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

Mustafa Al-Janabi, Einar Lennelöv and Akanshu Mahajan - Group 15

January 24, 2020

1 Main objectives and scope of the assignment

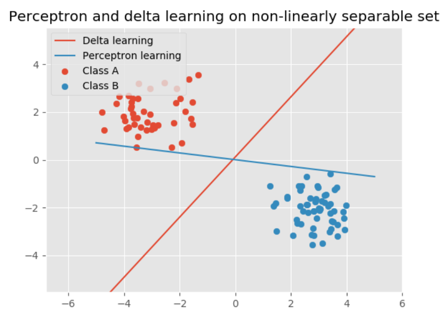
Our major goals in the assignment were

- Investigate the differences between perceptron learning and delta learning for perceptrons.
- Investigate the effect of non-uniform sampling of validation sets.
- Investigate the performance of multilayer perceptrons on the tasks of classification, autoencoding, function approximation, and time series prediction.

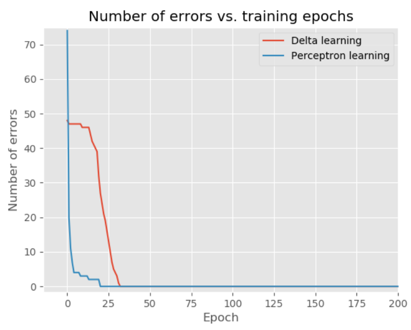
We have focused on qualitative findings that help in the understanding of the algorithms, as opposed to more rigorous numerical results.

2 Methods

In this report the majority of the code is written in Python, utilizing the libraries Numpy, Matplotlib, Pytorch. For the last part Matlab's NN toolbox is used to obtain alternative results.



(a) Decision boundaries



(b) Convergence of errors

Figure 1: Comparison of perceptron learning and delta learning on a single dataset.

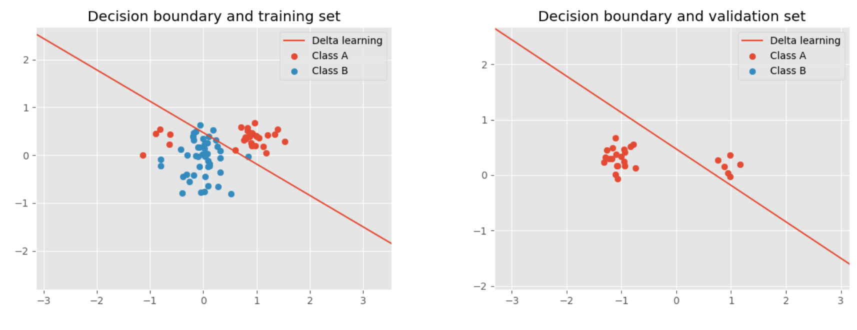
3 Results and discussion - Part I

3.1 Classification with a single-layer perceptron

We begin by training the models on a simple linearly separable dataset, and achieve the decision boundaries shown in Figure 1a. It is clear that the boundary given by the delta learning will generalize better. This illustrates the disadvantage of forcing the algorithm to stop once zero errors are achieved. Figure 1b illustrates the convergence rate. In this case the perceptron rule converged faster.

One way of measuring the performance of the model is by applying it to a validation set that is separate from the training set. We investigate the effect of sampling this validation set a linearly non-separable dataset. Table **??** shows the validation results for four different cases of subsampling. It is clear that the distribution of the validation set has a major impact on the validation accuracy. Figure 2 shows the most extreme case, when 20% of the samples from one of the two class A clusters are sampled, and 80% from the other.

Class A	Class B	EMS $\pm$ 1std.
25%	25%	0.76 ± 0.06
50%	0%	0.58 ± 0.09
0%	50%	0.43 ± 0.08
$\{20\%; x_1 \leq 0\} \cup \{80\%; x_1 > 0\}$	0%	0.23 ± 0.09
??		



(a) (b)

Figure 2: Decision boundaries with asymmetric subsampling.

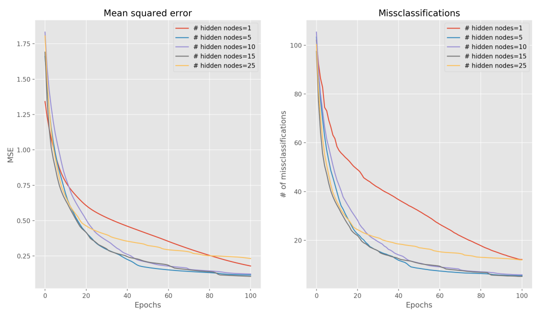
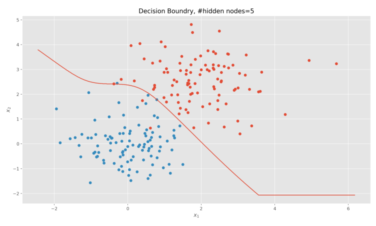


Figure 3: Studying the number of required hidden nodes.

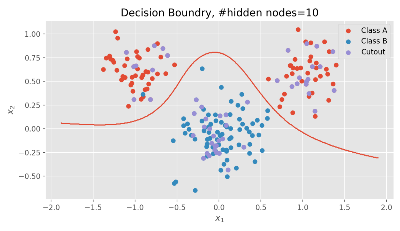
3.2 Classification and regression with a two-layer perceptron

3.2.1 Classification of linearly non-separable data

In this section we solve the problem of the linearly inseparable data using a multi-layer approach. By studying figure 3 which was generated by averaging over 200 different data generations and data fittings. We observe that the MSE and number of miss are similar. Just studying one of them suffices. After 100 epochs it is clear that 1 neuron in the hidden layer is note enough. However 25 neurons and above too many parameters for the network to learn within 100 epochs. Initially the higher the number of neurons the larger the error, this is due to the fact that the first guess of the weights is more likely to be wrong given more parameters. From the plot we conclude that the most appropriate number of neurons in the hidden layer is 5. This was performed on data randomized according to that shown in figure 4a. As we can see, for that particular randomisation it is not possible to achive perfect seperation even for a high order model.



(a) 3 misclassifications, with MSE= 0.07 and learning rate $\eta = 0.01$



(b) Decision boundary, with the cutout data highlighted. No misclassifications for either the training data and testing set

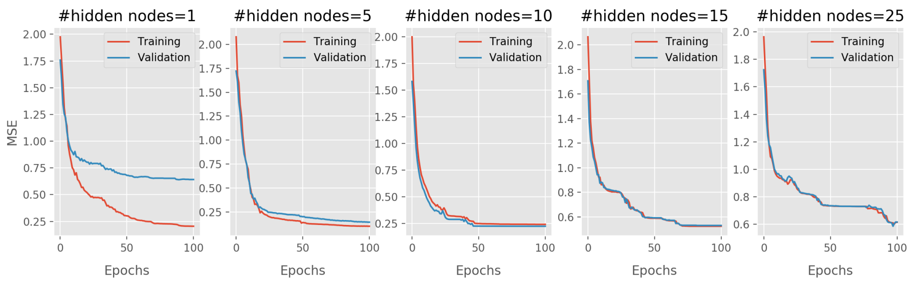


Figure 5: Comparing the number of nodes for the training and validation set.

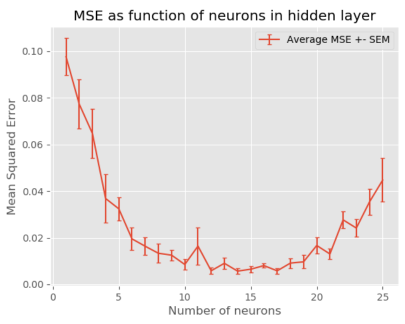
Figure 5 illustrates the effect that different number of nodes has on generalization ability. The validation is based on a sub sample of 25% from each class. With only one hidden node the model lacks the capacity to properly fit the data, leading to high error on the training set and even higher error on the validation set. With 5 and 10 nodes the validation performance is similar to the training performance, indicating good generalization. With more hidden nodes the model is unable to converge within the 100 epochs used for training. This highlights a drawback of additional hidden nodes. The decision boundry for the data randomization used for this part is shown in figure 4b.

3.2.2 The encoder problem

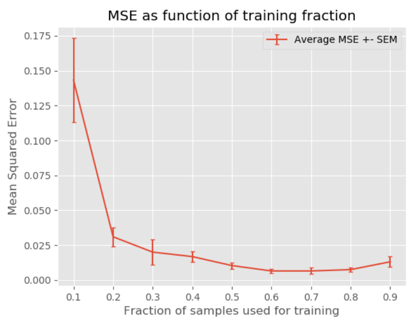
In the data encoder problem it was found that the algorithm didn't always converge. However accuracy seemed to increase with the increasing of the learning rate.

When applying the three node network on several inputs and after thresholding, the following result was obtained for the encoding in the hidden layer:

$[-1,-1,1,-1,-1,-1,-1,-1] \implies [-1,-1,-1]$
$[-1,-1,-1,-1,-1,1,-1,-1] \implies [1,1,-1]$
$[1,-1,-1,-1,-1,-1,-1,-1] \implies [1,1,1]$



(a) Effect of number of neurons.



(b) Effect of training set size.

Figure 6: Decision boundaries with asymmetric subsampling.

Which suggests that each configuration is stored in a $2^3 = 8$ combinations of 3-element vector.

In a two-neuron setup we obtain the following given a thresholding for -1 or 1.
$[-1,-1,1,-1,-1,-1,-1,-1] \implies [-1,-1]$
$[-1,-1,-1,-1,-1,1,-1,-1] \implies [-1,-1]$
$[1,-1,-1,-1,-1,-1,-1,-1] \implies [1,-1]$

If we threshold using a continous function, then even a two layer setup should in theory be able to encode all the input the outputs perfectly.

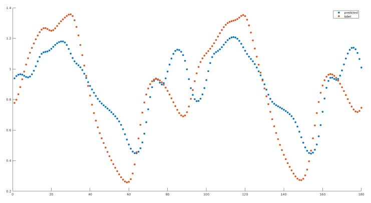
3.2.3 Function approximation

We examine the abilities of the two-layer perceptron to perform function approximation by attempting to approximation a two-variable gaussian. Figure 6a quantitatively shows the importance of selecting an appropriate number of hidden neurons to avoid overfitting and underfitting. We find 14 hidden neurons to be the top performer. 6b illustrates the strong generalization performance of the network. The reduction in MSE for high fractions of training data is strange, however, and we were not able to explain it.

4 Results and discussion - Part II

It can be inferred from Figure 7a and Figure 7b that a 2 layer architecture did not perform well and an average R value of approx 0.8 was observed. Further, changing the no. of hidden nodes from 5 did not help very much and hence a 3 layer architecture was tested.

It was observed that the error converged more rapidly for a 3 layer configuration


(a) Predicted (blue) and target (orange) time series.

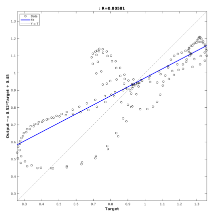
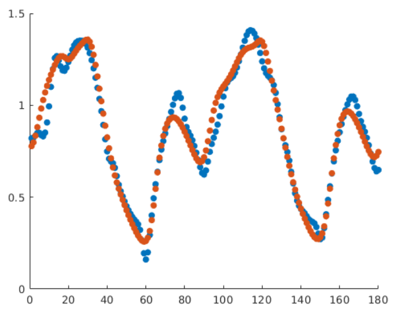
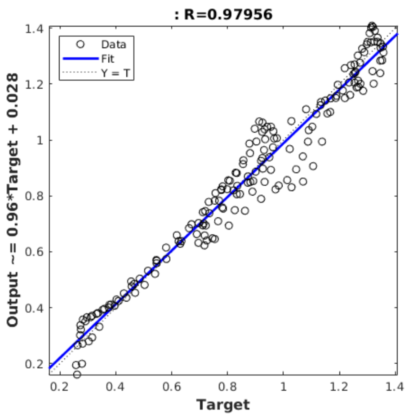

(b) Correlation between data and target.

Figure 7: Results with 2 hidden layers.



(a) Predicted (blue) and target (orange) time series.



(b) Correlation between data and target.

Figure 8: Results with 3 hidden layers.

and as soon as the validation error started to become equivalent to the training error, early stopping was used in order to avoid overfitting. It can be inferred from Figure 8 that the network performed well on held out data set

5 Final remarks

We felt that the overall coverage of the lab was very good, and that it has given us a good understanding of the foundations of simple perceptron networks. Being made to implement things from scratch helped a lot with the understanding. However, we also felt that there were too many subtasks. We spent a large amount of time writing code for the various plots and investigations, which meant that we had no time for investigating other things that we thought were interesting. We would have preferred a smaller set of larger, more theoretically demanding tasks.