HMM Algorithms

Saurabh Burewar (B18CSE050)

Viterbi Algorithm

We can write the HMM given in the question in the form of transition matrix, emission matrix and initial probabilities array (to denote the starting point of the HMM).

So, the given HMM will be -

```
Tm = np.array([[0.6, 0.4],
               [0.3, 0.7]])

Em = np.array([[0.7, 0.3],
               [0.4, 0.6]])

initialP = [1, 0]
```

Observation sequence -

```
y = [0, 1, 1, 0]
```

The viterbi algorithm takes all of these as input and gives the most likely state sequence as the output. The algorithm is implemented in python using numpy to implement operations like argmax.

Output -

```
Most likely state sequence:  [0 1 1 1]
```

We can try different observation sequences to get the corresponding most likely state sequence.

Evaluation problem

To calculate the probability of observing a given sequence we can use the forward or the backward algorithms. The forward algorithm first initializes the alpha vectors and then performs recursion. The probability can be given as -

$$P(o_1 o_2 .... o_k) = \Sigma_i \alpha_k(i).$$

Similarly, the backward algorithm initializes the beta vectors and then performs recursion. The probability can be given as -

$$P(o_1 o_2 .... o_k) = \Sigma_i \beta_1(i) Em_i(o_1) \pi_i$$

We use the same example used in the viterbi algorithm above here, and get the probability of the observation sequence using both forward and backward algorithms. It is found that both algorithms give very different results i.e. results are inconsistent for both algorithms.

Output for observation sequence 0110 -

Probability calculated using alpha vectors - 0.303072
Probability calculated using beta vectors - 0.07131599999999999

HMM training

HMM training is implemented using the forward-backward algorithm.

Q3 - Similar to before, we write the transition and emission matrix of the toy example HMM given and run the training algorithm on the given observation sequence for 100 iterations. We get the following results -

Transition matrix	Emission matrix
[[0. 1.]	[[0.667 0.333]
[1. 0.]]	[0. 1.]]

The result is not the actual HMM, so we increase the number of iterations and run again. But, even for 100000 iterations, the results stay the same.

Q4 - Now, we run HMM training for the HMM given in this question for 100 iterations and get the following results -

Transition matrix	Emission matrix
[[0. 0.5 0.5]	[[0.5 0.5]
[1. 0. 0.]	[0.5 0.5]
[1. 0. 0.]]	[0.5 0.5]]

This is not the actual HMM, so we increase the number of iterations. Again, we observe that upto 100000 iterations, there is no change in results.

References

- https://cse.buffalo.edu/~jcorso/t/CSE555/files/lecture_hmm.pdf
- http://www.adeveloperdiary.com/data-science/machine-learning/derivation-and-implementation-of-baum-welch-algorithm-for-hidden-markov-model/