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

Ivan Stresec, Frano Rajic

September 11, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to investigate the performance of classification single-layer perceptrons (SLPs) in various settings
- to observe the capabilities of MLPs in the context of solving non-linearly separable patterns in both classification and regression problems
- to implement MLPs in a chaotic time-series prediction[1]

2 Methods

For the first assignment we used Python as a programming language and its library Numpy for data representation and matrix operations. For graphs we used the Matplotlib library. In the second assignment we used Pytorch, as well as the aforementioned libraries. As an IDE, we've used JetBrains' PyCharm for both assignments.

[1] FIX THIS ONE

3 Results and discussion - Part I

3.1 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.

In solving classification problem using an SLP one can easily observe its efficiency and its limitations.

Firstly, we have implemented the perceptron rule (implying a sequential learning scheme) and the Delta learning rule (batch learning) to a randomly generated, linearly separable classification dataset. Both have performed well and converged quickly...

Secondly, we have tested the differences between batch and sequential learning for the Delta rule SLP. In terms of epochs, both approaches converge relatively quickly, with ... being somewhat quicker

Thirdly, we have tested removing the bias term with the Delta rule in batch mode. Even without testing it, it is clear that the algorithm can converge only when the data is linearly separable by a line which goes through the center of the coordinate system.

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

3.2.1 Classification of linearly non-separable data

3.2.2 The encoder problem

3.2.3 Function approximation

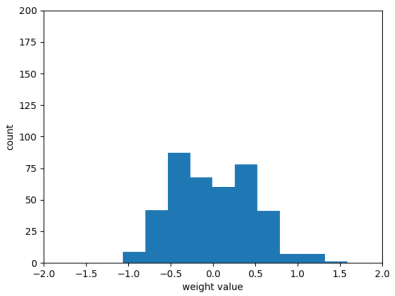
4 Results and discussion - Part II *(ca.2 pages)*

For this assignment, we have used the Mackey-Glass time series to evaluate a two-layer perceptron network. The data was simply generated by using the starting conditions and sequentially calculating values using the iterative formula given in the assignment. Furthermore, the data was split into three consecutive non-overlapping blocks for training, validation, and testing using 800, 200, and 200 values, respectively. For the regularisation method we have used

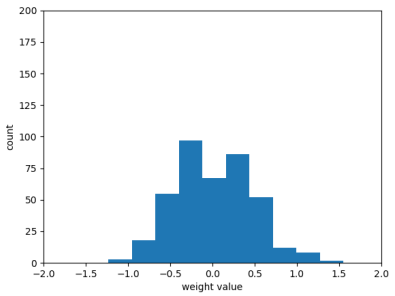
the standard L2 norm. Early stopping is implemented in such a way that if improvement (by at least 10^{-5}) in the validation mean square error (MSE) is not visible for 50 iterations the learning stops. Also, the maximum number of epochs allowed is capped at 5000.

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

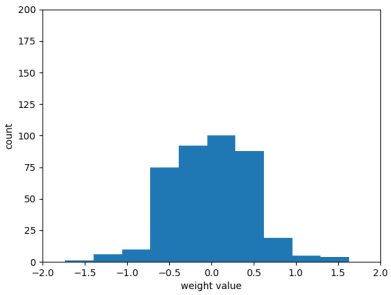
To test the influence of regularization methods on weight distribution we have used regularization rates of 0, 0.01, 0.1 and 1 with 10 networks with 8 nodes in the hidden layer each, and made histograms using the accumulated weights for each regularization rate. The histograms can be seen in figure **??**.



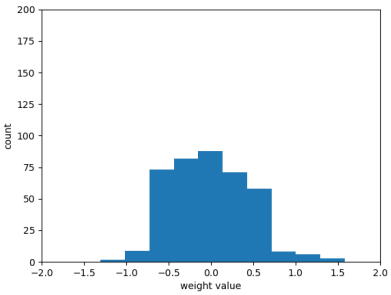
(a) No regularization



(b) L2 regularization with $\lambda = 0.01$



(c) L2 regularization with $\lambda = 0.1$



(d) L2 regularization with $\lambda = 1$

Figure 1: The distributions of weights for different regularization strengths

The differences between the distributions are not too clear, and this can probably be attributed to the fact that a simple two-layer network such as ours does not suffer from overfitting, and as such has no problems with overly large weights. Generally, networks which use higher regularization strength should stick closer to smaller weights, while those which do not show a more even distribution. This can be attributed to the fact that a penalty-based regularization, such as L2, forces weights to be smaller by 'punishing' their size in the cost function (which can be seen in equation 1) and in turn reduce the complexity

nodes	learning rate	MSE mean	MSE std
2	0.01	0.xxx	0.xxx
2	0.1	0.01488	0.00220
4	0.01	0.xxx	0.xxx
4	0.1	0.01556	0.00295
6	0.01	0.xxx	0.xxx
6	0.1	0.01488	0.00210
8	0.01	0.xxx	0.xxx
8	0.1	0.01354	0.00236

Table 1: Evaluation of various network configurations (L2 - $\lambda = 0.1$)

of the model.

$$cost = MSE + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m} w_{i,j}^2 \tag{1}$$

Table **??** shows the mean and standard deviation values of the loss function (MSE) for several configurations. Generally, networks with a lower number of nodes in the hidden layer tend to have larger errors, probably due to an overly simple model, i.e. underfitting. For experiments we have used a fixed L2 regularization with $\lambda = 0.1$.

Finally, we have decided to use a network with a learning rate of 0.1 with X nodes in the hidden layer, as well as the aforementioned L2 regularization with a rate of 0.1 to evaluate the final solution. Figure **??** shows error rates of such a network on the training and validation set, and figure **??** shows the predictions of the network alongside the data itself.

The network shows good generalisation on the test set and seems to have no trouble with overfitting.

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

5 Final remarks *(max 0.5 page)*