UNIVERSITÄT DES SAARLANDES

Statistical Natural Language Processing

Summer Semester 2020
Prof. Dr. Dietrich Klakow

Exercise Sheet IX

Submission Deadline: July 10th, 23:59

EM for Word Sense Disambiguation

In this exercise, you will implement expectation-maximization[1] for word sense disambiguation. You can find the pseudocode on slide 44 of chapter 7.
You will be provided with starter code to help you with the implementation, but you can also write your solution from scratch if you want to. If you are using the starter code, you will have to install the NumPy package on your machine.

The dataset

NLTK has a corpus reader for the Senseval 2 dataset. This data set contains data for four ambiguous words: hard, line, serve and interest. You can read more here: http://www.nltk.org/howto/corpus.html (search for "senseval"). The provided code loads the dataset for you and converts each sample into a sample object. The sample class has two attributes: context and label. Label is the ground truth sense of the ambiguous word. As EM is an unsupervised method, we will only use the labels for final evaluation. Context is the left and right context of the ambiguous word as word IDs: it is a list of integers, which you can use later to index matrices.

```
instances = senseval.instances(hard_f)

# All training samples as a list.
samples = [sample(inst) for inst in instances]

# Convert contexts to indices so they can be used for indexing.
for sample in samples:
```

[1] See `https://machinelearningmastery.com/expectation-maximization-em-algorithm/`

```
sample.context_to_index(word_to_id)
```

Now, "samples" contains all the sample objects for the ambiguous word "hard".

Implementation

There are three matrices provided for you, which you will have to use for the implementation: *priors*, *probs* and C.

- *priors*: It is a vector of length K, where K is the number of clusters. Each value corresponds to a cluster prior $p(s_k)$. It's initialized randomly.
- *probs*: It is a V x K sized matrix, where V is the size of the vocabulary. Each column is a conditional probability distribution over the words. $probs[i, k]$ is $p(v_i|s_k)$.
- C: contains the counts of the words in a given context. The size of the matrix is number of samples x vocabulary size. $C[i, j]$ is $C(v_j$ in $c_i)$, the count of word j in context i.

To get a better feel of how to use these matrices read the log_ likelihood_np(samples, probs, priors): function.
Get the IDs of the words. We will use this to index the rows of the probs matrix:

```
context_index = sample.context
words_given_sense = probs[context.index, :]
```

This is a matrix: the number of lines is the number of words in the context, the number of columns is number of senses. We multiply this column wise to get the probability of a context given a sense $p(c_i|s_k)$:

```
context_given_sense = np.prod(words_given_sense, axis=0)
```

This is a vector where each value is a class conditional probability (size is K).

Exercises

- Briefly describe what EM is and when using it is appropriate? Give an example use case (1 point).
- Write the code for the expectation step (3 points).

- Write the code for the maximization step. In order to check your implementation, make sure that the log likelihood increases over the iterations (4 points).
- The code will print the ordered frequency of the labels within each cluster. Each cluster corresponds to one of the senses of "hard". How does the algorithm perform? Briefly explain. (1 point)
- Does EM find the global optimum? Explain why/why not? (1 point)

Submission Instructions

The following instructions are mandatory. Please read them carefully. If you do not follow these instructions, the tutors can decide not to correct your exercise solutions.

- You have to submit the solutions of this exercise sheet as a team of 2 students.
- If you submit source code along with your assignment, please use Python unless otherwise agreed upon with your tutor.
- NLTK modules are not allowed, and not necessary, for the assignments unless otherwise specified.
- Make a single `ZIP` archive file of your solution with the following structure
  - A `source_code` directory that contains your well-documented source code and a `README` file with instructions to run the code and reproduce the results.
  - A `PDF` report with your solutions, figures, and discussions on the questions that you would like to include. You may also upload scans or photos of high quality.
  - A `README` file with group member names, matriculation numbers and emails.
- Rename your `ZIP` submission file in the format

  `exercise02_id#1_id#2.zip`

  where `id#n` is the matriculation number of every member in the team.

- Your exercise solution must be uploaded by only one of your team members under *Assignments* in the *General* channel on Microsoft Teams.
- If you have any problems with the submission, contact your tutor before the deadline.