

Natural Language Processing, Homework 4

Prof. Liang Huang

Due at 11:59pm on Mon Nov 10 on BB

This homework is about unsupervised learning (using the EM algorithm) of the alignment between English and Japanese phonemes in English-Katakana pairs. Recall from HW2/3:

```
AE K T ER      ;; English phoneme sequence for 'actor'
A K U T A A    ;; Same word, loaned into Japanese
1 2 2 3 4 4    ;; e.g., Japanese T maps to the 3rd English sound
```

But in this homework, you need to learn the alignments yourself. In fact, the alignments we gave you in HW3 were also learned by EM, and thus is *not* 100% correct. You will reuse the following files from HW3:

```
epron-jpron.data  a database of aligned English/Japanese phoneme sequence pairs
eword.wfsa        a unigram WFSA of English word sequences
eword-epron.wfst  a WFST from English words to English phoneme sequences
jprons.txt        a short list of Japanese Katakana sounds to decode
```

1 Complete Data, Incomplete Model (5 pts)

If we're given the following manually aligned pairs (English "boat" and "test"):

```
B OW T          T EH S T
B 0 0 T 0       T E S U T 0
1 2 2 3 3       1 2 3 3 4 4
```

What is the maximum likelihood estimate for $p(T | T)$ and $p(T 0 | T)$?

2 Complete Model, Incomplete Data (10 pts)

Given the following conditional probabilities,

```
p(B | B) = 1
p(0 0 | OW) = 0.8   p(0 | OW) = 0.2
p(T | T) = 0.3      p(T 0 | T) = 0.5   p(0 | T) = 0.1   p(0 T 0 | T) = 0.1
```

Enumerate (by hand) all legal alignments for the "boat" example above, and compute the probability and normalized probability of each alignment. Which one is the Viterbi alignment?

3 EM, Implementation I: Enumerating All Alignments (65 pts)

Now implement the EM algorithm:

1. initialize the conditional probabilities to uniform

2. repeat:
3. E-step:
4. for each English-Katakana pair:
5. enumerate all legal alignments for that E-K pair, along with their respective probabilities
6. renormalize the probabilities to get a distribution over these alignments
7. collect the fractional counts for this E-K pair
8. M-step: count-and-divide based on the collected fractional counts from all pairs to get new prob. table
9. until reached maximum iteration or converged

You should proceed with the following steps:

1. (15 pts) to start with, you should work on the simplest scenario (“wine” example): W A Y N / W A I N.

Print the following information in each iteration: 1) the new viterbi alignment (if changed), 2) the corpus probability, 3) the number of nonzero entries in the prob. table (≥ 0.01), and 4) the new prob. table. **Note:** the corpus probability should increase in each iteration (we have the proof).

For example, you should run `echo -e "W A I N\nW A I N\n" | ./em.py 5` and get the following (note that the data should be read in from standard input and 5 is the number of iterations):

```
iteration 0 -----
corpus prob= 0.00411522633745  changed = 0
AI|->  A: 0.33 A I: 0.33      I: 0.33
W|->   W: 0.67 W A: 0.33
N|->   N: 0.67 I N: 0.33
nonzero = 7
```

```
iteration 1 -----
W      AI      N
W      A      I      N
1      2      3      3  -->
1      2      2      3
```

```
corpus prob= 0.296296296296  changed = 1
AI|->  A I: 0.50      A: 0.25 I: 0.25
W|->   W: 0.75 W A: 0.25
N|->   N: 0.75 I N: 0.25
nonzero = 7
```

...

```
iteration 4 -----
corpus prob= 0.912659654395  changed = 0
AI|->  A I: 1.00
W|->   W: 1.00
N|->   N: 1.00
nonzero = 3
```

For your debugging convenience you can also print all possible alignments and their unnormalized and normalized probabilities in each iteration. Include the result of 5 iterations in your report.

- (5 pts) If you add `N AY N / N A I N` to it you should see it converge faster. Show the results.
- (5 pts) Now that you have passed “wine”, try the “test” example above (`T EH S T / T E S U T O`) and you’ll see that EM converges to something wrong. Show me the result of 5 iterations.

Now come up with a minimal second example so that when combined with the “test” example, EM will learn the correct alignments. This second pair has to be a real English word, and should be chosen from `epron-jpron.data`.

- (8 pts) Now come up with an example dataset where EM does not learn anything and is still ambivalent at the end (i.e., the final probability table is (almost) the same as the initial one). In the lecture we gave `B IY / B I I` as one such example, but can you come up with something bigger, i.e., longer sequences and multiple, interdependent pairs?
- (15 pts) If you have passed all previous questions, it’s now time to work on the whole dataset with the following command-line:

```
cat eprom-jpron.data | ./em.py 15 eprom-jpron.viterbi >eprom-jpron.probs 2>eprom-jpron.logs
```

(Ignore the alignment lines – your job is to learn them yourself!)

The conditional probability table (`.probs` file) should follow the same format as in HW3 (ignore < 0.01 entries), and the viterbi alignments file should follow the same format as `eprom-jpron.data` so that we can compare (see below). The logs file contains the logging info for all iterations (see above), but don’t print the prob. table in each iteration! and you only need to print the first five (5) changed viterbi alignments in each iteration, if any.

Note: Do not reestimate probs from the final Viterbi alignments – you’ll get HW3 probs that way.

Hint: The corpus probs are likely to underflow (beyond the range of Python `float`). Try fix it.

- (5 pts) Calculate the number of different viterbi alignments between your result and `eprom-jpron.data`. It should be less than 10 (out of 2684 examples).
- (7 pts) Convert your `eprom-jpron.probs` to `eprom-jpron.wfst` as you did in HW3. Decode `jprons.txt` from HW3 using `eword.wfsa`, `eword-eprom.wfst`, and your EM-learned `eprom-jpron.wfst`. Compare with your results from HW3: did you see any difference?

4 EM, Implementation II: Forward-Backward (DP) (20 pts)

The enumeration of all alignments is of course not the best idea, since in the worst case, there are exponentially many such alignments per example.

- (5 pts) BTW how many possible alignments are there for a pair of n English phonemes and m Japanese phonemes ($n \leq m$)? What pair(s) in `eprom-jpron.data` give the highest number of alignments?
- (25 pts) Implement the forward-backward algorithm (replacing lines 5–7 in the pseudocode above) so that you can do dynamic programming to collect fractional counts without enumerating all alignments. Make sure the two approaches match on their results.

Include a detailed pseudocode of your implementation. What is the complexity of this forward-backward algorithm for a (n, m) pair? Try optimize your program (`em2.py`). How does it compared to the enumeration approach in terms of speed? Part of the grade will depend on efficiency. (4 pts pseudocode, 3 pts complexity, 13 pts correctness of implementation, 5 pts efficiency).

Include all Python programs.