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

Benjelloun Hamza, Benchekroun Omar and Ait lahmouch Nadir

January 27, 2021

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to implement and apply both Perceptron and Delta rule algorithms and understand the underlying principles with visualization.
- to implement the Multi-Layer-Perceptron (from scratch) and compute it on nonlinear functions
- to understand the impact and role of some hyperparameters (initialization of the weights, learning rate, number of epochs, size of hidden layer, regularization) and their importance in the choice of a model.

1 Methods

In this Lab, we used **Python.** The **numpy** library was used in Part I for the matrix manipulations and operations. In Part II, we used **keras.**

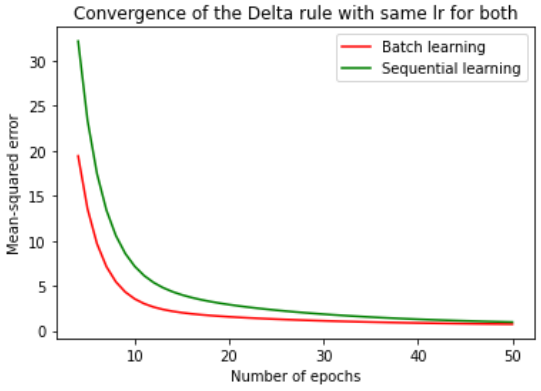
2 Results and discussion - Part I

1.1 Classification with a single-layer perceptron

A) Classification with a single-layer perceptron and analysis

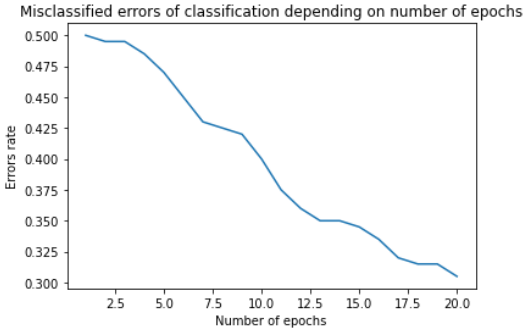
To answer these questions, we generated 100 points of class A and B each, our parameters are: mean_A= (1.5, 1.5), mean_B=(-2,-2), sd_A=0.4, sd_B=0.4. And for assessment, we used the ratio of misclassified elements.
We noticed that the results are sensitive to the initialization of the weight matrix W, so the following results are conclusions of the aggregation of 100 training sessions.

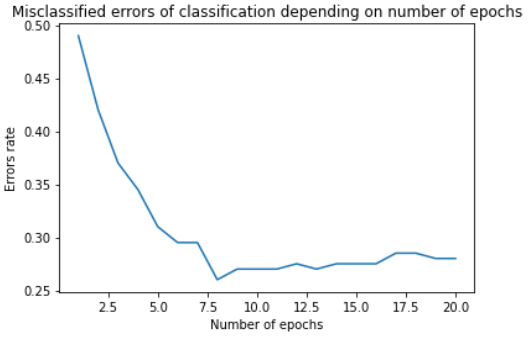
1. Both the Perceptron and the Delta rule learnings converge and separate well the data.
2. The convergence depends on the learning rate and the number of epochs.
3. On average, the sequential delta rule learning takes more time than the batch one.
4. For a fixed number of epochs 20, and on an average of 100 training sessions, the best learning rate for the perceptron learning was 0.001 (sd of 2.16). For the batch delta rule learning, it was 0.004 (sd of 0.002), and for the sequential one
5. To compare the convergence of the batch/sequential delta rule learning, we calculated the mean squared error for both learnings through epochs and with a fixed learning rate to see who converges faster.



It appears that the batch mode is faster in terms of epochs convergence.

Here the plots of the learning misclassified errors on 20 epochs for both the batch (left) and sequential (right) learnings:





6. After removing the bias, the perceptron only classifies the data samples separable by a line passing through the origin (0,0). So for example, two clusters with the means mA=(1,1) and mB=(3,3) won't get separated.

B) Classification of samples that are not linearly separable

After generating a non linearly separable data, we perform perceptron and batch delta rule learnings.

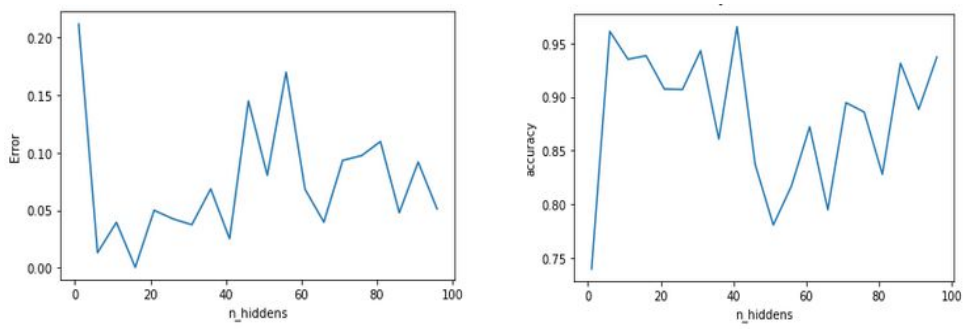
The perceptron is updating weights looking for a solution that doesn't exist so it stops after it reaches the maximum number of iterations. For the delta rule training, it always ends with an ineffective decision boundary misclassifying some points but minimizing the error.Now after using the dataset given in the lab, and performing the asked simulations on different subsamples of the data, and using the accuracy of each class as assessment. Indeed, we compute the accuracy of classification of the removed subsamples , i.e. we treat them as test sets to evaluate the generalisation of the learned model for each class and how the subsampling affected it. We reach to the following conclusions:

1. The generalisation of a class is negatively affected when the class is less represented.
2. An even representation of the classes involved in the training yield the best generalisation, i.e. the highest accuracies for the test data.
3. When a class training set is not representative of its real distribution, subsets of that class are not generalised.
4. Even classes and the fact that they should represent the real distribution of the data are necessary for a good generalisation.

1.2 Classification and regression with a two-layer perceptron

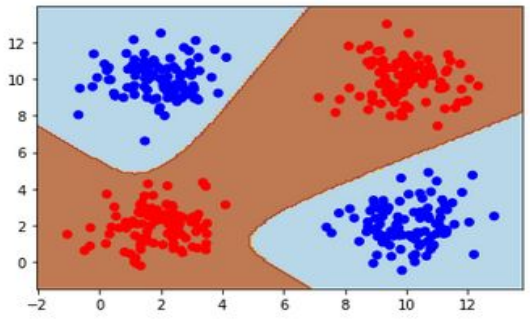
1.2.1 Classification of linearly non-separable data

In this part we will add a new hidden layer that uses a nonlinear activation function to deal with non linearly separable data. The training is based on backpropagation that is a generalization of delta rules used by the perceptron.



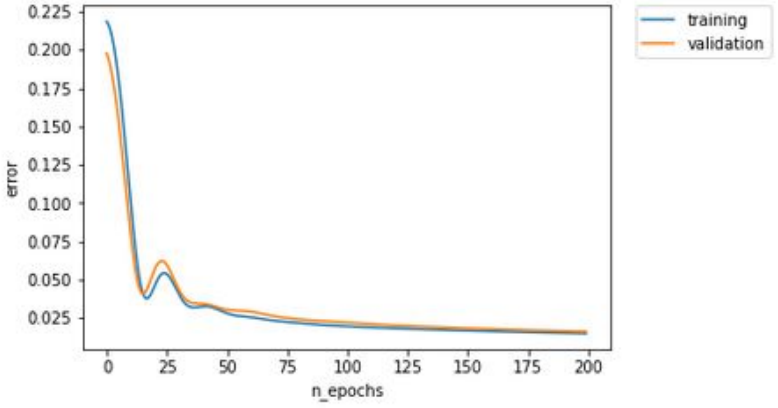
Let's start by seeing the effect of hidden nodes on the mean square error and the accuracy. By modifying the number of hidden nodes we notice that the accuracy and the error change considerably.The number of hidden nodes needed to perfectly separate a given data depends a lot on the structure of data. For a data generated from 2 gaussian distributions with the same parameter as stated in the pdf we need approximately 20 hidden nodes.

In the following figure we see that with a two-layer perceptron we are able to solve a problem similar to XOR which was impossible for a simple perceptron.

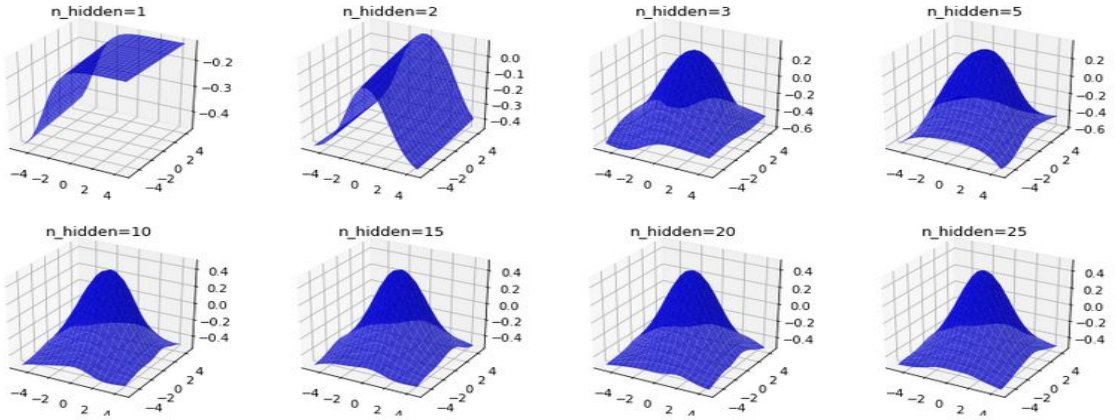


We notice that there is only a slight difference between the mean square error of training and validation sets while using the best **n_hidden** found in the previous question. That shows the capacity of our neuron network to generalize. MSE of both validation and training stabilize when the number of epochs is greater than 70 and they converge to 0.

There are several cases where the validation error is very different from the training error. The most obvious case is when we have overfitting for example by increasing the number of hidden nodes. The second scenario is when we remove a relevant part of the data from the training set and we put it in the validation set. For example the 4th sipliting scenario (as stated in the pdf) we put in the validation set 80% of the points which belong to class A and have a positive ordinate.



1.2.2 Function approximation



For n_hidden = 1 we approximate the gaussian function by a plane or a step function.

For n_hidden = 2 We approximate the function by a uni-dimensional function.

For n_hidden greater than 20 we notice that the MLP approximates the Gaussian function very well.

Then by increasing the number of hidden nodes the MLP learns can approximate more complex functions.

Above 20 hidden nodes there is no considerable changement. The best model is the one that is less complex and maintains a good approximation. Then the best model is the one with n_hidden = 20.

Rate: val/train	**Mean error training**	**Sd error training**	**Mean error validation**	**Sd error validation**
20%	0.00561	0.01083	0.00494	0.00936
40%	0.00780	0.00738	0.00926	0.00809
60%	0.01238	0.01885	0.01292	0.01940
80%	0.03714	0.03022	0.03674	0.02853

From the table we see that the error does not vary very much by changing the training / test ratio and it wasn't expected . Maybe because learning the Gaussian function is not very complicated for an MLP with 20 hidden nodes so it can easily generalize with only 20% of training.

We can speed up the convergence by increasing the learning rate. But we should make sure that it is small enough otherwise our MLP will never converge.

2 Results and discussion - Part II

2.1 Three-layer perceptron for time series prediction - model selection, validation

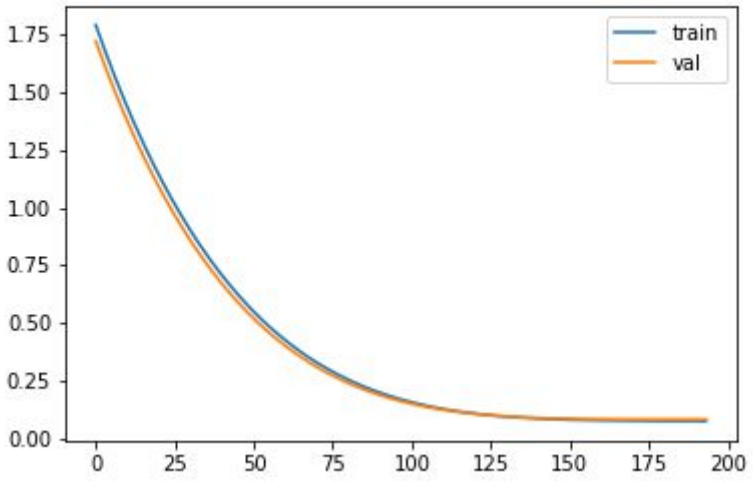
The network architecture used has the following properties:

- 2 hidden layers, with 8 and 9 units each (respectively).
- "sigmoid" is used in the hidden layers, and the "linear" identity function in the output
- the optimizer used is ADAM, with a learning rate l_r = 0.001 (a higher learning rate resulted in highly variant results)
- early stopping was used, with the objective to minimize the validation loss. Additionally, patience=10 which stops the early stopping from making the learning stop too prematurely
- The maximal number of epochs is 250, although the early stopping usually stops the

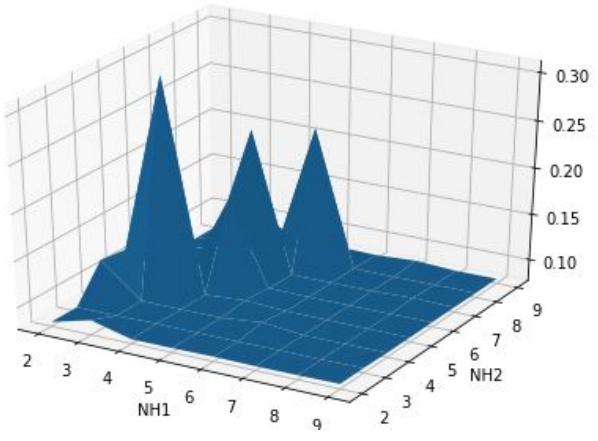
training at below 100 epochs.

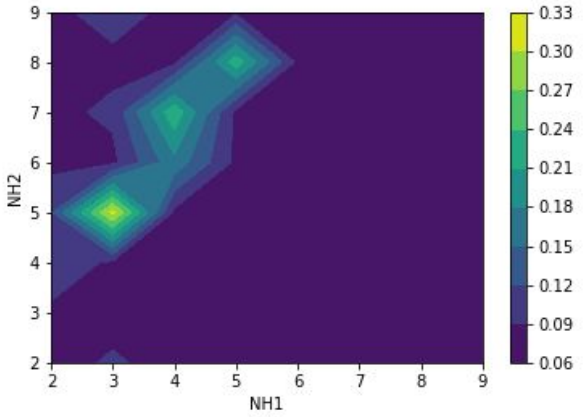
The evolution of the training loss and validation loss is for this network is as follows (x-axis is the number of epochs, y-axis the val_loss):

We notice the effect of early stopping, as the validation loss is at a plateau in the 160th epoch, it also seems that there is no premature stopping of the learning.



To select the best number of units in the 2 hidden units, we plot the following 3D figure, showing how the MSE varies with these parameters:

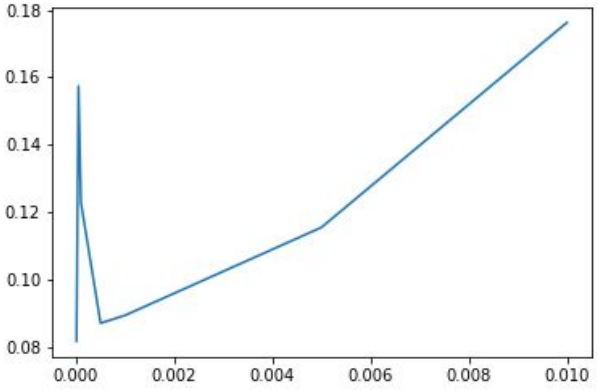




We conclude that the loss is particularly high in (3,5), and is generally high for values that have nh1<nh2. However, by choosing a value like (6,3), the loss is at a minimum.

2.2 Three-layer perceptron for noisy time series prediction with penalty regularisation

We experiment with the L2 regularization's different values of lambda:



The optimum is lambda = 0.0005 which results in minimum validation loss.

3 Final remarks

The lab was very instructive in the understanding of the construction of a perceptron and a Multi-Layer-Perceptron. Indeed, the implementation from scratch helped us understand the underlying principles and the role of some hyperparameters. We learned how the generalisation error depends on the learning rule, learning rate, whether it is in batch or sequential and some other parameters.

We also learned, through the second part of the assignment, the effect of the regularisation on the learning process and the generalization error.