Assignment 3: Artificial Neural Network

March 17, 2020

1 INTRODUCTION

A grocery-robot has no knowledge of what kind of product an apple is. However, it can determine the features of a product using its sensors. Based on features like the shape, color, weight and wrapping of a product, it can recognize specific classes.

The problem can be simplified into **classification**, which can be performed by an Artificial Neural Network (ANN). The ANN could be able to recognize products by detecting a number of separate features:

- Roundness
- Color
- Weight
- etc ...

Based on the value of these features, the ANN could then categorize the product into various classes such as:

- Fruit
- Meat
- Candy
- etc ...

1.1 ASSIGNMENT

You and your team will have to write a fully functioning neural network. You will also have to write a report explaining what you did. Please include answers to all the questions from Section 2 in a structured but natural fashion. Where possible, include plots and graphics to support your answers and explain these properly.

You will create a network that can classify an input of **10 features** into one of **7 classes**. The data you will need to train and test your network can be found on Brightspace. There are three files containing comma-separated values:

- *features.txt*: 7854 samples of 10 features
- *targets.txt*: 7854 target classes of these samples
- *unknown.txt*: 784 samples of 10 features, with no classes known

It's your job to load the data, divide it into training sets and use it to train a neural network. Finally, you should use your network to recognize the classes of the unknown samples. You can not know how well your network performed on these samples until after the deadline. Your grade will be based partly on the performance of your network to this set.

You are expected to write the entire network from scratch: **a feed-forward neural network** with a **back-propagation training algorithm**. Write clear code, indent where needed and add comments to clarify what is happening. We have provided you with a Jupyter Notebook that can efficiently create and train neural networks. You will compare the results of this notebook with your own neural network.

1.2 DELIVERABLES

You will have to create a report to answer the questions and a file containing the classes of the unknown samples. Besides this, you have to clean up your code and deliver it as a compressed archive. Have a `main` file which starts (and contains comments to explain) the entire training sequence. The following files should be delivered by replacing XX with your group number(eg. 12,31,03):

- **Group_XX.zip**: Compressed file for submission. Should contain the following:
  - ***Group_XX_report.pdf***: The report as **PDF Document (.pdf)** . The number of pages should not exceed 5.
  - ***Group_XX_classes.txt***: The resulting classes of the unknown samples should be outputted as comma-separated values. Must be contained in a **Text File(.txt)** with values in a single line. Example: `7,2,4,5,6,1,3,1, ...`.
  - ***Group_XX_code***: All your commented scripts and functions you created for neural network should be inside this **File Folder**

To speed up the grading process, we want you to deliver your work exactly as outlined above. **If your files do not contain this format, your work will not be graded!**

Note: You are free to use the programming language (Python/Java) and an environment of your choosing. You are also allowed to use NumPy or ND4J as linear algebra libraries. However, you will need Python 3 and Jupyter notebook for the last set of questions.

The deadline for the assignment is **Friday, April 3th at 23:59**.

2 QUESTIONS

The following sections can help you to walk through the development of the neural network. The questions should be answered in the report. It is very much recommended to take some time to think about the network architecture before starting development.

The most important task of this assignment is to get a fully working neural network with the back-propagation training algorithm. This means that if you get to question 10 and answer all questions properly you will get a sufficient mark.
For every question it is important that you provide **proper explanation** so we can see that you master the material.

2.1 ARCHITECTURE

The basic element of a neural network is a perceptron. A perceptron is able to learn from its errors and become better after training.

1. Develop (program) a single perceptron first. Show with three graphs (error over epochs) that the perceptron is able to learn the OR and AND function but not the XOR function.

An ANN is much more powerful than a single perceptron. In the book and during the lectures the multi-layer network for an XOR-operator was introduced, which might help you to create your - more sophisticated - network. You will now design the topology of your network. For every question, briefly discuss how you arrived at your answers.

2. How many input neurons are needed for this assignment?
3. How many output neurons do you require?
4. How many hidden neurons and layers will your network have? (Give an initial guess, later you will try different number of hidden neurons and analyze the network's performance).
5. Which activation function(s) will you use?
6. Give a schematic diagram of your complete network.

2.2 TRAINING

Now develop (program) your entire MLP network. Remember that the weight update rule for hidden neurons in an MLP is slightly different than it is for a perceptron. Also remember to take into account the derivative of your activation function. The network you've developed should now be trained to get accustomed to the type of data it will be processing. It's customary to divide data into a training, validation and test set.

7. How and why did you divide your data into a training, validation and test set?
8. How do you evaluate the performance of your network? Briefly justify your answer.
9. When and why do you decide to end the training?
10. Train your network 10 times, each with different initial weights. How does the initialization impact the performance?

2.3 OPTIMIZATION

You will now use cross-validation to choose the 'optimal' number of hidden neurons. It is important to keep all other parameters fixed (learning rate, momentum etc.) to ensure that the results only portray the influence of the number of hidden neurons. Because the performance of a trained network is somewhat random due to initializations, perform each training 10 times and use the average as an indicator for performance.

11. Train your network with different amounts of hidden neurons. At least 3 times chosen within the range of 7-30 hidden neurons. Generate a plot of the final performance versus the number of hidden neurons in the network. Explain what you observe and what might be the cause of these observations.
12. Pick the architecture with the best result and show a plot of the performance of the training set and the validation set during training, across epochs. Justify your choice.

2.4 EVALUATION

It is now time to evaluate your network. Show graphically how your network performs.

13. What is the success rate of your network on the test set? How does it compare to the results of the validation set?
14. Show and discuss a confusion matrix of your test set. How should it be read? Where did your network make the most mistakes? (Search online for the meaning and purpose of a confusion matrix).
15. Feed the unknown set (provided on Brightspace) to the network. Export the resulting classes as a comma-separated file exactly as detailed in section 1.2.

2.5 SCIKIT-LEARN

The Python library *scikit-learn* provides algorithms for training neural networks, among other things. We made a Jupyter notebook which uses these features to create and train an ANN, similar to the one you just built. You can find the notebook on Brightspace.

An interesting option in scikit-learn is grid search, which lets you exhaustively search a few points in the hyperparameter space for optimized values. The notebook uses grid search to train and estimate the performance of a neural network for you.

16. Download and open the Jupyter notebook. Tweak the settings of the grid search, and run all cells to find optimized parameters, according to scikit-learn. Are the values of these parameters different from the ones you chose? What differences can you see in the training behaviour and performance of the scikit-learn network, compared to yours?

17. Finally, take the optimal hyperparameters found by scikit-learn and plug them into your own network. Does this give you better performance? Why do you think these parameters are better / worse?