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 Yu and Ballard 2007).

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`.

Question:

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

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?

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?

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.

Question:

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

Question:

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

Exercise:

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

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?

References

Yu, Chen, and Dana Ballard. 2007. "A Unified Model of Early Word Learning: Integrating Statistical and Social Cues" 70 (4). Elsevier: 2149–65.