Analytics Vidhya · Follow publication

Viterbi algorithm for prediction with HMM — Part 3 of the HMM series

5 min read · Jul 21, 2019

Raymond Kwok Follow

Listen Share

How the best hidden state chain is obtained from all possible ways?

Disclaimer: this is what I have learnt about HMM in the past few days and written in a flow that I think easier to understand, and that is definitely not a systematic way a text book would offer you. The text books are always our friends which give a complete, and structured way to learn, and materials are presented without loss of generality, but I hope my article could give you a hand to get through some bottlenecks in your journey. Good luck!

Part 1: Architecture of the Hidden Markov Model
Part 2: Algorithm to train a HMM: Baum-Welch algorithm
Part 3: Algorithm to predict with a trained HMM: Viterbi algorithm

In the first article, I talked about the architecture and the parametrization of the Hidden Markov Model (HMM), and the meaning of variables that I will use here. In the second article, it was about the training algorithm for HMM. In this one, the focus will be on the prediction algorithm, which is called the Viterbi algorithm.

Like Baum-Welch algorithm, our training algorithm, the Viterbi algorithm is also a dynamic programming approach. As a reminder, the dynamic programming is an approach that you would reuse calculated result in the next calculation, and the act of reusing saves time! What's more, it is indeed a natural choice for HMM, take the

simple HMM as an example, in which any state depends on its first previous state. This statement, however, does not mean that there is no effect at all from the rest of the previous states to the current one, but their effects are all absorbed into the first previous state, and this first previous state becomes the only factor you need to consider when transiting to the next state.

Therefore, you may also say that, in a simple HMM, any state depends on all previous state via the first previous state, and I said dynamic programming is a natural choice here because it is also a mean to capture everything in the past and be re-used in the future.

Viterbi algorithm

The purpose of the Viterbi algorithm is to make an inference based on a trained model and some observed data. It works by asking a question: given the trained parameter matrices and data, what is the choice of states such that the joint probability reaches maximum? In other words, what is the most likely choice given the data and the trained model? This statement can be visualized as the following formula, and obviously, the answer depends on the data!

$$X^*_{0:T} = \underset{X_{0:T}}{\operatorname{argmax}} P[X_{0:T}|Y_{0:T}]$$

this says to find the states that maximize the conditional probability of states given data.

To find the best set of states, the following recursive formula is used. I recommend this YouTube Video and this YouTube video for its deviation.

$$\mu(X_k) = \max_{X_{0:k-1}} P[X_{0:k}, Y_{0:k}] = \max_{X_{k-1}} \mu(X_{k-1}) P[X_k|X_{k-1}] P[Y_k|X_k]$$

mu function. It depends on its previous step, the transition and the emission matrices.

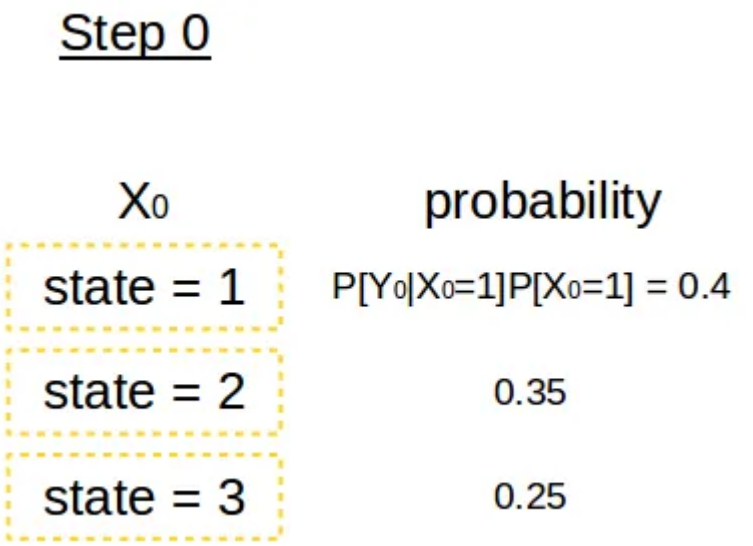
Let's substitute k=1,2,3 so that we can understand this recursion easily.

$$\mu(X_0) = P[Y_0|X_0]P[X_0]$$
$$\mu(X_1) = \max_{X_0} \mu(X_0)P[X_1|X_0]P[Y_1|X_1]$$
$$\mu(X_2) = \max_{X_1} \mu(X_1)P[X_2|X_1]P[Y_2|X_2]$$
$$\mu(X_3) = \max_{X_2} \mu(X_2)P[X_3|X_2]P[Y_3|X_3]$$

The first formula is the starting *mu* function, and results in a probability distribution for seeing different initial state given the initial observed data. Here, we do not constrain ourselves to any of these initial state, but we determine that in the next formula.

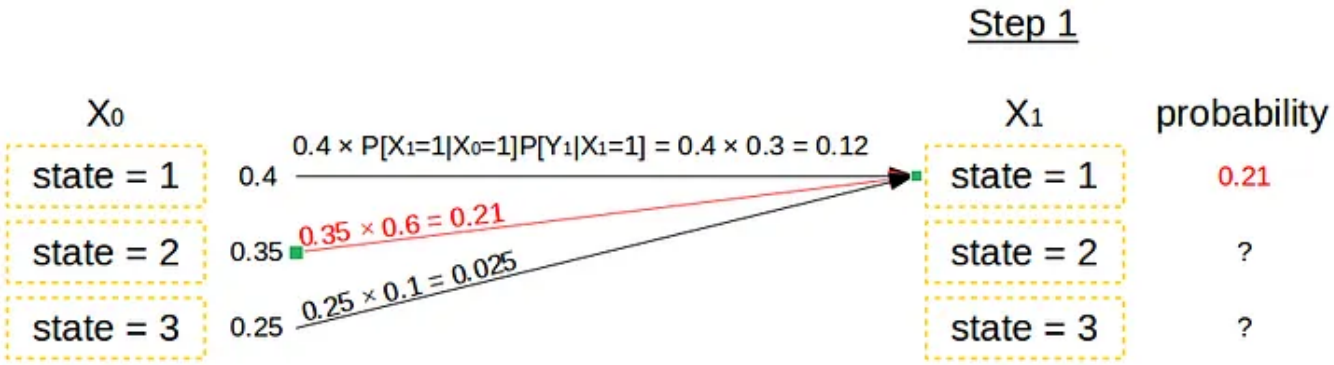
The second formula picks up the best initial state that maximize the product of the terms in the right hand side, and leaving the first state as a free parameter to be determined in the third formula. Similarly, the third formula picks the best first state, then leave the second state for the fourth formula.

Let us visualize these repeated processes in a diagram called trellis. In the diagram, we could see how each state is being chosen based on the rule of maximizing the probability, and to keep the diagram small, I would take an assumption that there are only three possible states for us to choose from at each time step. The same idea applies to any number of states, of course.

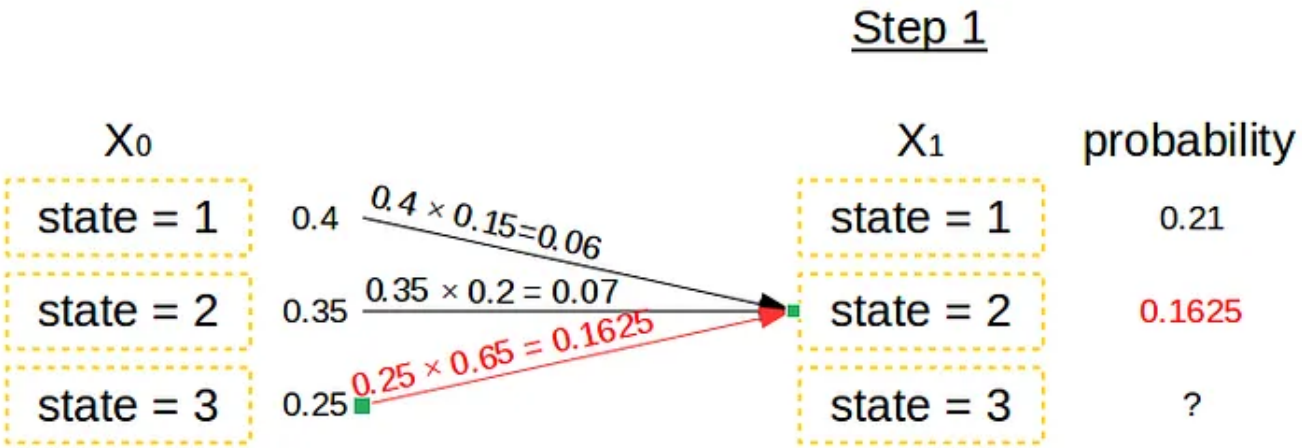


step 0. All three possible states are listed out for the initial state at time 0. The corresponding probability is assumed to be some actual numbers to facilitate the following discussion.

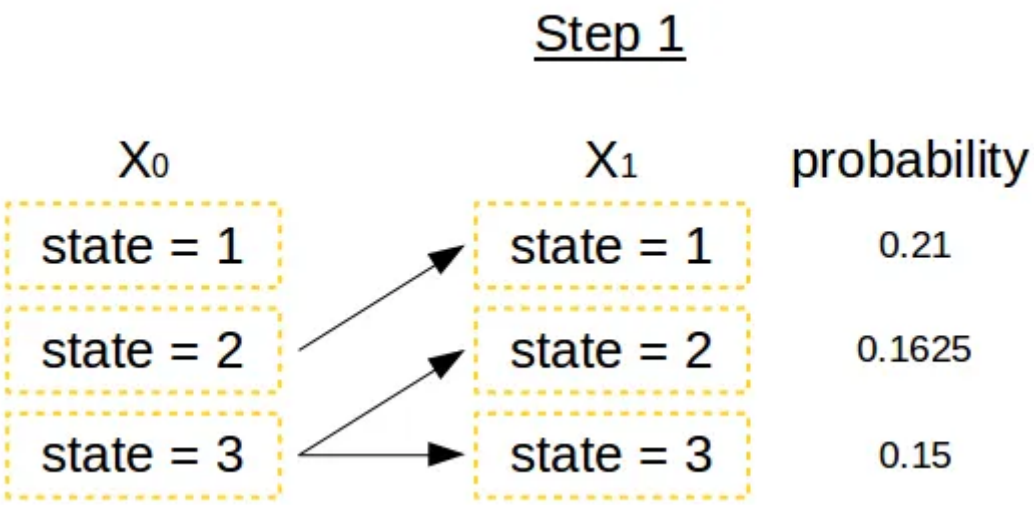
The step 0 is to just list out all possible state at time 0, and their corresponding probability values, and we do not decide which state is chosen at this stage.



step 1–1. For each possible first state, the best possible initial state is chosen. We find that initial state = 2 is most likely to lead to first state = 1.

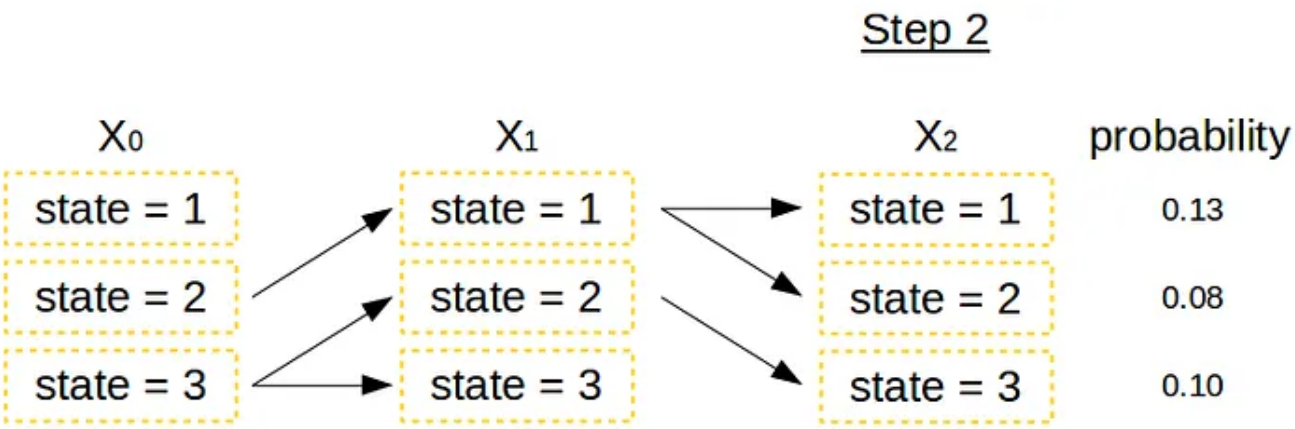


step 1–2. Here we find that initial state = 3 is most likely to lead to first state = 2.



step 1–3. We end Step 1 by finding the initial states that are mostly likely leading to the first state, and remembering their probability values which will be re-used in the next step.

In step 1, we iterate through all possible first state (state at time = 1), and find out their corresponding best initial state (state at time = 0) as illustrated in the above graphs. We then repeat the same procedure and finish step 2.



step 2. Repeating the procedure to go from step 0 to step 1 to get step 2 from step 1.

Now we begin to see some paths. For example, if we end the inference at step 2, then the most likely ending state would be state = 1, and the rest of the previous states could be back-traced through the arrows, which are state 2 at time 0, state 1 at time 1, and state 1 at time 2. The second likely path is 3–2–3, and the least likely path is 2–1–2. It is very unlikely that the path starts with state 1.

Summary

The Viterbi algorithm is an efficient way to make an inference, or prediction, to the hidden states given the model parameters are optimized, and given the observed data. It is best visualized by a trellis to find out how the path is select from a time step to the next. This ends the introduction for the series of Hidden Markov Model.