Learning from Data – Week 5
Assignment 5:
Neural Networks and Deep Learning

General remarks

In this assignment you will work on exercises dealing with both the theory and practice of using neural networks. The practical exercise will allow you to experiment with different neural network architectures with the goal of making the best possible system for the task of noun-noun compound classification. The results from this competition will be shown continuously on the big screen, and the winner will get a prize. The theoretical part will ensure that you understand the inner workings of neural networks.

You have to hand in:

- At least one and at most five evaluation attempts during the live evaluation in the lab. If you cannot attend the lab, send in at least one submission to `hessel.haagsma@rug.nl` before the assignment deadline, and you will get back how your system ranks compared to others.
- Python code showing your final system as described in 5.1.4
- research report (based on the template) that describes what you've done for Exercise 5.1, including the visualisation of your final system from 5.1.5. It should also include answers to all questions from the assignment. Please, make sure you submit a `pdf` file.

Deadline: 21st of December, 11:00pm. Note that the final system you describe does not have to be the one you complete in the lab.

Exercise 5.1 – Practice

Noun-noun compound classification is the task of identifying the semantic relation which holds within, e.g., 'leaf blower' (OBJECTIVE), 'police abuse' (SUBJECT), 'shoe box' (CONTAIN). The semantic relations (within brackets) can be made more clear by paraphrasing the semantic relations. A 'leaf blower' is a blower *for/of* leaves, hence the OBJECTIVE label. 'Police abuse' is abuse *by* the police, hence the SUBJECT label. A 'shoe box' is a box *containing* a shoe, hence the CONTAIN label. Note that these labels vary significantly between tag sets, and that they are subject to debate. This is an important task for many higher-level NLP problems, such as machine translation. The data set we will use contains roughly 19000 annotated noun-noun compounds. You will be given annotated training data, and unannotated test data on which we will evaluate your systems. In the literature, in 2015, the highest reported accuracy was around 77.12%.

5.1.1 Features

Word embeddings have been found to yield excellent results for the task of noun-noun compound classification. The features you are given consist of the concatenation of two embeddings - one for each noun in the compound under consideration. Altering these features by, e.g., including word embeddings from other sources would be one way of increasing system performance. However, since the goal of this exercise is to experiment with neural network architectures, the features you use are **not** to be changed as part of this exercise.

5.1.2 Basic system

The Python code in `basic_neural_network.py` contains code which should help you get started. Before changing anything, make sure that you've installed everything properly by attempting to run the system as it is. What score do you get on the development data?

5.1.3 Competition

The goal of this exercise is to achieve the highest possible performance. Use the following procedure when experimenting with your neural network:

1. Change or add some parameter of your neural network
2. Evalute this on the development set (or with cross-validation)
3. Note down which parameter you changed and how this affected your score
4. If your score increases, run your system on the test set and send us your output
5. If your score decreases, change something else (you may want to reset the previous change)

Make note of all the steps you take and the procedure you use and include this in the report. Document your Python code appropriately! You should only need to make changes in the `build_model` method and, if you want to add cross-validation, the `train_model` method.

You can submit your system's output by sending the output file to `hessel.haagsma@rug.nl`, using the following filename format: `tratz_test_output_YOURNAME_runX.npy`, where `X` is the number of the run you send in, and `YOURNAME` is your name, obviously, e.g. `tratz_test_output_Hessel_run1.npy`.

The scores of your systems on the test set will be shown continuously on the big screen. The designer of the best scoring system at the end of the lab will be rewarded with a small prize! Scoring will end at: **14:45 sharp!**

5.1.4 Architecture choices

Describe the architecture you've used in your system, in terms of all parameter settings (e.g. layers, optimisation algorithm, number of neurons, activation functions, dropout, other regularisation etc., etc.). Why did you make these parameter choices? Which parameters affected your results the most? Describe the method you used to choose parameters, as noted in the section 5.1.3.

5.1.5 Architecture visualisation

Draw the neural network you've created. You don't need to draw hundreds of neurons, but make it clear that you understand how your neural network is put together (there is an example on Nestor).

Exercise 5.2 – Theory

5.2.1 Logical functions

In this exercise, you will make some calculations based on the neural network in Figure 1. You do not need to provide an exact calculation in cases where the answers are approximately 1 or 0 (as in question 2). Assume that x_1 and x_2 are binary valued (0 or 1).

1. Make a truth table which shows the outputs of this neural network given all possible combinations of values of x_1 and x_2. Assume a sigmoid activation function.
2. What are the outputs of unit $a_1^{(2)}$ given each value of x_1 and x_2, if the activation function is the sigmoid activation function? Why? Add this to the truth table.
3. What are the outputs of the same unit if a rectifier (ReLU) is used? Why? Add this to the truth table.
4. Which logical function does each *hidden neuron* represent? Add the outputs of $a_2^{(2)}$ (assuming a sigmoid activation function) to the truth table as well.
5. Which logical function does the neural network as a whole implement?

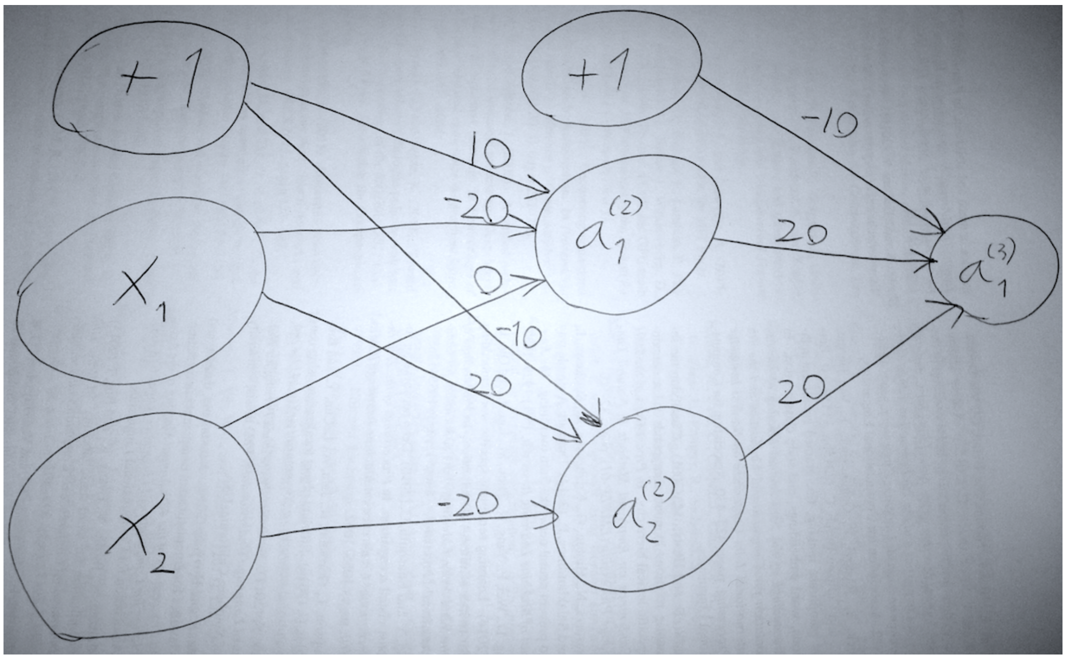

Figure 1: Neural network