N}(0, R_t)$, then the system turns into a linear Gaussian SSM with $s_t|s_{t-1} = \mathcal{N}(A_t s_{t-1}, Q_t)$ and $x_t|s_t = \mathcal{N}(B_t s_{t-1}, R_t)$. Now, consider this to be stationary, i.e, A_t, B_t, Q_t, R_t are independent of time.

We see how we can infer $p(s_t|x_1, x_2, ..., x_t)$ for such a system
By bayes rule,

$$p(s_t|x_1, x_2, ..., x_t) \propto p(x_t|s_t)p(s_t|x_1, x_2, ..., x_{t-1})$$

$$And, p(s_t|x_1, x_2, ..., x_{t-1}) = \int p(s_t|s_{t-1})p(s_{t-1}|x_1, x_2, ..., x_{t-1})ds_{t-1}$$

$$\implies p(s_t|x_1, x_2, ..., x_t) \propto p(x_t|s_t) \int p(s_t|s_{t-1})p(s_{t-1}|x_1, x_2, ..., x_{t-1})ds_{t-1}$$

We can see that the integral contains a term which is just the LHS of the final equation taken till time $t - 1$. This looks similar to the recurrence of $\alpha_t(j)$ defined for the forward algorithm in HMMs.
We know $p(x_t|s_t)$ and $p(s_t|s_{t-1})$ from the model specifications and we need to compute the term $p(s_{t-1}|x_1, x_2, ..., x_{t-1})$ to obtain our desired probability. The recursive relation present allows us to do what we have done in the forward algorithm in HMMs, i.e, compute the LHS till time j and extend it till time i by computing it at every time step. The analysis is true for any SSM. But, finding a closed form solution might not be possible for any random distribution chosen. That is the reason we assumed a Gaussian system. Gaussian distributions have properties which let's us find the closed form solution at every step of computation in the forward algorithm. Gaussian distributions are conjugate and the integral of two gaussians is also a gaussian. Hence, when we run our forward algorithm, it always results in a gaussian distribution for $p(s_j|x_1, x_2, ..., x_j)$. This property, ensured by the properties of Gaussian distributions, let's us find a close formed solution for Kalman filtering for Linear Gaussian SSMs.

References

[1] *Speech and Language Processing: Chapter 9* Daniel Jurafsky, James H. Martin
https://web.stanford.edu/~jurafsky/slp3/9.pdf

[2] *A tutorial on Hidden Markov Models and selected applications in speech recognition* Proceedings of the IEEE. 77 (2): 257–286, Lawrence R. Rabiner

[3] *Gene Discovery*
http://www.ics.uci.edu/~xhx/courses/CS284A/lectures/GeneDiscovery.pdf

[4] *Lecture 25 slide, PML, IITK by Piyush Rai*
https://web.cse.iitk.ac.in/users/piyush/tmp/pml_17/pml_lec25_slides.pdf