Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 1

## Learning and generalisation in feed-forward networks — from perceptron learning to backprop

Nuo Chen, Author 2 and Author 3

#### September 12, 2018

1. **Main objectives and scope of the assignment**

* Design and apply networks in classication, function approximation and generalisation tasks.
* Identify key limitations of single-layer networks
* Configure and monitor the behaviour of learning algorithms for single- and multi-layer perceptrons networks
* Recognise risks associated with backpropagation and minimise them for robust learning of multi-layer perceptrons.

# Methods

The algorithms are implemented in Python 3.8, executed and shared on Jupyter notebook. The following packages are used for matrix operations and plotting: Numpy, Plotly.

Mention here in just a couple of sentences what tools you have used, e.g. programming/scripting environment, toolboxes. If you use some unconventional method or introduce a clearly different performance measure, you can briefly mention or define it here.

# Results and discussion - Part I

### Classification with a single-layer perceptron *(ca. 1 page)*

Combine results and findings from perceptron simulations on both linearly separable and non-separable datasets. Answer the questions, quantify the outcomes, discuss your interpretations and summarise key findings as conclusions.

### 3.1.2.1

After 100 epochs and at the same learning rate (1e-3), both algorithms managed to classify all the points correctly. The perceptron learning algorithm had smoother learning curve than delta learning rule.

### 3.1.2.2

### For sequential mode, it takes much fewer epochs for the algorithm to converge, and it has much smoother learning curve than batch mode. Apparently, the algorithm is very sensitive to the distribution and the initial weights. A slight change of the initialization, has huge impact on the initial learning.

### 

### 3.1.2.3

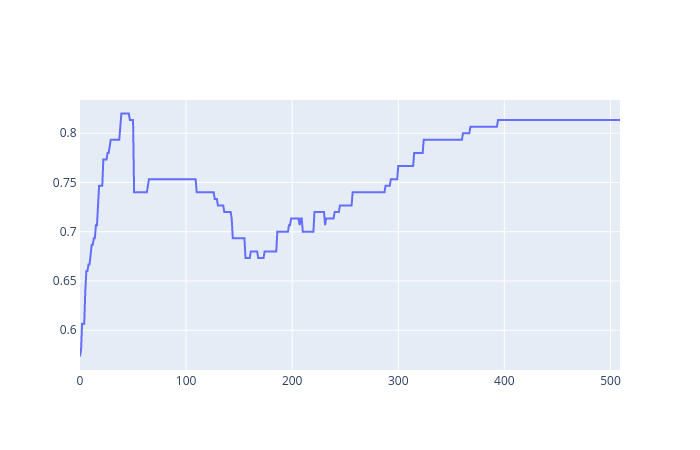
### Removing the bias term (the extra row) made the model dependent on the initial initialization, since its origin will be fixed at (0,0).

### 

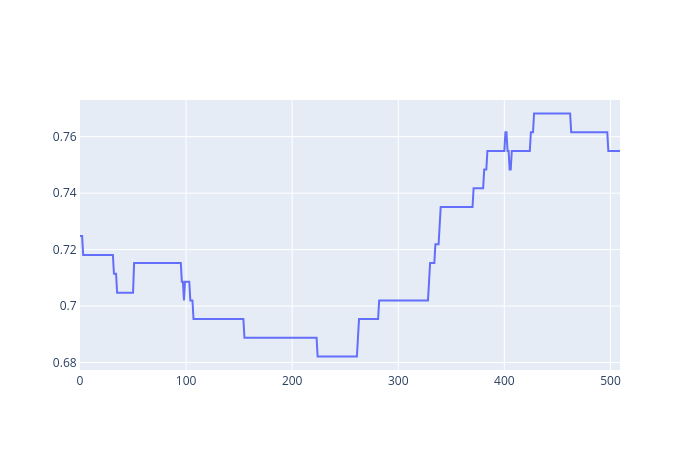
### 3.1.3.1

### The subsampling changes the boundary and the performance. By having more samples of one class, makes the model learn better to classify points from that class, thus changes the position and slope of the decision boundary. The performance is also decreased after first few iterations, but slowly increased afterwards.

### 

Figure 1: Scenario 1

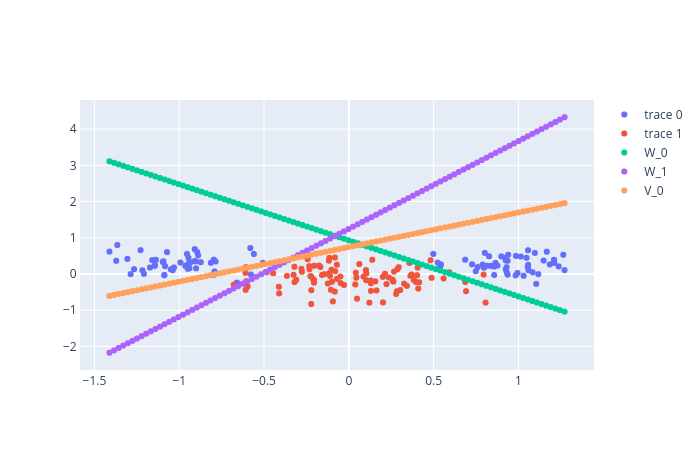
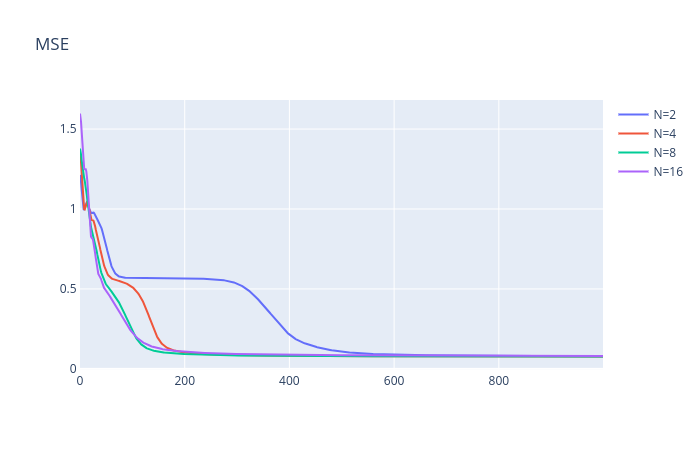
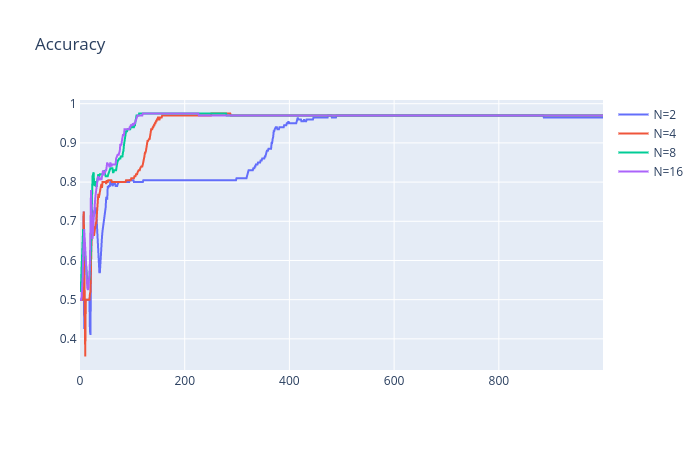
### 

Figure 2: Scenario 4

### The subsampling bias will cause the model to be biased as well, as it will perform better at classifying the one class that has dominant representation in the training set.

### Classification and regression with a two-layer perceptron*(ca. 2 pages)*

##### Classification of linearly non-separable data



With more nodes, the model can have more complex shape and it will fit the training data better. It takes 8 or more nodes to reach an accuracy higher than 95%.

The training curve and the validation curve are similar in most cases, with validation accuracy slightly lower than the training accuracy. When the model starts to overfit, meaning the model is to adapted to the training set, the training error will keep decrease but the validation error might start increase instead.

The more nodes, the complex the model will become, thus much easier to overfit. The model will perform worse on unseen data.

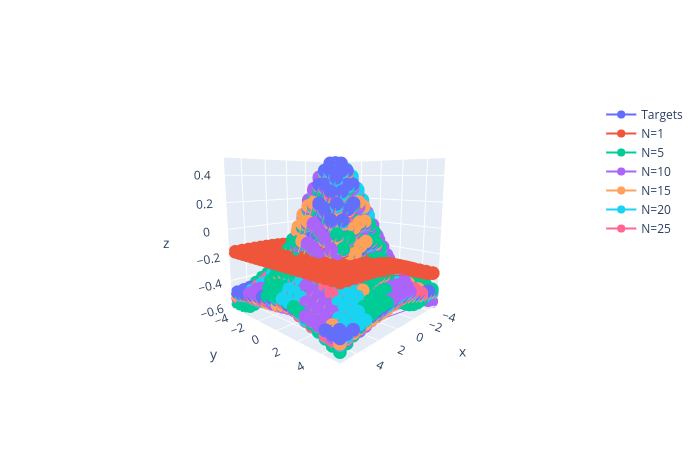
It seems that one (decision boundary) or two plots (including learning curves) should suffice. Build a story around the questions in the assignment. Include concise motivation for your findings and potential interpretations/speculations.

##### The encoder problem

##### The network will always converge in this case. The patterns of internal activations of all nodes should represent the input vectors. There are 8 combinations of input vector, and there are 3 hidden nodes, 2³ = 8. This network should be able to encode the inputs.

##### With n=2, it will still be able to map all input vectors, but not all activations will have unique values (signs)

##### One usage of autoencoder is to encode and compress the data, to represent it with fewer bits.

Here you do not really need any illustrations, this could be a very short section reporting on your experiments in line with the assignment questions.

##### Function approximation

The graph indicates that with 5 hidden nodes, the network should be able to approximate the function with acceptable deviations. With further experiments, it shows that it is enough to train this model for 100 epochs and a learning rate at 0.1, with 80/20 split ratio for training and validation sets. Since the momentum is used for updating weights, it allows us to use higher learning rate, hence making the learning faster.

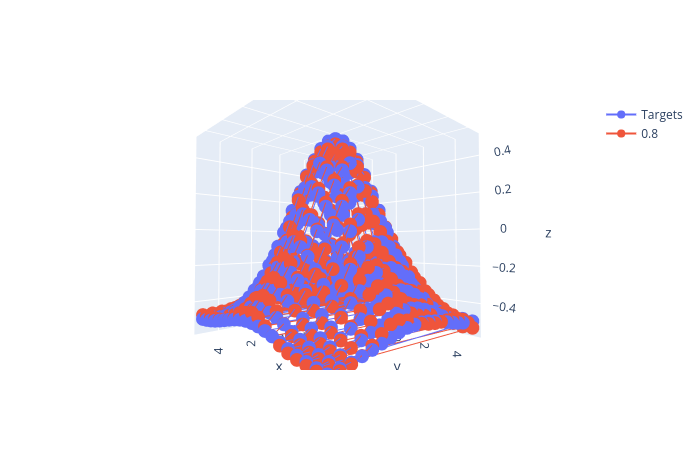
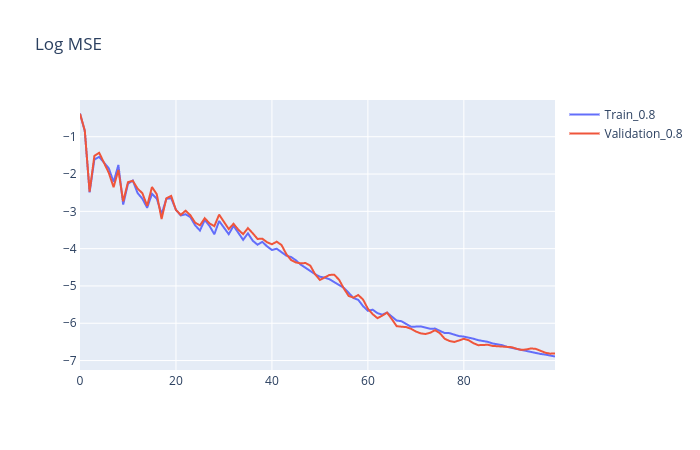


Figure 3: Epoch=100, ETA=0.1, Training\_size=80%,Momentum=0.9

This subsection requires plots to reflect intuitive visual interpretation of the results. Make sure that you condense information and avoid any excessive plotting. Here you may need to incorporate some illustration of the network’s generalisation performance or use a table to systematically report the results requested in the assignment.

1. **Results and discussion - Part II *(ca.2 pages)***

Here you do not have to introduce the problem or define Mackey-Glass time series, as you should focus on the results. You could divide them into two parts as the following two suggested subsections but you might as well keep your story under the main heading of Part II of the assignment. Importantly, always clearly state what network architecture you use, crucially with the number of hidden nodes, systematically report average results with various manipulations (regularisation etc.) and pay attention to differences between training, validation and test errors. Illustrating the outcome of your network predictions along with the original chaotic time series can also be very helpful. Finally, since you compare two- and three-layer architectures, make sure that you do not jump to any conclusions based on a small number of simulations unless you have statistically convincing evidence (when you compare the mean performance measures, their second moment is also relevant). In this part it may be particularly desirable to rely on tables.

### Two-layer perceptron for time series prediction - model selection, regularisation and validation

* 1. **Comparison of two- and three-layer perceptron for noisy time series prediction**

1. **Final remarks** *(max 0.5 page)*

*Please share your final reflections on the lab, its content and your own learning. Which parts of the lab assignment did you find confusing or not necessarily helping in understanding important concepts and which parts you have found interesting and relevant to your learning experience?*

*Here you can also formulate your opinion, interpretation or speculation about some of the simulation outcomes. Please add any follow-up questions that you might have regarding the lab tasks and the results you have produced.*