Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 1

Classification with a single-layer perceptron

Hasan Alzubeidi, Rakin Ali and Steinar Logi

January 22, 2024

1 Main objectives and scope of the assignment

The main objective with this assignment is to familiarize ourselves with the perceptron learning rule and the delta learning rule along with batch versus sequential learning method. Our major goals in the assignment were

- Implement and explore the perceptron learning rule for simple classification tasks
- Explore how the perceptron and delta learning rule performs on linearly separable data compare to a dataset that is not linearly separable
- Explore how the bias affects the convergence of the delta learning algorithm

This assignment was scoped on the delta and perceptron learning rule. For the delta rule we used batch learning and sequential learning; for perceptron learning rule, we only used sequential learning. The limitations was that we didn't test around a lot with different learning rates, we simply one the ones that seemed reasonable.

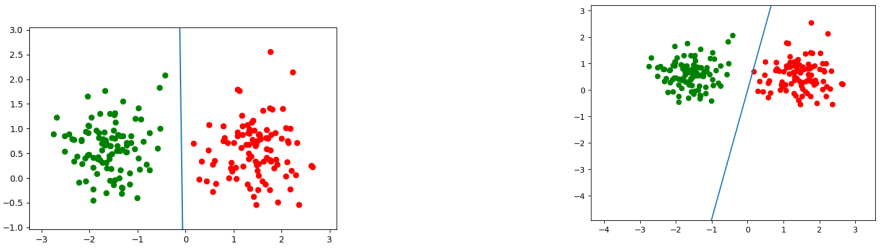
2 Methods

This lab was conducted three separate times as we did not collaborate when writing the code. We all did the code separately then compared our results. The code was written in Jupyter notebook and regular python v3.9 file. We all drew more or less the same conclusions based on our results however got different graphs. Numpy function and matplot were the only libraries used.

3 Results and discussion

3.1 Classification with a single-layer perceptron

Perceptron versus Delta Rule Learning: In this Section we compared two learning algorithms with a single perception. The algorithms compared were delta rule and perception learning. From the lectures we learned that perceptron learning converges to the "first best solution" it finds however that result does not generalize well compared to delta rule. This is evident in figure 1 as the boundary line generalizes better. Regarding convergence, delta rule converges faster as a lot less iterations were needed.



(a) Delta rule (b) Perceptron rule

Figure 1: Decision Boundary for Delta rule and Perceptron Rule

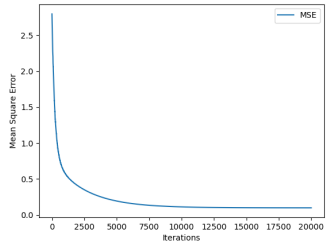
Removing Bias check: we also removed the bias to check how well delta rule converges. This can be seen in figure 3 The decision boundary passes through

(a) Delta rule with 8 iterations (b) Delta rule with 100 iterations

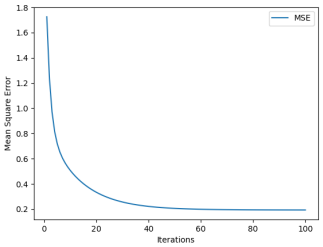
Figure 2: Decision Boundary for Delta rule without bias

origo regardless if it is not the optimal solution. The slope changes to become as optimal as possible. The bias simply changes the "starting" position of the slope which makes it easier to find the optimal decision boundary.

Batch versus sequential learning: Lastly we compared sequential versus batch learning. They converge to the same result however batch makes the changes faster after each iteration. It "stores" all the changes after each epoch then adds the changes while sequential ad



(a) Delta rule with sequential learning

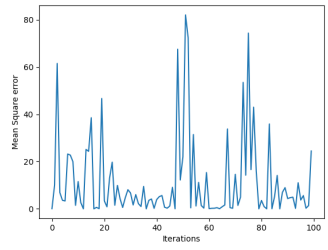


(b) Delta rule with batch learning

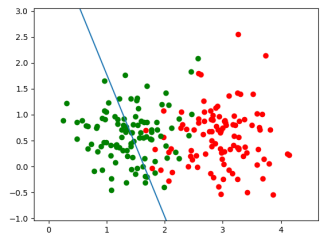
Figure 3: Mean square error comparison of online and sequential learning

3.2 Classification of data that are not linearly separable

Perceptron training: We created a dataset that is not linearly separable by using the same method as in the assignments above, moving the means of the two clusters closer together. First we tried fitting the single layer Perceptron using the Percepton learning algorithm. We expected that this would not work since it is only guaranteed that it converges if the dataset is linearly separable. Our simulations confirmed this. We tried fitting this Perceptron using the Perceptron learning rule. We calculated the mean square error in the same way as is done when using the Delta learning to get a glimpse into the error of the model. The model does not converge.



(a) Training error when using Perceptron

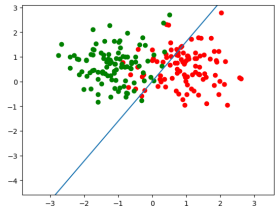


(b) Decision boundary

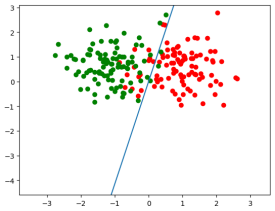
Figure 4: Error rate and Decision boundary of Perceptron Learning

The training error never reaches a point where it is stable. This makes sense since there will always be a point that is misclassified. When the training

algorithm reaches that point it moves the boundary in a such a way that it will be on the correct side of the boundary. That will make another point misclassified and therefore this boundary will always keep on moving to make the current point correctly classified. An illustration of this can be seen below.



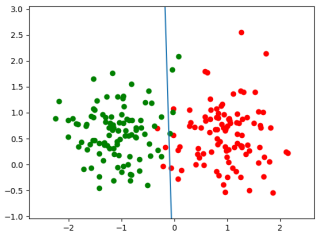
(a) Decision boundary after 10000 iterations



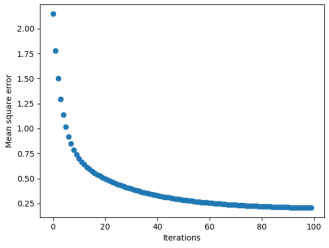
(b) Decision boundary after 11000 iterations

Figure 5: Two different decision boundaries obtained with the perceptron learning

Delta Rule Training: We used a single layer Perceptron, trained using the Delta learning rule to classify the same dataset seen in figure. We expected the learning algorithm to converge when using an appropriate learning rate. The algorithm did converge and we obtained the decision boundary shown in figure 6. The other picture shows how the training error developed through the training. It decreases fast in the beginning until it reaches convergence.



(a) Decision boundary on data set that is not linearly separable with Delta rule



(b) Delta rule with 100 iterations

Figure 6: Decision boundary and MSE using the delta rule

We created another kind of dataset that is not linearly separable. It was created in such a way that the data belonging to one of the classes was generated the same way as before, i.e. a multivariate Gaussian with a single mean and variance. The other class was created in such a way that half of the data points belonging to that class were generated using the mean mA and the other half generated using the mean $-mA$. We then trained the perceptron using the delta learning rule and obtained the decision boundary shown in figure 7a. We can see that the delta learning rule does converge but the mean square error is a lot

higher than for the previous data sets. And the accuracy obtained on this data set is 0.775

(a) The new data set

(b) Delta rule with 100 iterations

Figure 7: Decision boundary and convergance

Excluding data: We then tried a few ways to exclude data from the set of data that the model is trained on. The decision boundaries obtained for each of the scenarios can be seen in figure 8. We used the data that was left as unseen data to test the accuracy. The scenarios are the following:

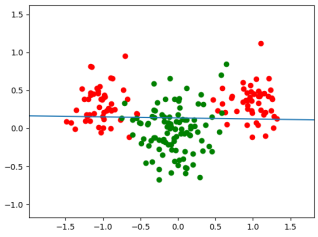
1. Random 25% from each class
2. Random 50% from class A
3. Random 50% from class B
4. 20% from a subset of classA for which $\text{classA}(1,:) < 0$ and 80% from a subset of classA for which $\text{classA}(1,:) > 0$

The accuracy of the training data in scenario 1 was 81% for class A and 81% for class B. The accuracy on the test data was 80% for class A and 72% for class B. This shows some form of generalization

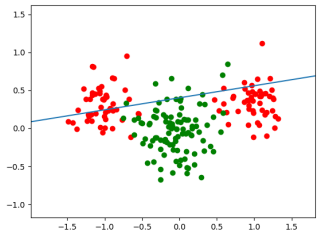
Scenario 2 had interesting results. The accuracy on the training set was 36% for class A and 93% for class B. The accuracy on the test set was 28% for class A. The training set is unbalanced and the wrongly classified points in class A do not affect the loss function which the delta function minimizes. Therefore accuracy is lower for class A.

Scenario 3 was very similar to scenario 2 but the other way around. The perceptron classified 97% of the points in class A correctly but only 28% of the test set points in class B.

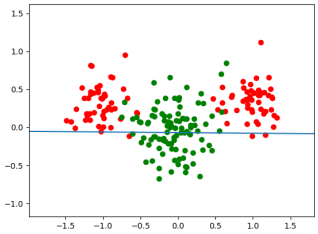
We found scenario 4 to be the most interesting one. The training accuracy was 76% for class A and 96% for class B. But the test accuracy was only 18% for class A. Since we removed most of the points in the right cluster of points belonging to class A and only a few from the left cluster, the perceptron benefits from classifying more points from class B correctly.



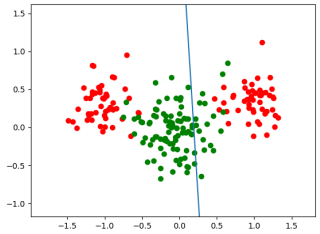
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Figure 8: Decision boundaries with different data removed

4 Final remarks

In this lab we implement the theoretical concepts practically. A single perception was created and trained using two learning algorithms being delta rule and perception training. What we learned was:

- The bias term matters. Removing it affects the decision boundary significantly and can decrease the performance of the perception.
- Initialization matters. We had to find different learning rules and continuously asked ourselves *is this result reasonable?* Many times it was due to the learning rate being too high or low, too few iterations and other stuff that mattered.
- Delta rule in most cases seems to be the better option. It produces a better decision boundary, converges somewhat faster than perception learning and generalizes better.
- Sequential versus batch should not matter much when dealing with small datasets. We did not notice a significant difference in terms of results other than the iterations were much higher for sequential. For large datasets, batch would have speed up computations significantly.
- Which data points you use for training and testing matters. If you have "too much" of point A compared to B then you'll notice overfitting on point A in your test data.