A Robot to Do Your Groceries

Assignment 1: Artificial Neural Network

September 9th, 2014

1 PROBLEM

The 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:

1. Roundness
2. Color
3. Weight
4. ...

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

1. Fruit
2. Meat
3. Candy
4. ...

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.

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 Blackboard. 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. MATLAB has a complete Neural Networking Toolbox that can efficiently create and train neural networks. You will compare the results of this toolbox 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.m` file which starts (and contains comments to explain) the entire training sequence. The following files should be delivered:

- *<group>_report.pdf*: The report as **PDF** file. The number of pages should not exceed 5.
- *<group>_classes.txt*: The classes of the unknown samples as comma-seperated values. Should contain a single line with values: `7, 2, 4, 5, 6, 1, 3, 1, ...`
- *<group>_code.zip*: The neural network code.

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: If you want to create the neural network in a different environment other than MATLAB, you are free to do so. However, you will still need MATLAB for the last set of questions.

The deadline for the assignment is **Friday, September 19th at 23:59**.

1.3 Tips

MATLAB has a number of useful commands to speed up the development of your neural network. Note that the last two commands are part of the neural network toolbox. Consult the MATLAB docs for examples of these commands.

- `dlmread`: Read ASCII-delimited file of numeric data into matrix. This command is useful to read the data files. Make sure to correctly transpose the matrix if needed.
- `dlmwrite`: Write a vector/matrix to ASCII-delimited file. Can be used to create one of the deliverables: the vector of recognized classes.
- `ind2vec`: Convert indices to vectors. This is useful to transform the target class to a network output vector.
- `vec2ind`: The opposite, converts the network output vector to an index.

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 9 and answer all questions properly you will get a sufficient mark.

2.1 ARCHITECTURE

You will now design the topology of your network. In the book and during the lectures the multi-layer network for an XOR-operator was introduced, which might help you to create this - more sophisticated - network. Briefly discuss how you arrived at your answers.

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

2.2 TRAINING

The network you've written 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.

6. How and why did you divide your data into a training, validation and test set?
7. How do you evaluate the performance of your network?

8. When and why do you decide to end the training?
9. 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.

10. 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. (*Hint:* you can use the MATLAB command `boxplot` or `errorbar`)
11. 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.

2.4 Evaluation

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

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

2.5 MATLAB's Toolbox

MATLAB offers a Toolbox dedicated to Neural Networks. The theory running behind this is a lot more complex than what is treated in the course. You can use the Toolbox to see how it's result differ from yours.

15. Use the Toolbox to create a network similar to the one you've just made. (You can start the toolbox GUI with `nnstart` or directly go to pattern recognition with `nprtool`).
16. Comment on the differences between your network's performance and the Toolbox.