Computational Cognitive Science (2018–2019)

School of Informatics, University of Edinburgh

This tutorial is designed to

1. give you a closer look at a model of word learning based on Brown et al.'s IBM model
2. provide with with some exposure to matlab code that may be useful preparation for the assignment, and
3. review expectation maximization (EM).

Getting Started with Matlab

Log on to your Dice account. Type `matlab` on the command prompt. This will bring up the Matlab desktop, which consists of four panels:

- current folder;
- command window;
- workspace;
- command history.

The desktop interface of Matlab should look very similar to R-studio. If you need more help with the interface, have a look at these tutorials. There is also a handy R to Matlab translation table. From the matlab command prompt, `helpwin <function>` or `help <function>` will bring up help. Make sure you are in the right directory; for example if the file is in `/User/me/Dowloads/tutorial06/`, you can check this by running `pwd` from the matlab command prompt, and you can set the directary with `cd('/Users/me/Downloads/tutorial06/')`.

A model of word learning

Download the archive containing the files for this tutorial, linked from the tutorials page. Unpack this file, which will result in two Matlab source files, `yuModel.m`, and folders called `data`, `helper_funcitons` and `baseline_functions`. Familiarize yourself with the downloaded files. Once you had a look at the files, open `yuModel.m`. This file implements the model [in @Yu2007].

To run the Matlab code for the model, type: `yuModel` in the command window. This should run the model and print out iterations of the EM algorithm. If you see the error `Undefined function or variable 'yuModel'`, make sure that the `yuModel.m` file is in Matlab's current working directory.

A Corpus for Word-Referent Pairs

Before looking in more detail at the model, let's inspect the data. Have a look at the `corpus` and `world` file provided in the tutorial. The script has already loaded the `corpus` and `world` object in the workspace. You can inspect them by clicking on it in the workspace pane of the editor, or typing `open world`.

Question:

Access the first row in `corpus` with `corpus(1)`. This row corresponds to the first scene in our data set. What do the two arrays `objects: [4 3 19 10]` and `words: [6 205 391 66 293 50 84]` correspond to? Hint: Have a look at `world.words_key` and `world.objects_key`.

Solution:

To efficiently encode the words and objects the data is stored as arrays of integers where each integer uniquely maps to an object or word. Together with the word and object keys in `world` we can reconstruct the words and objects in each scene. The two arrays correspond to: `words: ahhah look we can read books david objects: BOOK BIRD RATTLE FACE`

Question:

Now inspect the `world` object. How many different objects were present in the videos? How many different words were uttered in the videos?

Solution:

There are 419 unique words and 22 unique objects in `world`.

Question:

You can access the database that contains the original data here. The two corpora used were `Me03` and `Di06`. What do you expect the frequencies for words and objects to look like? In general, what do you expect the distribution for words and objects to look like for young children? Consider two extreme cases - one where all words and objects are equally frequent (a uniform distribution) and one where a few are very frequent and most other words and objects are rare. How do you think does the distribution over words and objects that children encounter early in development affect their ability to learn them?

Solution:

In general we would expect the distributions for words and objects to be skewed, with a few extremely frequent words and objects and many infrequent ones. This is a general property of word frequencies (see "Zipf's law"), but the effect could be magnified in early childhood because the child will mostly be confined to the family environment. Beyond that, we might also expect this to be true of referred-to objects in early in development because head stability, gaze or grasping are not fully developed,leading caregivers to preferentially refer to familiar toys or other objects that tend to be in easy view or reach of young children.

One might imagine that heavily skewed experiences could be beneficial to initial learning of word-object mappings, as focusing on a few frequent objects as candidate matches for words will reduce the memory and attention required to track the statistical co-occurrences to a few (highly frequent) candidates, supporting or scaffording subsequent "bootstrapping". However, the actual words in the list makes it clear that concrete nouns are relatively rare, and a naive statistical learner won't do well in attaching meanings to words.

Exercise:

The `yuModel.m` script plots the object frequencies in the videos. Using this example, plot the word frequencies as a bar plot. Make sure that you display the plot for sorted word counts, i.e. the most frequent object should be the first bar. Since there are many low-frequency words in the data set only plot the 30 most frequent words. What is the most frequent word?

Solution:

```
[sorted_words, sorted_word_i] = sort(world.words_freq,'descend');
figure(2) bar(sorted_words(1:30)) set(gca,'XTick',1:30, 'XTickLabels',
world.words_key(sorted_word_i(1:30)))
```

Expectation Maximization Algorithm

The model uses Expectation Maximization (EM) to find the MLE parameters. EM is an iterative optimization procedure. In each step it updates its estimate for the MLE parameters by alternating between an expectation step (E-step), and a maximization step (M-step).

Question:

For our model, the E-step is implemented in the `computeCounts()`. In your own words describe what `computeCounts` does.

Solution:

The E-step computes the expected number of times a word evokes or "translates to" a particular meaning in the corpus, based on the associative matrix created in the maximization step. In Equation 3 of Yu and Ballard, there are two terms: (1) the "weight" for a particular word generating a particular meaning in light of the context or learning situation, and (2) the number of times the word and the meaning co-occur.

Question:

The M-step for the EM algorithm is implemented in `computeAssociations()`. Again, describe what the code does.

Solution:

It computes the associations between words and objects (the probability of a meaning given a particular word) based on the corpus counts created in the E-step.

Question:

What is calculated in `thetas = counts ./ repmat(sum(counts),lex.num_objects,1);`? What does `./` do?

Solution:

It normalizes the associations, i.e. transforms them to be in $[0, 1]$ and sum to 1. `./` is array right division in Matlab, dividing each element on the left side of the operator by the corresponding element on the right side. If the two arrays aren't the same size, Matlab will throw a `Matrix dimensions must agree.` error.

Exercise:

Change the model to run for more EM steps until it achieves a better fit. How many steps would you recommend using?

Solution:

The implementation of the EM algorithm provides us with a measure of how much the likelihood has changed in each iteration. We would expect there to be only very little change in subsequent iterations if we have found an optimum. After around 100 iterations of EM the likelihood does not change much anymore. After 220 iterations we can't be sure it's non-zero, since we're only showing two decimal places.

Exercise:

The final plot displays precision, recall and F score for our model. In your own words, how is the model fit described by these 3 measures? In general, did the model learn the associations?

Solution:

Precision, Recall and the F-score give us a way to evaluate how well our model learns the word-object mappings. They encode the information of a confusion matrix, i.e. the amount of times our model predicts a true mapping (true positive), how often we predict a true mapping when there is no mapping (false positive), how often we miss a true mapping (false negative) and how often we we predict a mapping when there is none (false positive).

Precision is the number of correct (i.e. as in the gold standard corpus) word-object matches in the lexicon divided by the total number of mappings that were inferred. Recall is the proportion of correct word-object mappings divided by the size of the gold standard lexicon. The F-score combines the two via the harmonic mean. See the table for an illustration.

You can see how these are computed in `computeLexiconF.m`.

References