SNLP 2016
Exercise 8

Submission date: 24.06.2016, 23:59

EM for words sense disambiguation

In this exercise, you will implement expectation-maximization for word sense disambiguation. You can find the pseudocode on slide 43 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 ambigous words: hard, line, serve and interest. You can read more about how to import the data and use the reader here . Look for the dataset called senseval.
The provided code loads the dataset for you and converts each sample into 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:
        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.

```
context_index=sample.context
```

Get the IDs of the words. We will use this to index the rows of the *probs* matrix.

```
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 columnwise 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 probabilty (size is K).

Exercise

- (3 points) Write the code for the expectation step. The easiest way to do this is to create a matrix H where $H[i,k]$ is $h_{i,k}$ from the slides.
- (4 points) Write the code for the maximization step, in which you should update the values of *probs* and *priors*. In order to check your implementation, make sure that the log likelihood increases over the iterations
- (1 point) The code will print the ordered frequency of the true labels within each cluster. How does the algorithm perform?
- (1 point) Does EM find the global optimum?
- (1 point) In this case, the number of clusters was set to the number of senses provided in the corpus. In a real unsupervised scenario, this is not given. How would you decide on the number of clusters?

Bonus (2 points)

- Explain how the information in matrix C could be used to train a classifier (e.g. Support vector machine or KNN). What would be your features and targets?

1 Submission Instructions: Read carefully

- You can form groups of maximum 3 people.
- Submit only 1 archive file in the ZIP format with name containing the MN of all the team members, e.g.:

  `Exercise_01_MatriculationNumber1_MatriculationNumber2_MatriculationNumber3.zip`

- Provide in the archive:
  - your code, accompanied with sufficient comments,
  - a PDF report with answers, solutions, plots and brief instructions on executing your code,
  - a README file with the group member names, matriculation numbers and emails,
  - Data necessary to reproduce your results [1]
- The subject of your submission mail must contain the string [SNLP] (including the braces) and explicitly denoting that it is an exercise submission, e.g:

  `[SNLP] Exercise Submission 08`

- Depending on your tutorial group, send your assignment to the corresponding tutor:

[1] If you feel that these files are beyond reasonable size for an email submission and also reasonably convenient, please provide a means for us to access them online

- Sedigheh Eslami: *eslami@mpi-inf.mpg.de*
- Naszdi Kata: *b.naszadi@gmail.com*
- Stephanie Lund: *stflund@gmail.com*

2 General Information

- In your mails to us regarding the tutorial please add the tag [SNLP] in the subject accompanied by an appropriate subject briefly describing the contents.
- Feel free to use any programming language of your liking. However we strongly advise in favor of Python, due to the abundance of available tools (also note that Python3 comes with an excellent native support of UTF8 strings).
- Avoid using libraries that solve what we ask you to do (unless otherwise noted).
- Avoid building complex systems. The exercises are simple enough.
- Do not include any executable files in your submission, as this may cause the e-mail server to reject it.
- **In case of copying, all the participants (including the original solution) will get 0 points for the whole assignment. Note: it is rather easy to identify a copied solution. Plagiarism is also not tolerated.**
- **Missing the deadline even for a few minutes, will result in 50% point reduction. Submission past the next tutorial, is not corrected, as the solutions will already be discussed.**
- **Please submit in your solutions necessary to support your claims. Failure to do so, might results in reduction of points in the relevant questions.**
- Each assignment has 10 points and perhaps some bonus points (usually 2 or 3). In order to qualify for the exams, you need to have 2/3 of the total points. For example, in case there are 12 assignments, you need to collect at least 80 out of the 120 points to be eligible for the exams. A person that gets 10 plus 2 bonus points in every exercise, needs to deliver only 7 assignments in order to be eligible for the exams, since 7*12=84.
- Attending the tutorial gives 2 points increase for the corresponding assignment.
- Exercise points (including any bonuses) guarantee only the admittance to the exam, however have no further effect on the final exam grade.