Homework4 Hidden Markov Model

Yuan Bowei 1001916

1.Parameters in HMM

HMM consists of two parts, the ***transition parameters*** and ***emission parameters***.

Initially the goal of this model is to get a mapping function:

$$f(x_1, x_2, \ldots, x_n) = argmax_{y_1, y_2, \ldots, y_n} P(x_1, x_2, \ldots, x_n, y_1, \ldots, y_n)$$

Which means that we are trying to find out the tags for a sequence of words which makes the joint probobality $P(x_1, x_2, \ldots, x_n, y_1, \ldots. y_n)$ to be maximum. This joint probability is mathematically equivalent to conditional probability (1): $P(y_1, y_2, \ldots, y_n)P(x_1, x_2, x_3, \ldots, x_n|y_1, y_2, \ldots, y_n)$

1.1 Transition Parameters

The first term is the probability (1) to get one particular sequence of tags, it can be extended into:

$P(y_1, y_2, \ldots, y_n)$

$= P(y_0)P(y_1|y_0)P(y_2|y_1, y_0)\ldots P(y_{n+1}|y_0, y_1, \ldots, y_n)$

Specially, y_0 and y_n respectively represent $START$ and $STOP$ tags. As a sentence must 'start', so $P(y_0)$ is regarded as 1.

Here we assume the probability of appearance of each tag is associated with the appearance of all other tags in front of it, it is intuitively correct but a result is that there would be an extremely long dependency list. Therefore, we can make a very strong independence assumption, the incidence of each tag is only dependent on the incidence of the tag justly in front of it. Therefore, a more concise expression is:

$\approx P(y_1|y_0)P(y_2|y_1)P(y_3|y_2)\ldots P(y_n + 1|y_n)$

$= \prod_{i=1}^{n+1} P(y_i|y_{i-1})$

This parameter, or this product of parameters, named ***transition parameters*** which describes the probability of transitioning from one tag to another.

1.2 Emission Parameters

Now let's look at the second term in (1), the conditional probability, which could be extended to:

$$\prod_{i=1}^{n} P(x_i|x_1, x_2, \ldots, x_{i-1}, y_1)$$

And here we can make another strong assumption that the probability of the incidence of each word, is ***only*** dependent of its corresponding tag. So finally the second term in (1) can be presented as:

$$\approx \prod_{i=1}^{n} P(x_i|y_i)$$

This is named ***emission parameter*** which defines how one word is 'emitted' from its corresponding tag.

1.3 How these parameters are exactly estimated

Under the two strong assumptioins, expression(1) can be presented in following concise form:

$$P(x_1, x_2, \ldots, x_n, y_0, y_1, \ldots, y_{n+1}) \approx \prod_{i=1}^{n} P(y_i|y_{i-1}) \prod_{i=1}^{n} P(x_i|y_i)$$

Now we need to compute the estimator for the $P(y_i|y_{i-1})$'s and $P(x_i|y_i)$'s.

First step, the product form of this equation would cause problems when taking the derivative, instead we can take the logarithm of it to transfer them into su forms.

$$logP(x_1, x_2, \ldots, x_n, y_0, y_1, .., y_{n+1}) = \sum_{i=1}^{n} logP(y_i|y_{i-1}) + \sum_{i=1}^{n} logP(x_i|y_i)$$

Another two facts that we can use to facilitate our further calculation are that:

$$\forall TAG \in D_n, \sum_{i=1}^{n} P(next = y_i|current = TAG) = 1$$

$$\forall TAG \in D_n, \sum_{i=1}^{n} P(word = x_i|tag = TAG) = 1$$

Second step, optimization. In order to get the maximized log likelihood, we take derivative according to $P(y_i|y_{i-1})$ and $P(x_i|y_i)$ and set them to zero for the optimal value. We get following two expressions (2) and (3), where W_i stands for $word_i$, and T_i stands for Tag_i :

(2): $\frac{\partial logP(x_1,x_2,...,x_n,y_0,y_1,...,y_{n+1})}{\partial P(next=T_i|current=T_j)} = 0$

(3): $\frac{\partial logP(x_1,x_2,...,x_n,y_0,y_1,...,y_{n+1})}{\partial P(word=W_i|tag=T_i)} = 0$

With these two equations and two conditions given above, we can reach the estimation of the ***transition probability*** and ***emission probability***:

*Transition Probability: $T(a|b) = \frac{count(current=b,next=a)}{count(b)}$

*Emission Probability: $E(x|y) = \frac{count(label=y,word=x)}{count(label=y)}$

1.4 In this example:

With the estimators we got from 1.3, the ***transition matrix*** and ***emission matrix*** are following:

- **Transition matrix:**

Transition	current=START	current=X	current=Y	current=Z
next=X	$\frac{1}{2}$	$\frac{2}{7}$	$\frac{1}{4}$	$\frac{1}{7}$
next=Y	0	0	0	$\frac{4}{7}$
next=Z	$\frac{1}{2}$	$\frac{3}{7}$	0	$\frac{1}{7}$
next=END	0	$\frac{2}{7}$	$\frac{3}{4}$	$\frac{1}{7}$

- **Emission matrix:**

Emission	X	Y	Z
a	$\frac{3}{7}$	$\frac{1}{2}$	$\frac{1}{7}$
b	$\frac{2}{7}$	0	$\frac{4}{7}$
c	$\frac{2}{7}$	$\frac{1}{4}$	$\frac{1}{7}$
d	0	$\frac{1}{4}$	$\frac{1}{7}$

2. Viterbi Algorithm

- **Step 1.** At the first position, we calculate the posibility for each label *y*, where $P = T(next = y|current = START)E(word = a|label = y)$

X	Y	Z
$\frac{1}{2} * \frac{3}{7} = \frac{3}{14}$	0	$\frac{1}{2} * \frac{1}{7} = \frac{1}{14}$

- **Step 2.** At the second position, we compute:

  1.the maximum probabilities of each labels at this position, by multiplying the trasition probability from that label and the corresponding emission probability for different labels and take the maximum one among them

  2.by the way memorise from wich possible label at the previous position we generated this maximum probability, in other words, the parent node (previous label) of this label.

X	Y	Z
$\frac{1}{14} * \frac{1}{7} * \frac{1}{7} = \frac{1}{686}$ from Z	$\frac{1}{14} * \frac{4}{7} * \frac{1}{4} = \frac{1}{98}$ from Z	$\frac{3}{14} * \frac{3}{7} * \frac{1}{7} = \frac{9}{686}$ from X

- **Step 3.**

Based on the result of first two steps, now we have to compute the probability of each label when the setence end. Just multiply the probability of the label at the final position with the label's probability to occur at the end, and take the maximum one.

End: $\frac{1}{98} * \frac{3}{4} = \frac{3}{392}$ from Y.

So we have got the longest path, and therefore the most probable label sequence is:

START -> Z -> Y -> END

3. 2-Order Hidden Markov Model

Generating phase

To apply this 2-order HMM, at each position, the tag is dependent on two tags in front of it. There for at position p_i we can maintain a $T * T$ matrix to record the the highest score of tag combinations at (p_{i-1}, p_i). As tag in p_i is dependent on tag pairs at (p_{i-2}, p_{i-1}), therefore to compute the highest score for each pair of words in sequence (p_{i-1}, p_i), we have to enumerate all T possibilities in p_{i-2}, and take the maximum one of them.

So the procedure is:

1. enumerate all positions p_i. $O(n)$
2. at each position, enumerate all possible tags y_i at position p_i. $O(T)$
3. for each possible tag y_i at position p_i, we enumerate all possible tags y_{i-1} at position p_{i-1}. $O(T)$
4. for each tag pair (y_{i-1}, y_i) at their respective position (p_{i-1}, p_i), we enumerate all possible tags at position p_{i-2} to compute a score of transitioning to y_i from (y_{i-2}, y_{i-1}) in current sentence, and take the maximum one as the score of y_{i-1}, y_i. $O(T)$
5. The score function is:
   $Score(y_{i-1}, y_i) = \max_{y_{i-2}^{(m)}, m \in (1,2,...,T)} Score(y_{i-2}^{(m)}, y_{i-1}) * P(y_i | y_{i-1}, y_{i-2}^{(m)}) * P(x_i | y_i)$. $O(1)$

Therefore in this phase, the time complexity is O(nT^3) and space complexity is O(nT^2).

Decoding phase

Finally after our generating phase, at the $n + 1$ position where the sentence's END. And we can get a pair of labels for position pair (p_{n-1}, p_n) from where we reached the end of sentence.

Then with this pair of label in (p_{n-1}, p_n) and the $O(T^2)$ matrix we stored in p_n we can generate the label pair for position pair (p_{n-2}, p_{n-1}). So on so forth, till back to the beginning, we can decode the entire sentence. This process is O(n).

4.

Introduction

Firstly we define three probabilities in this algorithm:

1. Transition probability $\alpha(Z_i, Z_{i+1})$, which is basically the same as the transition probatility in the HMM we have discussed in class, standing for the probability of transitioning to state Z_{i+1} when we know Z_i, that is:

$$\alpha(Z_i = a, Z_{i+1} = b) = P(Z_{i+1} = b|Z_i = a)$$

2. First Emit Probability $\beta(Z_i, X_i)$, denoting the probability of emitting a certain tag X_i from the state Z_i, that is:

$$\beta(Z_i = a, X_i = b) = P(X_i = b|Z_i = a)$$

3. Second Emit Probability $\gamma(Z_i, X_i, Y_i)$, standing for the probability that we generate a certain Y_i with given Z_i and X_i, in a concise form:

$$\gamma(Z_i = a, X_i = b, Y_i = c) = P(Y_i = c|Z_i = a, X_i = b)$$

In this problem, analogously to HMM model we have discussed in class, given the observation paris $(X_1, Y_1, X_2, Y_2, \ldots, X_n, Y_n)$, we want to generate optimal state sequence $Z_1, Z_2, \ldots, Z_{n+1}$ such that (suppose $Z_0 = START$ and $Z_{n+1} = STOP$):

$$Z_0^*, Z_1^*, \ldots, Z_{n+1}^* = argmax_{Z_0, Z_1, \ldots, Z_{n+1}} P(Z_0, Y_1, X_1, Z_1, Y_2, X_2, Z_2, \ldots, Y_n, X_n, Z_n, Z_{n+1}; \theta)$$

$$argmax_{Z_0, Z_1, \ldots, Z_{n+1}} \prod_{i=1}^{n+1} \alpha(Z_{i-1}, Z_i) \prod_{i=1}^{n} \beta(Z_i, X_i) \prod_{i=1}^{n} \gamma(Z_i, X_i, Y_i)$$

$$= argmax_{Z_0, Z_1, \ldots, Z_{n+1}} \prod_{i=1}^{n+1} P(Z_i|Z_{i-1}; \theta) \prod_{i=1}^{n} P(X_i|Z_i; \theta) \prod_{i=1}^{n} P(Y_i|X_i, Z_i; \theta)$$

So we can apply the Dynamic Programming Algorithm to this problem, at each position i in one sentence, for each possible state $Z_i^{(j)}, j \in (1, 2, .., T)$ at each position, we define a forward score:

$$Score(Z_i^{(j)}) = \max_{Z_{i-1}^{(m)}, m \in (1,2,\ldots,T)} P(Z_i^{(j)}|X_i, Y_i) = \max_{Z_{i-1}^{(m)}, m \in (1,2,\ldots,T)} Score(Z_i^{(m)}) P(Z_i^{(j)}|Z_{i-1}^{(m)}) P(X_i|Z_i^{(j)}) P(Y_i|X_i, Z_i^{(j)})$$

Besides the general case formula, we also define the Score function at the START and STOP position:

- At START position(the first word of a sentence):
  $Score(Z_i^{(j)}) = P(Z_i^{(j)}|START)P(X_i|Z_i^{(j)})P(Y_i|X_i, Z_i^{(j)})$
- At STOP position(after the last the word of a sentence):
  $Score(STOP) = \max_{m\in(1,2,...,T)} Score(Z_i^{(m)})P(STOP|Z_i^{(m)})$

Thus our forward procedure is:

1. Enumerate all positions in one sentence. $O(n)$
2. At each position i, we enumerate all possible tags at this position. $O(T)$
3. For each possible tag Z_i at this position, we enumerate all possible tags at the position in front of the current position, Z_{i-1}. For each Z_{i-1}, we calculate the score with the Score function introduced above, finally we take the highest score among all Z_{i-1} and save it as the score of tag Z_i. $O(T)$
4. In conclusion, the forward procedure takes $O(nT^2)$ time and $O(nT)$ space

The decoding procedure has 2 versions of solutions:

1. The simple solution:
   - In the step3 of forward procedure mentioned above, when we save the maximum score for tag $Z_i^{(j)}$, we also save from which one of $Z_{i-1}^{(m)}$ we get this maximum score.
   - Then if we can go reversely from the STOP to the START of a sentence, when we know Z_i, we can definitely find Z_{i-1} with the information we have saved, and hence we could recover the states sequence. The time complexity would be $O(n)$. But the space complexity would be doubled.

2. The complicated solution：
   - with the score at the position i, we can compute the score at the position by the formula: $Score(i-1) = \frac{Score(i)}{P(Z_i|Z_{i-1})P(X_i|Z_i)P(Y_i|Z_i,X_i)}$, and with this score we can determine the optimal tag at the prior position.
   - Therefore in the STOP position $n+1$, we can determine the optimal tag at position n. So when we move backword from STOP to START, the sentence is decoded. The time complexity would be $O(nT)$, the space complexity remains the same.