Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 1

Classification with a single-layer perceptron

Stefan Christiansson, George Malki and Rasmus Hammer

30 januari 2022

1 Main objectives and scope of the assignment

Our main objectives for this assignment were:

- To implement an on-line version of the single-layer perceptron using the perceptron learning rule and delta rule
- To implement a batch version of the single-layer perceptron using the delta rule.

By completing the prior objectives we wished to learn:

1. How the perceptron rule compares to the delta rule on the online version of the single-layer perceptron by varying the learning rate to study convergence and perform MSE comparisons between the two (with linearly separable data).
2. How the online version using the delta rule compares to the batch version with the delta rule by adjusting the learning rate and MSE performance (with linearly separable data).
3. How bias affects both the online version and batch version of the single-layer perceptron (with linearly separable data).
4. How the batch version performs on a non-linearly separable dataset by generating a non-linearly separable dataset and running the batch version on different subsamples.

The scope of this assignment is the implementation of the single-layer perceptron with a focus on the classical perceptron and the delta rule learning algorithms. The assignment was also coded using python since it's more used within the Machine Learning community than the MATLAB recommendation.

2 Methods

In this lab the perceptron algorithm was implemented, using both the classical perceptron rule, as well as the delta rule in both sequential mode, i.e. sample by sample, and batch mode. For the perception rule the learning rule used a bipolar representation (-1/1), which was compliant with the labels of the data generated when using the perception rule. The activation function using the classical perceptron rule can be defined as:

$$f(x) = \begin{cases} 1, & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ -1, & \text{otherwise} \end{cases}$$

where $\vec{w} \cdot \vec{x}$ is the dot product $\sum_{i=1}^{m} w_i x_i$, where m is the number of inputs to the perceptron and b is the bias. Further the weights for the classical perception rule are updated according to:

$$w_i(t+1) = w_i(t) + \eta \cdot (d_j - p_j(t))x_{ji} \;\; for \;\; all \;\; features \;\; 0 \leq i \leq n$$

where η is the learning rate, $w_i(t+1)$ is the weight i at time t, d_j is the desired value (or target value) of the output of the perceptron for the input, $p_j(t)$ is the actual predicted value at time t, and x_{ij} is the value of the i:th feature of the j:th training input vector. The other way that the perception was trained was using the delta rule, defined in matrix form by:

$$\Delta W = -\eta(W\bar{x} - \bar{t})\bar{x}^T$$

When using the delta rule the same activation function was used, which was compliant with the labels for the data generated when using delta rule. The training was conducted using all three variants (perception, sequential delta, batch delta), for learning rates 0.1 and 0.01, and 0.001 and with 1, 10, 30, and 50 epochs. To measure the difference in performance between the perceptron learning rule and the delta rule the mean square error (MSE) was used.

We used python as the programming language to implement the perceptron algorithm with perceptron learning rule and delta rule. Visual studio code was used as text-editor, and Git was used for code-sharing. For the plots we used the matplotlib library.

3 Results and discussion

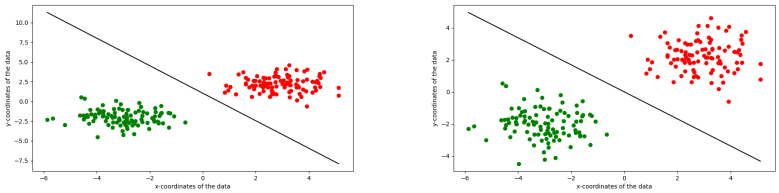
3.1 Classification with a single-layer perceptron

The results show a significant increase in performance using the sequential delta rule in comparison to using the classical perception learning rule, at lower learning rates, as can be seen in Table 1. For learning rate = 0.1, the results seem to indicate that sequential learning fails in performance, by being unable to converge towards a solution. Moreover perception rule seems to converge faster than delta rule in sequential mode, but often with a sub-optimal solution. This can be observed in Figure 2.

	Perceptron rule MSE	sequential delta rule MSE
Lr = 0.001	0.963	0.439
Lr = 0.01	0.512	0.150
Lr = 0.1	2.369	$1.385e^{37}$

Table 1: MSE for perceptron rule and sequential delta rule, at learning rate 0.001, 0.01 and 0.1, with 50 epochs total.

When comparing the graphs of the perception rule and the sequential delta rule, it is possible to observe a larger margin for the boundary created by using the sequential delta rule, for smaller learning rates, which also can be seen by looking at the MSE in Table 1. Figure 1 shows the two boundaries on the same dataset. Figure 2 shows the graph of the MSE for percepron rule and sequential delta rule, which indicates that the MSE for delta rule continues to drop, and converges at around 20 iterations, while it for the perceptron rule stops dropping once the perceptron has found a solution that does not miss-classify any points.



(a) Online solution using perceptron rule (b) Online solution using delta rule

Figure 1: Converged online solutions using a learning rate of 0.01 and 30 iterations.

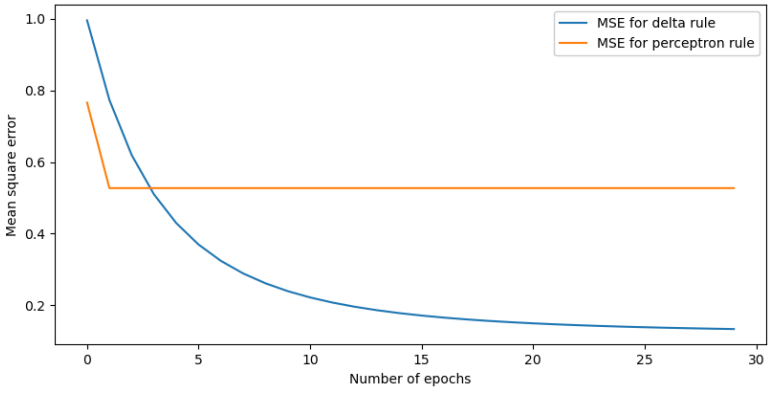
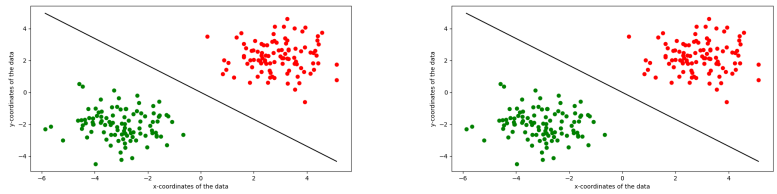


Figure 2: MSE comparison between perceptron learing rule and delta rule. Smaller values equal better performance.

When comparing the delta rule in sequential mode to the delta rule in batch

mode, it is possible to see that the batch delta rule generally converges faster than the sequential delta rule. Particularly for the learning rate 0.01 the results seem to indicate that the sequential delta rule converges at around 15-20 epochs, while it for the batch delta rule seem to converge at around 10-15. Moreover when the learning rate is increased batch delta rule seem to be able to converge towards a solution while sequential delta rule does not. However both methods seem to produce a very similar end result at lower learning rates. This can be observed in Figure 3.



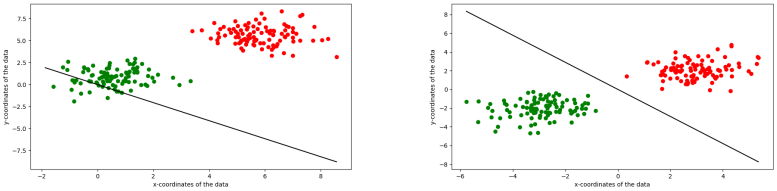
(a) Solution using delta rule in sequential mode

(b) Solution using delta rule in batch mode

Figure 3: Converged solutions using a learning rate of 0.01 and 30 iterations.

Looking at Figure 4a we can see that without the bias term we are unable to fit a line that separates data that isn't centered around origo. However, as long as it's centered around origo as in 4b we are able to fit a line that separates both classes. This is due to that the bias term gives the boundry transitional freedom in vertical direction due to the additional constant bias term that's added into the straight line equation shown in equation (1).

$$y = x_1 w_1 + bias \tag{1}$$



(a) Batch version run without bias on non-centered dataset

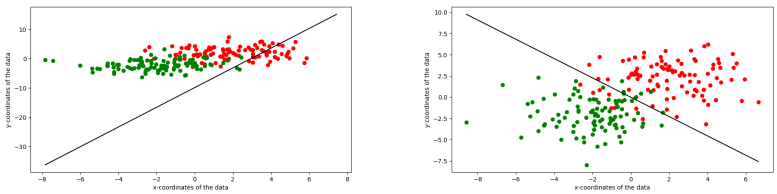
(b) Batch version run without bias on centered dataset

Figure 4: Solution for delta rule in batch mode without using bias on non-centered and centered dataset.

3.2 Classification of data that are not linearly separable

Figure 5 shows the solutions from using perceptron rule and delta rule in sequential mode on the non-linearly separable dataset. The results clearly indicate that perceptron rule is unable to find a solution, and thus wont converge. Delta

rule find a solution by trying to minimize the MSE, and is thus able to find a boundary that separates the data. This result can also be seen in figure 6 which shows the MSE for the perceptron rule and delta rule.



(a) Solution using the perceptron rule (b) Solution using delta rule in sequential mode

Figure 5: Solutions using percepton rule and sequential delta rule on non-linearly separable dataset.

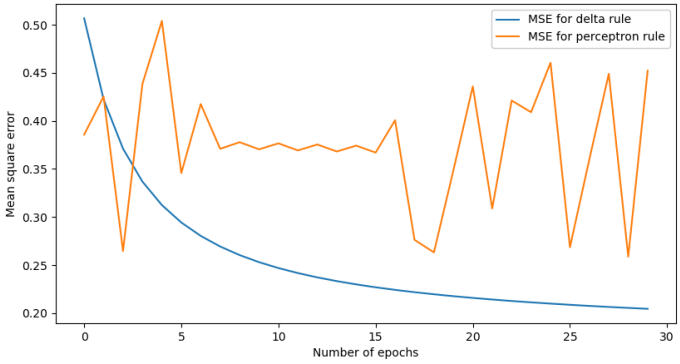
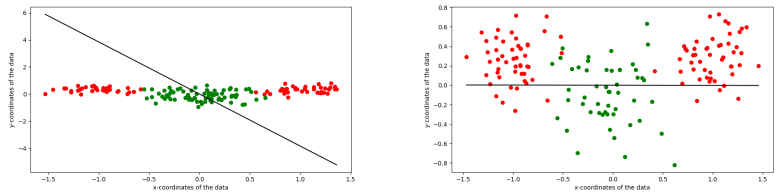


Figure 6: MSE for perceptron rule and delta rule in sequential on non-linarly separable dataset.

Figure 7 shows results from using delta rule in batch mode on some variations of a non-linearly separable dataset. In (b) in which 50% of the data from class B has been removed, the results show that the perceptron is able to find a solution that separates the data somewhat accurately.



(a) Data with 25% removed samples from each class (b) Data with 50% removed samples from class B

Figure 7: Delta rule in batch mode on non-linear dataset.

From the results it is possible to draw the following conclusions:

- The perceptron algorithm will converge using both the classical perception rule and delta rule in sequential or batch mode, if the data is linearly separable. However the performance of the model using delta rule will most likely be higher when classifying new data. This is due to the fact that the decision boundary that the delta-rule creates will have a wider margin.
- Delta-rule in sequential mode isn't able to converge with higher learning rates (≥ 0.1), because the weights are updated too fast. This would result in the algorithm not converging to a local minimum. There could also be the possibility that there were some errors in the code that made the perceptron using delta-rule in sequential mode not act as it should for higher learning rates. It seems reasonable that the first would be true, since the weights are updated for each sample in the data, and thus would be more sensitive to high learning rates.
- Delta-rule in sequential mode and batch mode perform similarly with the latter being more efficient.
- Using the perceptron rule it is not possible to find a solution for when the dataset is not linearly separable. Delta rule however converges to a solution that minimizes the MSE.
- The bias term gives the boundary transitional freedom, meaning that it is possible to classify data that is not centered around origo, which would not be possible without bias.

4 Final remarks

We found the lab to be interesting, and learned a lot about how the perceptron algorithm is implemented in practice. We were able to achieve a clearer understanding of why the delta rule is able to achieve a more intuitive and optimal boundary, by using the MSE to find a solution. Further we found it interesting to investigate how the learning rate affected the model, and the fact that delta rule in sequential mode seemed to be so sensitive to higher learning rates, if we did not make some mistake that made it behave it this way. It seems however reasonable that the delta rule in sequential mode would be sensitive to higher learning rates, as the weights are updated for each sample. This, and also the other parts of the results, also seemed to be consistent with the current theory. We found all parts of the lab relevant, however it was somewhat unclear what metric should be used when comparing the different models, or what would be the best way to test the performance of the model. The first part as well as half of the second part provided great insight in how the perceptron algorithm works in different scenarios, and why, while the rest did not contribute as much to our understanding, but seemed to just further emphasize that the perceptron algorithm does not work as well (or at all) for non-linearly separable data.