Task 8: Viterbi algorithm

This is the first part of Tick 6.

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 a 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. using generics.

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 states (suggested: `List<Map<DiceType, DiceType>>`). 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 probabilities of a hidden state at each step assuming that the previous hidden state was the one which maximises this probability (suggested: `List<Map<DiceType, Double>>`). These are called δ in the lecture.

Initalizing

For the first step the probabilities of each hidden state are just the inital probabilities from the HMM multiplied by the probability of producing the first observed dice roll in that state. For example, if the first element of the observed sequence of dice rolls is 5, the viterbi probability $\delta_F(1)$, corresponding to the probability of being in state F at time $t = 1$, is:

$$\delta_F(1) = a_{0F} b_F(5)$$

where a_{0F} is the probability of starting with the fair dice, and $b_F(5)$ is the probability of emitting a dice roll of 5 from the fair state.

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.

You should now store the scores for each state in the data structure 2 which stores path probabilities.

What happens in step $t = 2$?

In this step and every next one you need to additionally account for transition probability from the previous state to the current one – you need to find the most probable previous-current state pair that accounts for the observed emissions.

For example, let's assume that the roll of 5 at $t = 1$ was followed by the roll of 3 at $t = 2$. The probability of this roll being produced by a fair dice depends on what the previous state s was. Let's assume that the previous score was such that the probability is maximised. Then for the fair state you should store the following in the path probabilities data structure:

$$\delta_F(2) = \max_{i \in F,L} (\delta_i(1) a_{iF} b_F(3))$$

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, $\delta_i(1)$ is the probability that state i was the hidden state at $t = 1$, which is stored in your path probabilities data structure (2).

Now you need to repeat this calculation for all the possible current states and store the value in the path probabilities data structure.

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

Grand finale

Once you repeat the above calculation for every sequence element, i.e for all ts, you can then figure out the most probable sequence.

1. At time T+1, you multiply the probability of reaching the final state from each possible state, with the corresponding delta probabilities.

$$P(O, X|\mu) = \max_{i \in F,L} (\delta_i(T) a_{if})$$

At that stage, you have found the probability you were looking for, and with it, the last state of your hidden state sequence.

2. We now reconstruct the state sequence by backtracing. What is the best previous state for the best last step you just found? Hint: check the data structure 1 at the last t. This is the penultimate state of your hidden state sequence.

3. Continue backtracing along the best previous state data structure until you reach the beginning of the hidden sequence. Remember that there is no previous state at $t = 0$!

Make sure that your hidden sequence has the same length as the observed one (modulo start and end state).

Viterbi calculates the most likely single sequence. How might you go about calculating the probability of being in a state at a particular time point? 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 sequence is. We are mostly interested in the loaded state so the accuracy measure you used so far is not very informative.

In this step you should calculate **precision**, **recall** and **F-measure** that your algorithm achieves for the loaded state across a data set. In `Exercise8Tester` we split the data into a sample train:dev:test split, so that you can easily check if your code works. You will need to replace this code with 10-fold cross-validation 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.