COMP5400M: BIC
Coursework 2
Perceptron Algorithm and Backpropagation
Due Date: 8 May 2020, 5 pm

Marc de Kamps

April 8, 2020

Marking

This coursework is summative and counts towards your final mark for 25 %. This means that coursework 1 and 2 together account for 40 % of your final mark. It requires some implementation work, which has been restricted considerably to reflect current circumstances.

Sickness, bereavement, mental health problems, etc.

I understand that there may be many reasons why you are unable to complete the coursework by the deadline. Requests for consideration of these circumstances should go to the **SSO of the School of Computing**. I'm personally very sympathetic of the difficulty of the circumstances under which we ask you to work.

Availability of Software

Part of this coursework requires no implementation. Part of the coursework requires implementation of the perceptron learning rule. This can be done on any laptop that runs Python. Part of the coursework requires access to Keras. If you already have *anaconda* installed, installation of tensorflow and keras is straightforward through the **conda install** mechanism. If this does not work for you, consider the Google collab.

The keras implementation we will provide also runs in the Google collab. If you want to run your experiments there, you must have a Google account. With it, you can run the notebook we have provided, and you can adapt your code for the iris data set. We can demonstrate how to do this, and you should not hesitate to ask for help.

If for whatever reason you are unable to do implementation work, you must ask for an exemption of the part of the coursework that requires implementation through the SSO. This is only possible as a last resort, and you will have to make a really good case for why you can't do this part of the coursework.

Submission Instruction and Note on Plagiarism

You are allowed to discuss ideas with your colleagues. You should run the simulations yourself and not share simulation results, unless explicitly allowed (see below). The answers you provide should be based on the simulations *you* have run. All answers should be in a written report.**The report should be a PDF and your student ID, but not your name should be on the front page**. All code you have produced during the coursework must be submitted. Code that does not compile and run will be disregarded. Acceptable languages for submission of the perceptron algorithm are C/C++, Java, Python or Matlab. Submit a single tar file via the VLE, **no rar, no jar, no webarchive**.

The IRIS Data Set

Introduction

In Fig. 1 you see the picture of a flower: an iris. There are several types of irises and each one can be identified easily by a human observer.

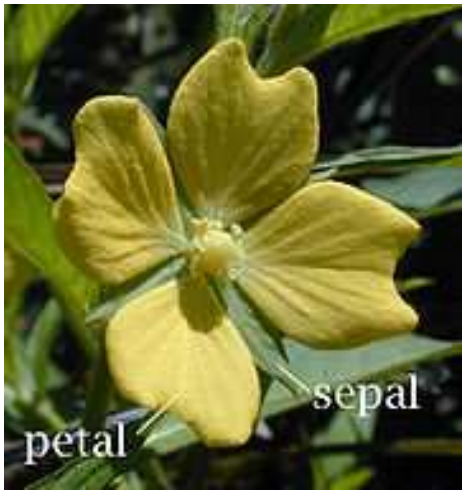

Figure 1: An iris with petal and sepal shown. Image taken from Wikipedia.

The figure also shows the petals and sepals of the flower. For 150 flowers the width and length of both petal and sepal have been measured. The measurements were obtained from 50 flowers from three iris varieties: *iris setosa*; *iris versicolor*; and *iris virginica*. The *iris data set* is a list of these measurements labelled with the variety of the flower from which the measurement were taken. Entries from the data set look like Table 1.
The full data set can be obtained in the VLE as text file. This data set was created by Anderson [1] and used by Fisher[2] to demonstrate the performance of linear classifiers.

Objectives

Clearly it is interesting to see if we can train a neural network to classify iris varieties. Given sepal and petal lengths and widths, can a neural network establish the iris variety, i.e. can the network

sepal length	sepal width	petal length	petal width	variety
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
⋮	⋮	⋮	⋮	⋮

Table 1: The first entries of the iris data set.

do iris *classification*? You will first investigate whether a perceptron network can classify iris variety, providing your own implementation of the *perceptron algorithm*. Then you will implement the backpropagation algorithm to investigate if and how this network is superior to the perceptron network.

Implementation

1. Make a *scatter plot* of petal length against petal width. This means that for each entry in the data set you plot a point in the 2D plane with petal length as y coordinate, and petal width as x coordinate. Use three different markers (or different colours), one for *setosa*, one for *versicolor* and one for *virginica*. Plot other sepal/petal length/width combinations as well.

   [10 marks]

2. Based on the plots, do you believe that *setosa* vs. non-*setosa* can be learnt by a perceptron? That is, given a perceptron with 4 inputs, for petal length, petal width sepal length, and sepal width, can you find four weights and a bias which can classify *setosa* vs. non-*setosa*? Explain your answer. If you can, give these weights and bias. Draw the decision line of your perceptron on the plots of the last question.

   [15 marks]

3. Implement the *standard perceptron algorithm without learning rate* and train a perceptron for the *setosa* vs. non-*setosa* classification problem. Do you expect the algorithm to converge? Explain why! Does the algorithm converge? Is the output correct? If it does not converge, now introduce a learning rate, and use a sensible stopping criterion. Report the learning rate, stopping criterion and argue whether the result you obtain is in line with what you know the algorithm is capable of. The same questions about *virginica* vs. non-*virginica*? *versicolor* vs non-*versicolor*? There is a major difference between versicolor and the other two. Explain the difference using the plots you made.

   [25 marks]

4. Using your earlier results, build a neural network with four inputs and three outputs. The network should produce the following desired classification: *versicolor* = (1, 0, 0), *setosa* = (0,1,0), *virginica* = (0,0,1). Do not use backpropagation. Combine the classifiers you have developed so far with some elementary logic - which you should implement using artificial neurons! Create a working program that implements your network. Your program must compile and run. Upon running, it must ask for 4 numbers: PL PW, SL, SW and produce a classification. In the report, explain the strategy you used, draw the full network and give all weights and biases. Evaluate the accuracy of your network. **[20 marks]**

5. A tutorial will demonstrate an application of the Keras framework to solve the XOR problem. This implementation will be made available to you. You will then adapt this implementation for use on the iris data set.

   Apply your implementation in a program that can classify the iris data. Create a demo program that can take 4 numbers PL PW, SL, SW and produce a classification. Evaluate the performance of your algorithm. Explain whether it is better or worse than the network you built in question 4. Give two reasons why your network cannot achieve perfect classification.

   Experiment with the number of hidden nodes. Consider experimenting with the loss function. Make sure you create a training data set, and not to evaluate your network on this training data set. **[30 marks]**

[100 marks total]

A Where are the files?

There is a file `iris.data` in the coursework 2 section on the VLE.

References

[1] E. Anderson. The irises of the gaspé peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935.

[2] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.