DD2437 - Artificial Neural Networks and Deep Architectures

Short Report On Lab Assignment 1a

Classification with a single-layer perceptron

Alex Norlin, Athanasios Charisoudis and Giovanni Castellano

January 26, 2022

1 Main objectives and scope of the assignment

In this laboratory, we studied different methods to train a single-layer perceptron network on binary classification task. Such kind of neural network can be used to linearly separate a dataset into two classes.

Our major goals in the assignment were:

- to understand the main differences between the perceptron learning and the Delta rule.
- to study how the learning rate affects the two algorithms.
- to understand the difference between sequential and batch learning algorithms.
- to understand the role of the bias in the learning process.
- to study the behaviour of the algorithms using non-linearly separable data.
- to study the behaviour of the algorithms using unbalanced classes.

In order to easily visualize the results, we tested the algorithms on a two dimensional dataset. Moreover, we assumed to work with single output networks (i.e. one neuron in the output layer).

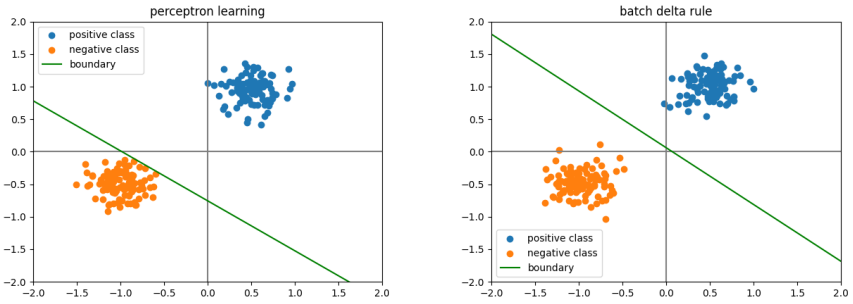
2 Methods

We decided to use the programming language Python to implement and test the two algorithms. Thanks to the library Numpy we could easily perform mathematical operations efficiently. Moreover, we used mathplotlib to plot the results.

3 Results and discussion

3.1 Classification with a single-layer perceptron

Initially, we drew two clusters of data points from two Gaussian distributions. Adjusting the values of the means and the variances we could generate two linearly separable clusters. Subsequently, we trained a neural network with two input nodes (plus 1 for the bias) and one output node to classify the data in the two clusters. In figure 1, we can observe the learned decision boundary of the network when trained with two different algorithms. As we can observe, both decision boundaries separate the clusters, however, the margin of the boundary generated by the Delta rule is much larger. This means that the network trained with the Delta rule is more likely to correctly classify unseen data.

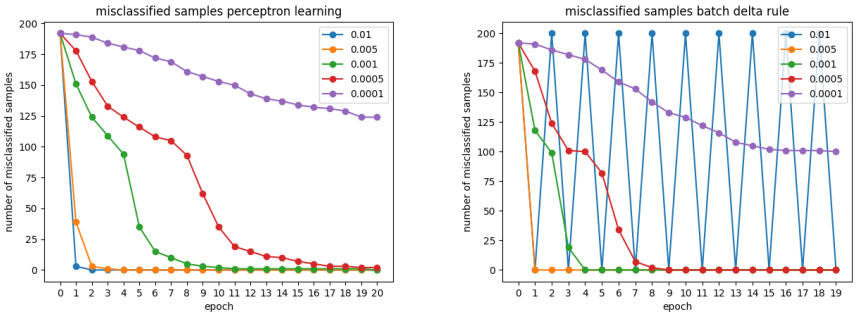


(a) (b)

Figur 1: (a) Perceptron learning decision boundary. (b) Batch Delta rule decision boundary.

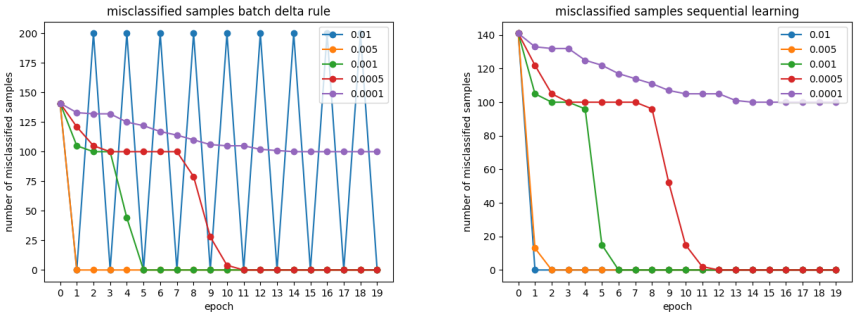
Both algorithms depend on a parameter called learning rate, which determines how quickly the weights vector W is updated. Figure 2 illustrates the number of misclassified samples during training for both algorithms and for different values of the learning rate. This accounts for the training error. We can observe that, in general, the higher the learning rate the faster the algorithms will converge. However, the Delta rule can become unstable and never converge if the learning rate is too high. It is worth mentioning, that the number of iterations (epochs) needed to converge also depends on the position of the data points and the initial conditions of the training (e.g. initial weights, ordering).

Figure 3 shows the differences between batch and sequential Delta rule in the training error learning curves. We did not observe any oscillations in the curves for the sequential Delta rule, however this does not mean that it cannot happen. Apart from this difference (i.e. if we ignore $lr = 0.01$ curve for the batch Delta rule), the plots look very similar. It is interesting to study how sensitive these algorithms are to different initializations of the network; Figure 4 illustrates the number of misclassified points for five different initializations and for a fixed



(a) (b)

Figur 2: Number of misclassified points for different values of the learning rate. (a) Perceptron learning. (b) Batch Delta rule.

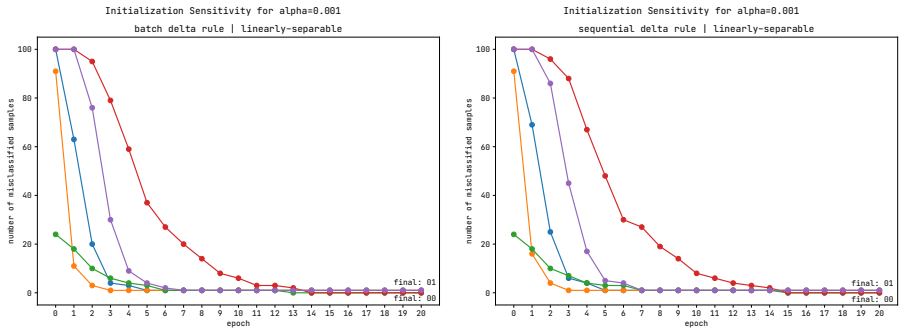


(a) (b)

Figur 3: Number of misclassified points for different values of the learning rate. (a) Batch Delta rule (b) Sequential Delta rule.

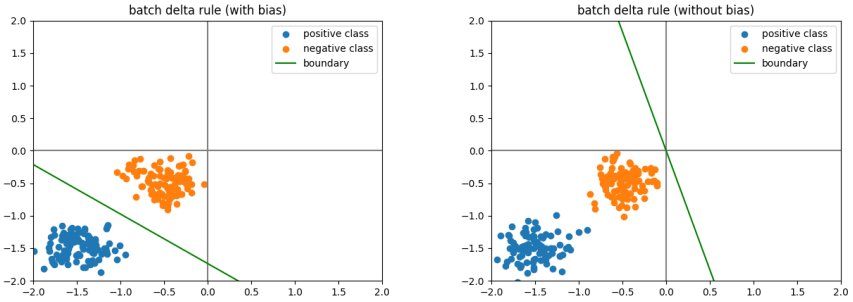
value of the learning rate. Also in this case the plots look very similar, and both sequential and batch learning are sensitive to the initial condition but to a rather small extent. However, the two plots could appear very different if we work with a multilayer network; we know in fact, that the stochastic gradient descent algorithm could be able to escape local minimum. In the case of single layer networks, the cost function is convex, therefore there is only one minimum. In general moreover, we know that sequential learning algorithms are sensitive to the order of the data.

Finally, we studied the behavior of the Delta rule with and without the bias term. Figures 5 shows the learned decision boundaries with the networks trained without the bias term; they are not able to separate clusters the means of which are placed in the depicted positions. This happens because without the bias the decision boundary is forced to pass through the origin. In other words, the bias determines the translation of the line, whereas the rest of the weights in the network determine the slope of the line.



(a) (b)

Figur 4: Number of misclassified points for different initializations of the W vector. (a) Batch Delta rule (b) Sequential Delta rule (learning rate = 0.001).



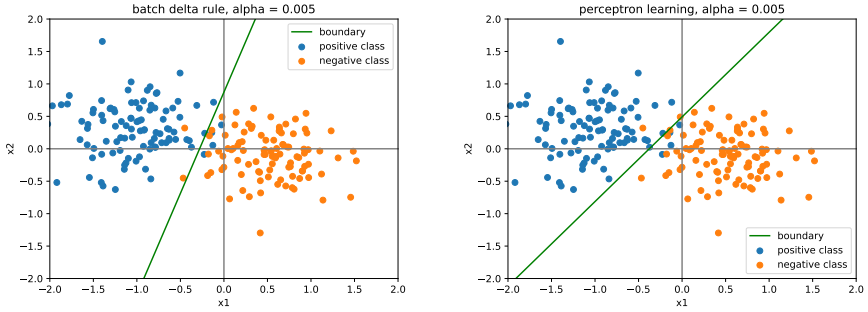
(a) (b)

Figur 5: (a) Batch Delta rule decision boundary using bias. (b) Batch Delta rule decision boundary without using bias.

3.2 Classification of data that are not linearly separable

The result of repeating the first part of 3.1 this time with data that isn't clearly linearly separable (i.e. the supports of the Gaussians have some overlap) can be seen in figure 6. Both learning methods found a suitable solution in most cases. For the Delta rule it failed to find a good solution when the learning rate was equal to 0.01 while no oscillation was observed in the perceptron for any value of the learning rate. As we saw in the case where the data was linear, the Delta rule seem to generalize better than the perceptron.

As the final experiments, the results of subsampling our data -generated by the given script in 3.1.3- in four different ways before the training can be seen in figures 7 and 8. The different sampling scenarios were: we removed 1) 25% of points from each class, 2) 50% from class A, 3) 50% from class B, 4) 20% from a subset of class A where the first input was less than 0 and 80% from the rest



(a) (b)

Figur 6: Decision boundaries and data points. (a) Batch Delta rule. (b) Perceptron learning.

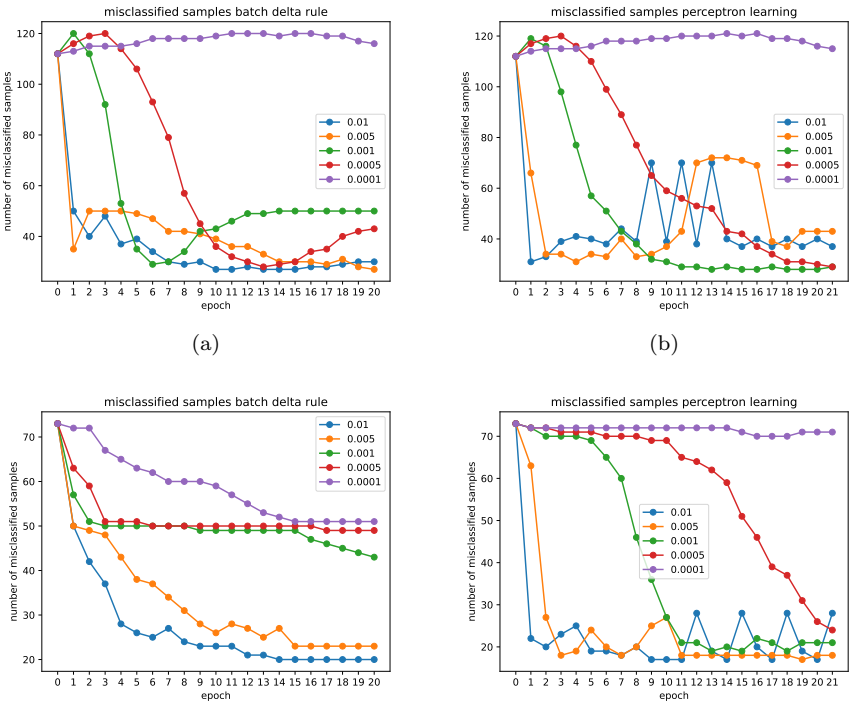
of class A. The plots for scenario 1 is not included in figure 7 since the decision boundary looked almost identical to the boundary of the non subsampled data. Figure 8 shows the decision boundaries instead of the misclassifications because it best illustrates the problem with this sampling.

Both models seemed to perform poorly in the sense that it's hard to get a good classification of this data with linear methods. The decision boundary was identical when an equal amount of points were removed from each class. In scenario 2 & 3 the decision boundary moved closer to the cluster from which points were removed without rotating for the best of the solutions we obtained (i.e. tried to correctly classify more of the non-removed class samples). This should result in a poorer generalization. The times the solution differed substantially were in scenario 2 or 3 where it sometimes classified all points as belonging to the class with the larger sample count. This can be seen in figure 7 (c) and (a) where for a number of epochs the number of misclassified points are constant 50 for some values of the learning rate.

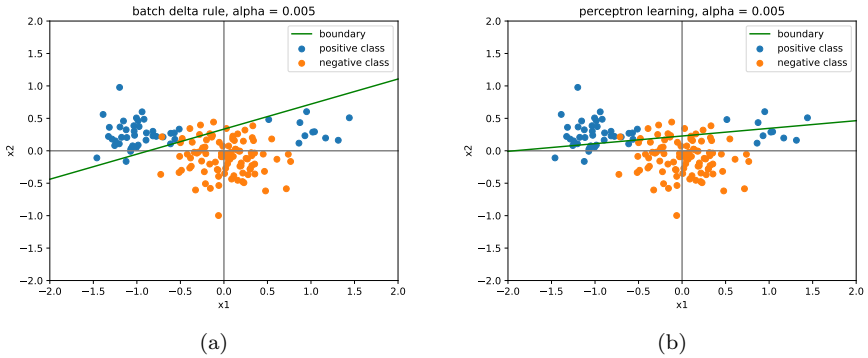
Perhaps the most important result was from scenario 4. This skewed the data into misclassifying the 10 points in the smaller cluster at a much higher rate. This means that it would generalize the worst to our original data.

4 Final remarks

The lab was straightforward for the most part. In some parts it were a little bit harder to know which plot to include since we only got 6 pages to argue for our points. In the accompanying presentation we also include MSE plots, were it is visible that the Perceptron learning rule is not optimizing MSE whereas the Delta rule continues training even after having found an initial solution (for solvable problems). What is clear, though, is that both algorithms are sensitive to initial conditions but Delta rule is very sensitive to the learning rate.



Figur 7: Number of misclassified points for different values of the learning rate for the different subsampling scenarios for both the Delta rule and perceptron. (a) & (b) are from scenario 2, (c) & (d) scenario 3.



Figur 8: Decision boundary and data points for both the Delta rule and perceptron in scenario 4. (a) Batch Delta rule (b) Perceptron learning.