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

Fernando Garcia, Lucas Q. Gongora and Maria Bjelikj

January, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to implement single and multiple layer artificial neural networks
- to apply the implementation for classification of data that is linearly separable and for data that is not linearly separable, and for time-series data
- to analyse the results and compare the different approaches

2 Methods

Python version 3 was used for this assignment. For the single layer implementation, the Numpy toolbox was used, as well as Matplotlib for plotting. For the second part of the assignment, the packages Keras and Sklearn were used.

3 Results and discussion - Part I

3.1 Classification with a single-layer perceptron

3.1.2

Figures 1 and 2 are comparing the delta learning rule in batch mode and perceptron learning. Both algorithms effectively separate the two different classes for all data samples with error at 0 at convergence. Figure 2 shows the learning curve for 50 different learning rates η in the interval $[0.0002, 0.01]$. The convergence criterion is defined as a change in the weight matrix smaller than 0.001. In this interval for the learning rate, we can see that perceptron learning takes less epochs to converge the greater the learning rate is, however for the delta learning, the convergence stops at learning rate 0.0035. Thereafter, the learning rate is too large and the weights still keep changing too significantly for the convergence criterion.

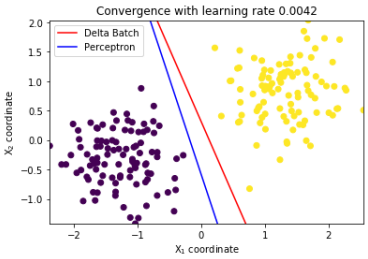


Figure 1: Classification of linearly separable data using the delta learning rule in batch and perceptron.

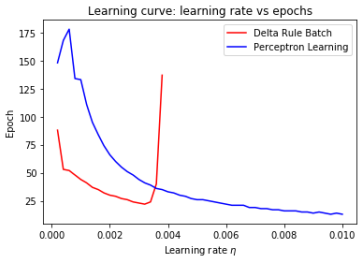


Figure 2: Learning curve plotted as learning rate versus epoch.

When comparing the classification for the batch learning approach versus the sequential approach, the decision boundaries from each of the algorithms overlap. This was explored for different learning rates in the interval $[0.0001, 0.001]$, and the decision boundary overlapped in all cases after convergence. The delta learning in batch converges in less epochs than the delta learning sequentially. Learning was not very sensitive to the random initialization, in the sense that both algorithms behave similarly when compared; however, convergence could take a few more epochs for different cases in initialization.

Without bias, the decision boundary cannot translate and can instead only rotate around the origin. Therefore, some points remain misclassified, depending on how the data is generated in regards to the origin.

These algorithms are not suited for linearly non-separable data, no matter how this data is generated. For non-linearity, hidden layers are required in the neural network. Figure 3 shows the decision boundary on unseen data, which consists of 25% of data from class A and 25% from class B. Figure 4 plots epochs vs. accuracy for different sizes of test data.

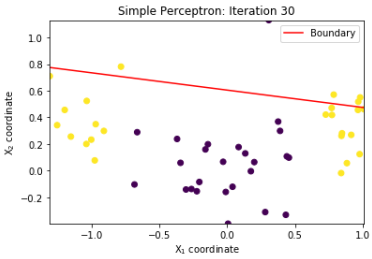


Figure 3: Classification using perceptron learning on linearly non-separable test data consisting 25% class A, and 25% class B.

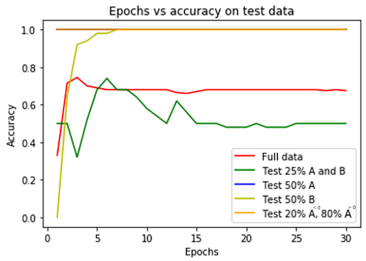


Figure 4: Learning curve for test data consisting of various sizes.

3.2 Classification and regression for two-layer perceptron

3.2.1 Classification of linearly non-separable data

The following results (table 1) have been obtained generating data for two 100-size different classes and trained with different sizes for hidden layer.

	H.L. Size 1	H.L. Size 2	H.L. Size 3	H.L. Size 5	H.L. Size 10
Final Error (MSE)	126.407	13.962	3.613	7.029	7.751
Misclassified Data	41.0	4.33	0.67	1.67	2.67
Misclassified Ratio	0.205	0.0217	0.003	0.008	0.013

Table 1: Benchmarking with different hidden layer sizes (averaged).

Results show that data is properly separated when three-size hidden layers are used, and therefore this configuration is further used for testing. Then, several analysis are done: 1^{st} consists of 25% of the elements of each class of the original dataset, 2^{nd} of 50% of the elements of class A, 3^{rd}, 50% of the elements of class B, and 4^{th} of elements from class A fulfilling several conditions.

	25% both	50% A	50% B	Subsets A
Final Error (MSE)	3.249	5.966	7.175	13.975
Misclassified Data	0.8	1.8	2.0	4.2
Misclassified Ratio	0.016	0.036	0.040	0.056

Table 2: Benchmarking multilayered perceptron (averaged).

Applying the same dataset, it is possible to obtain the learning curves during the training phase and during the testing one, by number of epochs, and observe their evolution.

For the plot 5 3-size hidden layers were used, providing an higher error for testing than for training after converging. The opposite happens when using a greater number of nodes. This allows the definition of more complex boundaries, which take more time to converge, but provide better results, even though, they have a tendency to overfit.

The boundaries in the plot 6 represent the result for the decision boundary, which is generated using 75% of both classes for training, and 25% for testing.

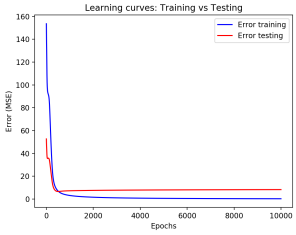


Figure 5: Learning curves employing a non-linearly-separable dataset composed by a training set of 75% of class A and class B and a test set of a 25% of each class. 3 nodes were used.

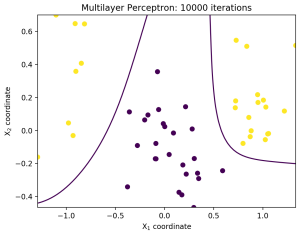


Figure 6: Classification using a multilayer perceptron on linearly non-separable test data consisting 25% class A, and 25% class B.

When the model is trained using batch learning mode, all data points are used in the same iteration, providing a faster training than that of the sequential one, which adds one sample per iteration when training. Nevertheless, sequential learning allows to train over each sample, so the training process will be more

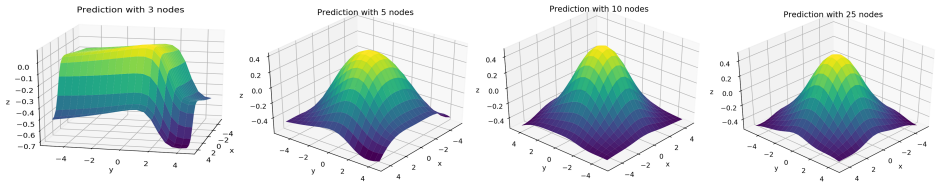
complete, since it uses the previous information when training over a new data point. This allows to keep training a model when new information is received, meanwhile with batch learning, the model has to be entirely retrained [2].

3.2.2 The encoder problem

Setting the convergence criterion as 10000 iterations or as a 0.1 difference between two consecutive MSE resulted in no misclassified elements and a MSE of 0.9804 in the 8-3-8 architecture. Using the 8-2-8 architecture, the number of misclassified predictions is 7 and the MSE is 2.2310, because it reaches the convergence criterion before reaching error 0, which it should not; once for binary data, a 2 size hidden layer will solve only $2^2 = 4$ dimensional system. The gradient of the C function will show the weight sign and the intensity of how much it should change to minimize the cost function.

The internal code is described by a hidden layer which represents the input, containing two parts. The first part is called the encoder, where the input is compressed, and the second part (decoder) is where the input is transformed back again. Originally, the autoencoder was created to reduce dimensionality, but nowadays it is more often used as a feed-forward network and trained with gradient descent [1].

3.2.3 Function approximation



The plot above compares models of 3, 5, 10 and 25 nodes, respectively. These models present the following errors (MSE), respectively: 15.399, 1.906, 0.386, 0.266. According to it, it can be seen that the model trained with 25 nodes is the closest one to the real values, even though it might overfit, something that could be checked during the validation phase.

Now, observing that apparently there is not a big difference between the model of 10 nodes and the one of 25, it is possible to check the error in validation for both models, to find the one which is more reliable:

Table 3 shows that the greater the training dataset size over the test, the lower the test error. For all cases, the 25 nodes model performs better than the other, so it can be said that the 25 nodes model is the best model.

H.L. Size	80% Tr.	60% Tr.	40% Tr.	20% Tr.
25 nodes	0.004	0.091	0.799	21.699
10 nodes	0.012	0.066	0.420	47.091

Table 3: Final Error (MSE) of Benchmarking for different nodes multilayered perceptron (averaged).

Convergence can be improved increasing the learning rate. A greater rate will trigger faster changes in the model, allowing to achieve convergence before, getting the same model generated in more iterations with a lower rate.

4 Results and discussion - Part II

4.3.1 Two-layer perceptron for time series prediction

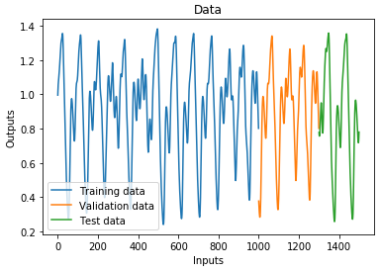


Figure 7: Mackey-Glass data as split for training, validation and testing.

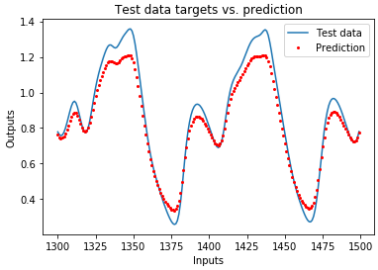


Figure 8: Prediction of test data obtained with a two-layer 8 units NN and ridge with $\lambda = 0.01$.

L_2 distance	$\lambda = 0$		$\lambda = 0.001$		$\lambda = 1$	
Hidden units	Val. MSE	Test MSE	Val. MSE	Test MSE	Val. MSE	Test MSE
2	$0.0253 \cdot 10^{-3}$	$0.0309 \cdot 10^{-3}$	0.871	0.8408	0.7816	0.7548
5	$0.0295 \cdot 10^{-3}$	$0.0329 \cdot 10^{-3}$	0.00086	0.00091	0.7988	0.7713
8	$0.0291 \cdot 10^{-3}$	$0.0298 \cdot 10^{-3}$	0.00334	0.0032	0.7834	0.7565

Table 4: MSE for the validation and test for hidden layer configurations.

Table 4 compares the validation and test MSE for different configurations in terms of hidden units and the Ridge regularization parameter λ. The smallest test MSE was obtained for the 8 units in the hidden layer and $\lambda = 0$. A two-layer NN is not prone to overfitting, thus regularization is not necessary. Figure 9 shows the weights histogram for the selected values for λ. The higher the value of λ, the closer the weights are to 0.

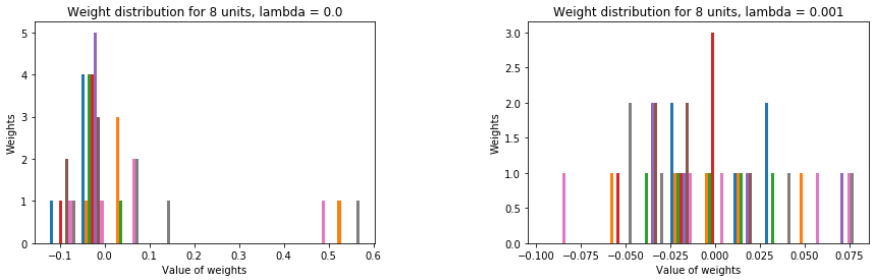


Figure 9: Weights histogram for $\lambda = 0.0$, $\lambda = 0.001$.

4.3.2 Comparison of two- and three-layer perceptron for noisy time series prediction

Using a configuration of 4-1 nodes per hidden layer, and applying an error of $\sigma = 0.03, 0.09, 0.18$ in each case, the following results were obtained.

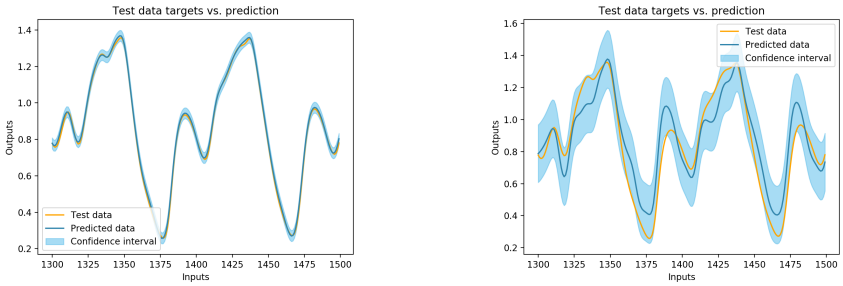


Figure 10: Predictions for $\sigma = 0.03$ and $\sigma = 0.18$.

For $\sigma = 0.03$, a validation MSE = 0.002 was obtained, and a test MSE = $9.92{\cdot}10^{-05}$. For $\sigma = 0.09$, the validation MSE is 0.012 and a test MSE is $2.96{\cdot}10^{-03}$. $\sigma = 0.18$ presents a validation MSE = 0.022 and a test MSE = 0.014. As it can be seen, the greater the σ, the greater the error in the predictions, since data are more spread.

When applying regularization for 8-8 size of hidden layers, with a $\lambda = 0.01$, validation MSE and test MSE were [0.0117, 0.0030] for $\sigma = 0.09$ and [0.0229, 0.0014] for $\sigma = 0.18$. For $\lambda = 0.1$, $\sigma = 0.09$ got [0.0166, 0.0099] and $\sigma = 0.18$ got [0.440, 0.0053]. In this case, applying regularization, the errors obtained are much lower than those of the previous configuration without applying any regularization factor.

Setting $\sigma = 0.09$, the best three-layer model, without regularization, is the one with 2-4, the best configuration was made based on the MSE of validation set, which got a result of 0.0099, also, for the MSE test set we got 0.0024 error. The performance of the three-layers is very similar to the two-layers for $\lambda = 0$ with σ of 0.09 (MSE validation = 0.010 and MSE test = 0.002). The noise will give more random data specially for higher values of noise, which increases the test error of the prediction. For two-layers with size 8 the time cost was 4.87 seconds and for three-layers of sizes 8 and 4 the time was: 5.19 seconds. For different size 8 and 8 of the three-layers we got: 5.36 and for 8 and 1: 5.01.

5 Final remarks

First of all, we achieved our goal of learning and implementing different kinds of artificial neural networks. However, we thought that the assignment was slightly repetitive one we had to provide a lot of different tests that yielded quite similar results. The page limit of the assignment constrained us diving into to much detail and exploring even more different cases.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[2] Z. Runxuan. Efficient sequential and batch learning artificial neural network methods for classification problems. *Singapore*, 2:825–845, 2005.

[3] U. Tutorial. Autoenconders. `http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/`, Acessed in January, 2020.