Hidden Markov Models for Machine Learning

Kubička Matěj

May 29, 2012

Abstract

This document describes what Hidden Markov Models (HMMs) are and how they can be used in Machine Learning for partially supervised learning. This document was written and submitted as a student analysis for the Machine Learning course given by professor Al-Ani Tarik at ESIEE Engineering.

1 Introduction

A classical example of Hidden Markov model was Given by L. E. Baum and his colleages at [2]. It is called *The Urn and Ball model*:

> *We assume that there are N (large) glass urns in a room. Within each urn there is a large number of colored balls. The physical process for obtaining observation is as follows. A genie is in the room, and according to some random process, he (or she) chooses an initial urn. From this urn, a ball is chosen at random, and its color is recorded as the observation. The ball is placed back to the urn and a new urn is selected randomly. Then the ball selection is repeated. Obviously, this entire process generates a sequence of random colors.*

We will see that this system can be modeled as simplest possible Hidden Markov model, where each state correspond to chosen urn. Each urn has its own probability distribution function (shortly *pdf*) for the color of newly taken ball.

First of all, we will define statistical modelling framework of Markov sources. Description of Markov Property, Markov Models, Markov Chains follows. Then the Hidden Markov Model will be defined, together with common algorithms used for their computation.

The second part of the paper discuss applications of Hidden Markov Models in Machine Learning.

1.1 Markov property

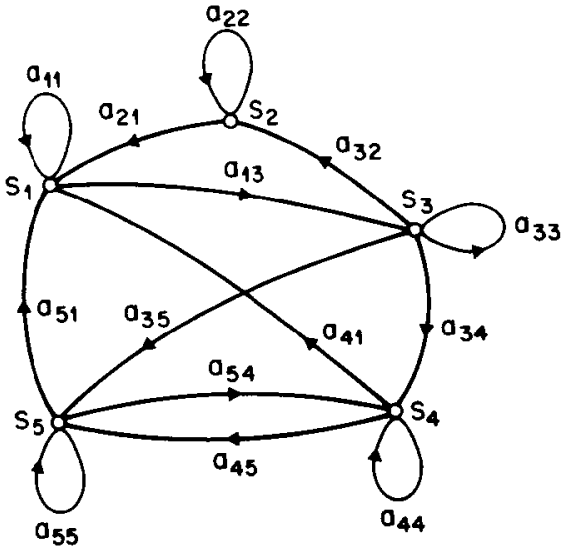
Suppose we have some random process with finite number of states. For example, to get back to Lamb's urns, chosen urn is that state. There are N urns in the room, and the genie chooses urns randomly one at a time. The resulting sequence of chosen urns is a random sequence of states.

Markov property is a property of randomized sequences like the one with urns. It is satisfied if conditional probability of future states depends only on present state (and therefore not on previous states). As a consequence, Markov property is memoryless and allows us to generate some kind of forecast for future states based only on the current state.

1.2 Markov Model

Process of generating random sequence which satisfy Markov property is called *Markov Process*. If this process is discrete in time, autonomous and fully observable, we use term *Markov Chain*. Good example of a simple Markov Chain could be a coin tossing experiment, where Heads and Tails are two different states and a sequence of repeated coin tossing is our Markov Chain.

I will borrow next example from [1]. Consider a system which may be in one of 5 distinct states $S_1, S_2, \ldots S_5$ at any time. The system is required to change state at every discrete time instant (possibly to the same state). The system (outlined in figure below) is clearly a Markov Model. Full probabilistic description of that system is given by probability of transition from any current state to any new state. In other words, we need $N \times N$ square matrix $\mathbb{A}$, where each element a_{ij} is probability of traversal from state i to state j.



This kind of probability matrix has to obey standard probabilistic constraints.

It means that following equalities and inequality on matrix $\mathbb{A}$ has to hold:

$$a_{ij} = P(S_j|S_i)$$

$$0 \leq a_{ij} \leq 1$$

$$\sum a_{ij} = 1$$

Additionally to matrix $\mathbb{A}$, we need initialization vector $\pi \in [1 \times N]$ which describes probability of starting in i-th state ($1 \leq i \leq N$)

The system on figure above, described by some matrix $\mathbb{A}$, is clearly a Markov Model, since all its states are directly observable. This model generates sequence of states according to the matrix $\mathbb{A}$ and the Markov property. The algorithm for generating the Markov chain is as follows:

1. Choose initial state, set time $t = 1$
2. Choose next symbol, according to largrest traverse probability rule
3. Transit to a new state
4. Return to step 2 and repeat

This algorithm is rather simple, thanks to Markov property. She allows us to forget past and choose next state only according to currently most probable option. In next chapter, we will look at Markov models with little adjustment - the states won't be observable anymore. We will see that it complicates things a bit, but also gives more realistic modelling properties.

2 Hidden Markov Model

When the states of the Markov Models are hidden and we cannot directly observe them, we decide to observe some other physical phenomenon, which gives us information about the latent states. Of course, we cannot say for sure what the hidden states are, but we can make a guess. This Markov Model is called *Hidden Markov Model* to emphasize the hidden nature of its states.

In real-life problems, Markov Models seem to have only rarely observable states. On the other hand, the certainty that observed state links to specific hidden state can be quite high. That is typically consequence of some physical constraints and the nature of the problem. (for example, some, but not all, states can be observable, which narrows possible candidates for resulting hidden states)

2.1 From Markov model to Hidden Markov model

Hidden Markov model is really just a extension of Markov Model. Observable Markov model is described by number of states N, the transition matrix $\mathbb{A}$ and by initialization vector π. Hidden Markov model is described by N, $\mathbb{A}$ and π too, but additionally we need to deal with probabilistic mapping from observable phenomenon to hidden (latent) states. There is no way to say for sure what is the real hidden state, but we can make a guess by watching its consequences. In order to properly define Hidden Markov model, we need following (naming convention taken from [1]):

- Number of states N
- Number of observations M
- Traversal probability matrix $\mathbb{A}$
- Emission probability matrix $\mathbb{B}$
- Initial state π
- Final time T

Note: this text uses packed notation of Hidden Markov model: $\lambda = (N, M, \mathbb{A}, \mathbb{B}, \pi)$

Parameters N, $\mathbb{A}$ and π have the same meaning as before. New parameters M and $\mathbb{B}$ constitute kind of a wrapper around original Markov model. At each time instance, the Hidden Markov model is in one of N states and as a consequence of being in that state, we observe one of M observable symbols. In other words, Hidden Markov models are like doubly embedded probabilistic systems.

Lets define M observations as $O_1, O_2, \ldots O_M$. Element b_{ij} of matrix $\mathbb{B}$ describes probability that state S_i is expressed by observation O_j:

$$b_{ij} = P(O_j|S_i)$$

Obviously, same probabilistic constraints apply to matrix $\mathbb{B}$ as they apply to matrix $\mathbb{A}$. Size of $\mathbb{B}$ is $M \times N$.

In the beginning of this chapter was stated that Hidden Markov model is an extension of the Markov model. It is simple to prove by showing that special matrix $\mathbb{B}$ causes the Hidden Markov Model to become a Markov Model. Just consider $\mathbb{B}$ to be unit matrix. Then for each hidden state we have one unique observation. The hidden states are no longer hidden.

2.2 A note on usability

Biggest strength of Hidden Markov model is in Markov property. It is assumed that the past do not affect the future. The decision is always based only on present, not past. This simplifies things a lot and gives us a powerful

framework to model many real life problems. On the other side, using it to model physical phenomena where Markov property doesn't hold well leads to misleading models. For example, if some model is in state where the largest traversal probability from this state is back to the same state, the model get stuck, infinitelly looping back. This *de facto* removes the random character of the modelled process.

Rabiner in [1] identifies 3 fundamental problems which needs to be solved in order to make Hidden Markov models usable in real-life applications:

1. What is the probability of observed sequence?
2. What is the (most probable) hidden state sequence, based on given observation?
3. How to adjust model parameters to maximaze probability of observed sequence?

2.3 Calculating probability of observed sequence

Lets suppose we have model and we want to calculate probability of some observed sequence. The straightforward approach is to use joint probability of observed and hidden state sequences. The algorithm goes like this: we calculate probability of every possible state sequence happening together with the observed sequence. Then we sum those probabilities. It is simple enough and this algorithm does the job correctly. Nonetheless, it is exponential in time - the algorithm is not computationally feasible even for really small models. For detailed description see [1].

A computationally feasible algorithm was discovered in 1980s. It is called Forward procedure [3] [4] [1]. Lets suppose we have observed sequence $E = r_1 r_2 r_3 \dots r_T$, where r_t is observed symbol at time t and a state sequence $Q = q_1 q_2 q_3 \dots q_T$. We define so called "forward variable" α as:

$$\alpha_t(i) = P(r_1 r_2 r_3 \dots r_t, q_t = S_i | \lambda)$$

Simply said, $\alpha_t(i)$ is probability that for observed sequence $r_1 r_2 r_3 \dots r_t$, the internal state at time t is S_i. Notice that forward variable takes only part of observed sequence. It is a sequence covering all steps until time t, it don't go further to the "end of time" T.

Having this forward variable $\alpha_t(i)$ allows us to apply recursive dynamic programming techniques. The algorithm is as follows:

1. Initialize $\alpha_1(i) = \pi_i b_{i1}$
2. Calculate recursively $\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij}$ until $t = T$
3. Repeat previous step for every $i \in \{1..N\}$
4. Finalize $P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i)$

The algorithm first calculates forward variable $\alpha_t(i)$ for all the internal states and then sums them together. This works only because there is only finite number of states and thus all possible sequences remerge from those states no matter the size of observation sequence. Asymptotic complexity of this algorithm turns out to be $O(N^2T)$, which can be considered as polynomial (cube) of the larger variable in the expression.

2.4 Calculating most likely sequence of states

Lets suppose we have following setup: an observed sequence $E = r_1 r_2 \dots r_T$ and Hidden Markov model $\lambda = (N, M, \mathbb{A}, \mathbb{B}, \pi)$. Now, the question to answer is what is the most likely sequence of hidden states $S = q_1 q_2 \dots q_T$.

What we do here is trying to find state sequence S which maximize probability of being cause of observation O. In other words, following:

$$\arg\max_S P(S|O, \lambda)$$

The probability of $P(S|O, \lambda)$ is the equivalent to joint probability of occurence of S and O together, hence:

$$P(S|O, \lambda) = P(S, O|\lambda)$$

And so we can fully solve our problem by using following equation:

$$S^* := \arg\max_S P(S, O|\lambda)$$

We want some efficient algorithm to find the solution for this equation. The naive approach would be to calculate the joint probability for all the possible states, but that would be, once again, at least asymptotically exponential in time. Luckily, there is more efficient approach called the Viterbi algorithm.

2.4.1 The Viterbi algorithm [5] [6] [1]

It is similiar approach like Forward algorithm described before. It uses incremental construction of the most likely path.

Heart of the algorithm lies in recursive formula expressing maximum probability along a single path at time t. Lets suppose we have observation symbol k given by observation r_t. Then, based on this observation we can define following recursion:

$$\delta_t(j) = b_{jk} \cdot \max_{1 \le i \le N} (\delta_{t-1}(i) a_{ij})$$

The $\delta_t(j)$ is the highest probability along a single path at time t, which accounts for first t observations and ends in state j. The b_{jk} is probability that observed symbol k is a consequence of hidden state j. The $\max(\delta_{t-1}(i)a_{ij})$ selects path which ends at time $t-1$ in state i with maximum transition probability into new state j.

The algorithm itself needs to initialize $\delta_0(j)$ and then recursively calculate partial results until final $\delta_T(j)$ is found. The problem is that this gives only probability of most likely path. In order to retrieve inner state sequence, we have to store state at every step into some data structure (like array).

For more details see [6], [5], or [1]. The algorithm has asymptotic complexity $O(N^2T)$, like the Forward algorithm.

2.5 How to train Hidden Markov model

This is a complicated task - what we try to achieve is to change parameters $(N, M, \mathbb{A}, \mathbb{B}, \pi)$ of Hidden Markov model λ in order to maximize likelihood of some input. There is a number of methods to do that.

Supervised learning with complete training data is easy - we can solve it analytically. If we have training observation sequences and corresponding hidden states available, then we can calculate probabilistic matrices $\mathbb{A}$, $\mathbb{B}$ and initialization vector π (we suppose that M, N are known). Unfortunatelly, that is almost never the case.

In case when we have only a set of labeled observation sequences, we have to deploy somewhat more advanced learning techniques. There are several approaches - for example Maximum Likelihood [1], Viterbi learning, Expectation-Maximization, or Simulated Annealing.

Viterbi training and Maximum Likelihood methods fits models to data by re-estimating the model parameters to increase likelihood. The training goes until the result starts to converge to proper identification. Note: For Viterbi training, the result converge only if the system has adequate initialization vector.

Expected-Maximization (EM) method is designed to cooperate with some "fuzzy", unknown, prameters. It finds locally optimal parameters of the model to maximize training data likelihood. It doesn't try to decode the hidden state sequence explicitely (unlike the Viterbi method). EM also critically depends on initialization, because it only finds local maximum by re-estimating unknown parameters (like Viterbi method).

3 Machine learning applications

Now, we have answer for the three fundamental questions: (1) given the model, how likely is observed sequence. (2) given the model and observation sequence, what is most likely the sequence of hidden states. (3) How can we adjust parameters of the model to maximize likelihood of observed sequence.

It all boils down to machine learning method which takes advantage of Hidden Markov modelling capabilities. Lets suppose we have two systems, one is real-life phenomenon and the second is it's Hidden Markov model. Lets assume that Markov property holds well enough for the real-life phenomenon and also assume that the observations are statistically independent. Then we can train our own model.

At first, we create a training data - a set of observed sequences. Additionally, we label all the sequences with appropriate events (suppose we have K events). For each label we create a Hidden Markov model λ_i (where $i \in 1, \ldots, k$) and train it with one of the methods listed above (Viterbi, Maximum likelihood, EM, ..)

When we have trained the model, we can use it. There are two things to do. Firstly, for any given observation sequence, decode the most likely sequence of hidden states by using the Viterbi algorithm. Secondly, for any observation sequence calculate "most likely" event by calculating:

$$\lambda_i^* = \arg \max_{1 \leq i \leq K} P(O|\lambda_k)$$

The λ_i^* is our detected event.

References

[1] Dr. Lawrence Rabiner, *"A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition"*. Proceeding of the IEEE, Vol. 77, No. 2, Frebruary 1989

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss *"A maximization technique occuring in the statistical analysis of probabilistic functions of Markov Chain"*. Ann. Math. Stat., vol. 41, no. 1, pp. 164-171, 1970

[3] L. E. Baum and J. A. Egon, *"An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology"*. Bull, Amer. Meteorol. Soc., vol. 73, pp. 360-363, 1967.

[4] L. E. Baum and G. R. Sell, *"Growth functions for transformations on manifolds"*. Pac. J. Math., vol. 27, no. 2, pp. 211-227, 1968

[5] A. J. viterbi, *"Error bounds for convolutional codes an asymptotically optimal decoding algorithm"*. IEEE Trans. Informat. Theory, vol IT-13, pp. 260-269, Apr. 1967.

[6] G. D. Forney, *"The Viterbi Algorithm"*. Proc. IEEE, vol. 61, pp. 268-278, Marc. 1973