In this reading we recap hidden Markov models. We demonstrate how to compute probabilities of tag sequences and re-estimate parameters with Baum-Welch algorithm. You need to understand this to complete the quiz of this week.

Hidden Markov Model

Consider the following rhyme from "Jabberwocky" by Lewis Caroll:

'Twas brillig, and the slithy toves

Did gyre and gimble in the wabe;

All mimsy were the borogoves,

And the mome raths outgrabe.

Let's take the last phrase, ***"the mome raths outgrabe"***, as an example. Let's build a hidden Markov model for predicting part of speech tags in this sentence. For simplicity, suppose that we have just three tags: N (noun), V (verb) and D (determiner). We need to specify initial probabilities of these tags and transition probabilities from one tag to another. Surely, these probabilities may be estimated using some annotated corpus. But let's suppose for now that all tags and all transitions are equiprobable:

$$\mathrm{p(N \mid start) = p(O \mid start) = p(D \mid start) = 1/3} p(N|start)=p(O|start)=p(D|start)=1/3$$

$$\mathrm{p(N \mid N) = p(V \mid N) = p(D \mid N) = 1/3} p(N|N)=p(V|N)=p(D|N)=1/3$$

and so on.

Dealing with HMMs, we also need to specify output probabilities of words given the tag. For simplicity, we consider the following outcomes:

N: mome | raths | outgrabe

V: raths

D: the | a

Let all these outcomes be also equiprobable, i. e,

p(mome | N) = p(raths | N) = p(outgrabe | N) = 1/3$p(mome|N)=p(raths|N)=p(outgrabe|N)=1/3$

p(raths | V) = 1$p(raths|V)=1$

p(the | D) = p (a | D) = 1/2$p(the|D)=p(a|D)=1/2$

Note that some words like "raths" may be generated from different tags (otherwise, the tagging is trivial). Note also, that our test phrase does not have to contain all of these words. For example, "a" is absent in the test sentence.

Probabilities of tag sequences

Given this toy model, let's find the probabilities of possible tag sequences for the phrase ***"the mome raths outgrabe"***. In other words, these are the conditional probabilities: p(XXXX | phrase)$p(XXXX|phrase)$, where each tag X is either N, or V, or D.

First question for you: how many different tag sequences exist? Second question: which of them could happen in our case with the transition and output probabilities defined above?

Answers: there are 3^4 = 813$4=81$ sequences, but only **two of them** are possible in our case. "the" can be generated only from D, "mome" and "outgrabe" can be generated only from N, and "raths" can be generated wither from N or V. So we can have either **DNNN** or **DNVN.**

So we have just seen, that probabilities of 79 tag sequences are equal to 0, and we need to compute these two: p(DNNN | phrase) and p(DNVN | phrase). According to the HMM model, the joint probabilities are:

p(DNVN, phrase) = p(D | start)\, p(the | D)\, p(N | D)\, p(mome | N)\, p(V | N)\, p(raths |V)\, p(N |V) \, p(outgrabe | N)$p(DNVN,phrase)=p(D|start)p(the|D)p(N|D)p(mome|N)p(V|N)p(raths|V)p(N|V)p(outgrabe|N)$

$p(DNNN,phrase)=p(D|start)p(the|D)p(N|D)p(mome|N)\boldsymbol{p(N|N)p(raths|N)p(N|N)}p(outgrabe|N)$

At this point, you could just use the given values above to compute these expressions. After that, you would need to normalize them in such a way that they sum into 1, since it should be a **distribution,** and all other 79 values are known to be 0.

But let's reduce our calculations and look closer into the two formulas above. The only term that differs there is this one: p(raths |N) = 1/3$p(raths|N)$=1/3 while p(raths|V) = 1$p(raths|V)$=1. So, for some multiplier x,

p(DNNN | phrase) = x \cdot 1/3$p(DNNN|phrase)$=x·1/3

p(DNVN | phrase) = x \cdot 1$p(DNVN|phrase)$=x·1

Since these probabilities must sum into one, we find that they are equal to 1/41/4 and 3/43/4, and we are done.

Baum-Welch probability re-estimation

Now let's see how to re-estimate transition or output probabilities in our model, given the same sentence ***"the mome raths outgrabe"***. It means performing one iteration of Baum-Welch algorithm (= EM-algorithm). Actually, we have just done E-step by computing probabilities of tag sequences. Now let's see how the M-step works. For example, let's re-estimate the transition probability p(V|N)$p(V|N)$, that used to be one-third.

We need to find ***the expectation of the number of transitions from N to V*** and divide it to ***the expectation of the number of transitions from N to any tag***. The expectation is taken with respect to the probabilities of tag sequences (computed above).

Remember, there are only two possible sequences. We have 0 transitions from N to V in the DNNN sequence and exactly one such transition in the DNVN sequence. So the expectation for (N -> V) transitions is 1/4 \cdot 0 + 3/4 \cdot 1 = 3/41/4·0+3/4·1=3/4.

Similarly, we have two transitions from N to some tag in DNNN and 1 such transition in DNVN. The expectation for (N -> ?) transitions is 1/4 \cdot 2 + 3/4 \cdot 1 = 5/41/4·2+3/4·1=5/4.

Thus, the new estimation for the transition probability p(V | N)$p(V|N)$ is 3/53/5. This is exactly the probability, that would be assigned to the corresponding HMM parameter if we were training it with Baum-Welch on this one phrase.

In real examples though, the flow of computations in Baum-Welch is a bit different. First, usually you have too many tag sequences. So it's impossible to compute there probabilities and take the expectation with respect to them. E.g. imagine you have 20 possible tags and a sequence of length 10. Then you would have 20^{10}2010 tag sequences! So instead of the probabilities of the whole sequences, you would compute the probabilities of two sequential tags (see the slides). These probabilities would be enough to perform M-step. Second, it is hard to compute even these tag pair

probabilities, so here a so called Forward-Backward algorithm is used. It's a dynamic programming algorithm that allows efficient computations.