Task 8: Viterbi algorithm

Step 1: Viterbi

Imagine you observed a sequence of dice rolls produced by two dice: fair and loaded. Can you figure out when a fair dice was rolled and when a loaded one? If you have a Hidden Markov Model of the rolls, you can use the Viterbi algorithm to calculate the most likely sequence of hidden states underlying an observed sequence.

In this task, you will implement the Viterbi algorithm for the dice rolls. In the next task, you will be asked to use it for a different type of data, so you might want to make your code flexible, e.g., not hard-coding the states.

The Viterbi algorithm operates in time steps, where each step corresponds to an element in the sequence. Throughout the algorithm, we maintain two data structures, which keep track of:

1. the most probable previous hidden state for each of the possible current hidden states (suggested: `List[Dict[str,str]]`). This was called ψ in the lecture. Make sure to account for the fact that there is no previous step for the first step.
2. path probabilities, i.e., the probability of a hidden state at each step based on the previous hidden state that maximizes this probability (suggested: `List[Dict[str, float]]`). These are called δ in the lecture.

Initializing

The probability of a state starting the sequence at $t = 0$ is just the probability of it emitting the first symbol. There is no transition, so make sure to account for it in the best previous state structure (1). In fact, the first observed symbol is always the special start symbol which can only be emitted by the special hidden start state. You shouldn't have to code this information explicitly. Can you see how it is expressed in your transition and emission tables?

Step by step

In every next step $t > 0$, you need to account for both emission and transition probabilities, i.e., you need to find the most probable previous–current state pair that accounts for the observed emissions. What is the probability of a transition from the previous state to the current one?

For example, let's assume that the roll of 5 at time $t = 6$ in the observed sequence was followed by the roll of 3 at $t = 7$. The probability of the last roll being produced by a fair dice depends on the probability of the hidden states at $t = 6$. Specifically, we want the hidden state at $t = 6$ that maximizes the probability of the roll being produced by a fair dice at $t = 7$. This means that, for the fair

state, you should store the following in the path probabilities data structure:

$$\delta_F(7) = \max_{i \in F,L} (\delta_i(6) a_{iF} b_F(3))$$

Again, $b_F(3)$ is the probability of emitting a dice roll of 3 from a fair dice. a_{iF} is the probability of a transition from state i to the fair state. Crucially, $\delta_i(6)$ is the probability that state i was the hidden state at $t = 6$, which is stored in your path probabilities data structure (2).

In fact, you should know by now to use logarithms instead of pure probabilities. (Why?) Make sure to adjust your formula in the code accordingly.

Now you need to repeat this calculation for all the possible current hidden states [in our example, $\delta_L(7)$] and store the value in the path probabilities data structure.

The best previous hidden state for the fair state is the state i which maximizes the formula above. You should store it in the best previous state data structure (1), together with the equivalents for all other possible current hidden states.

Grand finale

Once you repeat the above calculation for every element in the sequence, i.e., for all ts, you can then figure out the most probable hidden state sequence that produced the observations (including the special start and end observations) by backtracking.

The final hidden state in your sequence should be the special end state emitting the special observed end symbol. What is the best previous state for this end state?

Continue backtracing along the best previous state data structure until you reach the beginning of the hidden sequence. Make sure that your hidden sequence has the same length as the observed one.

Viterbi calculates the most likely single sequence of hidden states. How might you go about calculating the probability of being in a state at a particular time? Is the most likely sequence the same as the sequence of most likely individual states?

Step 2: Evaluation

It's time to check how good your Viterbi prediction of the hidden state sequence is. We are mostly interested in our performance on the loaded state, so the accuracy measure you used so far is not very informative as it is a summary metric over all cases.

In this step, you should calculate the **precision**, **recall** and **F-measure** that your algorithm achieves for the loaded state across a data set. The methods to

compute these measures take as input integer representations of the sequence labels, with 0 being the states we are not interested in. In `tick8.py` we split the data into a sample train:dev split, so that you can easily check if your code works. You will need to replace this code with 10-fold cross-validation in the method `cross_validation_sequence_labeling(data)` to get a tick.

Precision

Precision is the fraction of the states which were classified as the interesting state (loaded in this case) that are really that state.

$$precision(L) = \frac{\text{number of correctly predicted L}}{\text{number of predicted L}}$$

Recall

Recall is the fraction of the interesting states that were correctly predicted as such.

$$recall(L) = \frac{\text{number of correctly predicted L}}{\text{true number of L}}$$

F-measure

F-measure is a combination of precision and recall. There are different forms of F-measure which weight the precision and recall differently, but the one we will be using is the **F1-measure** which weights them equally. This is calculated as the harmonic mean of precision and recall:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

How would you calculate each of the above scores for a single sequence? When calculating the scores for a dataset, remember that sequences can have different lengths.