Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 1

Classification with a single-layer perceptron

Widad Farhat , Saly Al Masri and Xiaolei Liu

January 20, 2025

1 Main objectives and scope of the assignment *(ca. 0.5 page)*

Our major goals in the assignment were

- to implement and apply single-layer perceptrons with classical perceptron and Widrow-Hoff delta rule learning algorithms for classification tasks.
- to adjust several aspects of the architecture, algorithms and dataset of perceptrons and understand their influence to the performance and stability of models.
- to identify in practice and theory the core limitation of single-layer networks.

2 Methods *(ca. 1 page)*

The experiments are conducted in jupyter python3 environment, the libraries concerned are numpy, matplotlib for visualization, and also shuffle method in sklearn. We apply error rate on whole dataset as performance measurement.

3 Results and discussion

3.1 Classification with a single-layer perceptron *(ca.1.5-2 pages)*

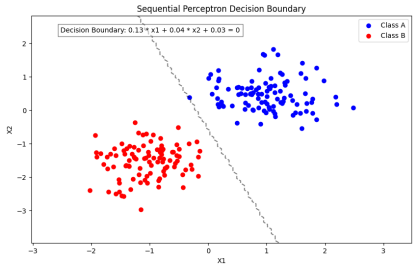
This section presents a detailed analysis of the results obtained, connecting the observed behaviors in the graphs to the mathematical principles discussed in the lectures. Each graph is analyzed in terms of its behavior, exact values, and its correlation with the theoretical foundations.

3.1.1 Sequential Perceptron and Delta Rule Decision Boundaries

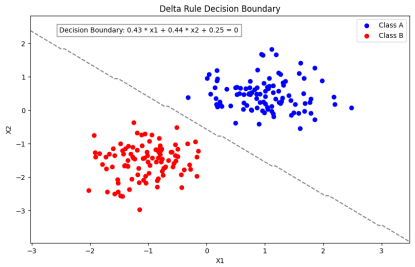
The decision boundaries for the Sequential Perceptron and Delta Rule algorithms highlight their capacity to separate two distinct classes, represented as red and blue points, in linearly separable datasets.

For the **Sequential Perceptron**, the algorithm iteratively adjusts weights based on misclassified points, eventually stabilizing to define a separating hyperplane. This behavior is consistent with the perceptron convergence theorem, which guarantees a solution for linearly separable data. By the final epoch, the model achieves perfect separation, with an error rate approaching zero.

In comparison, the **Delta Rule** refines its decision boundary by minimizing the mean squared error (MSE) using gradient descent. Unlike the perceptron, which targets misclassifications, the Delta Rule focuses on reducing the overall magnitude of errors, leading to smoother convergence over time. This approach results in a decision boundary that maintains greater separation between the classes, ensuring a more robust classification.



(a) Perceptron Decision Boundary ($w_1 = 0.13, w_2 = 0.04, \theta = 0.03$).



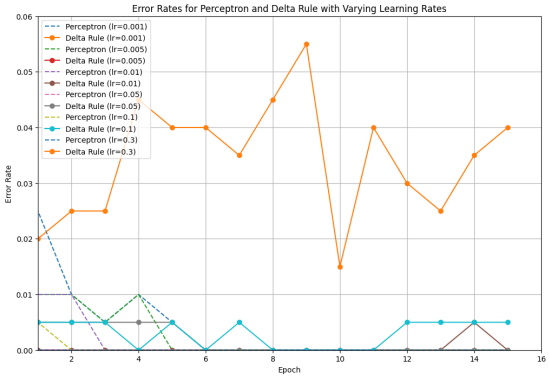
(b) Delta Rule Decision Boundary ($w_1 = 0.43, w_2 = 0.44, \theta = 0.25$).

Figur 1: Comparison of Decision Boundaries for Sequential Perceptron and Delta Rule. The perceptron converges quickly to a separating hyperplane, while the Delta Rule achieves smoother convergence and improved separation by minimizing MSE.

3.1.2 Error Rates for Perceptron and Delta Rule with Varying Learning Rates

The error rates for the perceptron and delta rule with varying learning rates highlight contrasting behaviors. For small learning rates ($\eta = 0.001$ and $\eta = 0.005$), both algorithms converge steadily, with the perceptron achieving zero error by epoch 4 for $\eta = 0.005$. As the learning rate increases ($\eta = 0.05$ to $\eta = 0.1$), the perceptron maintains lower error rates and continues to converge effectively, even with slight oscillations. At higher learning rates ($\eta = 0.15$ and $\eta = 0.3$), the perceptron remains stable and achieves minimal error rates despite the larger step sizes.

The delta rule, in contrast, exhibits increasing error rates as the learning rate rises. While it demonstrates steady convergence for smaller learning rates, its performance degrades significantly for $\eta \geq 0.1$, failing to maintain consistent learning. This is due to the gradient descent approach used by the delta rule, which becomes unstable with larger step sizes, as weights overshoot optimal values.



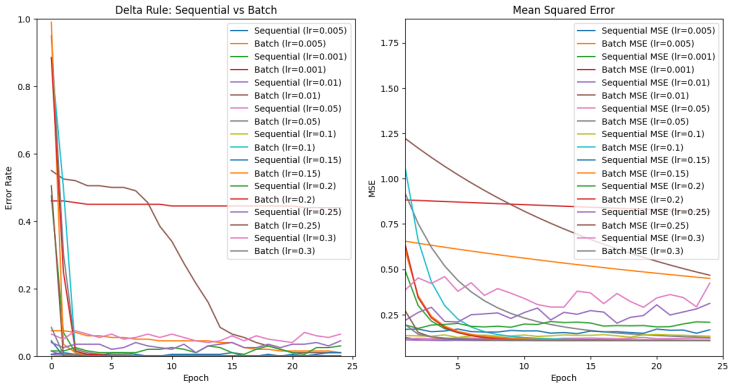
Figur 2: Error Rates for Perceptron and Delta Rule with Varying Learning Rates. The Perceptron maintains low error rates even at high learning rates, while the Delta Rule's error rates increase as learning rates rise.

3.1.3 Delta Rule: Sequential vs. Batch Learning

There is no fair comparison between the two, since in one epoch the later updates only once, while the former updates as much times as the scale of dataset. Symmetric problem lies in comparing error rate descending per update, since batch-processing perceptron has 'viewed' more samples.

As shown in figure 3 Sequential learning updates weights after each data sample, achieving rapid early convergence, such as reducing MSE from 0.2268 to 0.1196 by Epoch 2 with a learning rate of 0.01. However, it often leads to late-stage instability, with MSE oscillating around 0.1169. Batch learning, in contrast, updates weights only after processing the entire dataset, resulting in slower initial progress but smoother final convergence, like reducing MSE from 0.9632 to 0.4580 by Epoch 2 and reaching 0.5214 by Epoch 25.

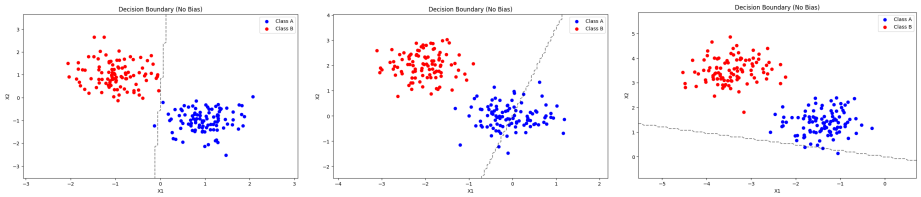
While comparing the behaviors of Delta Rule Perceptrons feature sequential and batch processing, we believe it is worth noticing that there is no fair comparison between the two, since in one epoch the later updates only once, while the former updates as much times as the scale of dataset. Symmetric problem lies in comparing error rate descending per update, since batch-processing perceptron has 'viewed' more samples.



Figur 3: Delta Rule: Sequential vs Batch Learning. Sequential learning shows faster initial convergence but higher variability, while batch learning exhibits smoother and more stable convergence.

3.1.4 Decision Boundary Without Bias

The absence of a bias term in the Perceptron restricts the decision boundary to always pass through the origin, limiting its ability to adapt to datasets not centered around it. as shown in figure 4 In Dataset 1 ($[1, -1]$, $[-1, 1]$), this constraint results in significant overlap between the classes, failing to provide an optimal separation. In Dataset 2 ($[0, 0]$, $[-2, 2]$), the misaligned boundary results in poor classification. Similarly, in Dataset 3 ($[-1.5, 1.5]$, $[-3.5, 3.5]$), the boundary fails to separate the shifted data. Experiments further illustrate this limitation: as classes are translated relative to the origin, the boundary either cuts through a class or fails to separate them entirely. This demonstrates the model's inflexibility and its unsuitability for asymmetrically distributed data.



(a) mA=(1,-1), mB=(-1,1) (b) mA=(0,0), mB=(-2,2) (c) (-1.5,1.5), (-3.5,3.5)

Figur 4: Btach Delta Rule Boundary when moving distribution centers roughly along the direction of their connecting line.

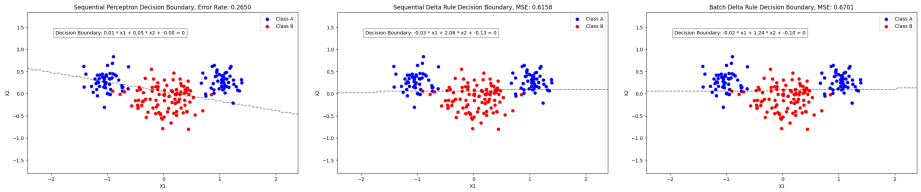
3.2 Classification of data that are not linearly separable

3.2.1 Classic Percertron and Delta Rule on not linear-separable data

Perceptron Sequential Learning:The Perceptron Sequential algorithm produced a decision boundary with a final error rate of 26.5%. Despite iterative weight adjustments, the perceptron could not perfectly separate the non-linearly separable data, as shown in Figure 1. The fluctuating error rates between 20% and 35% highlight the algorithm's instability and inability to handle non-linear relationships, confirming the limitations described in the lecture.

Delta Sequential Learning: The Delta Sequential algorithm yielded a decision boundary and a final MSE of 0.6158 (Figure 2). While the delta rule minimizes error and ensures smoother convergence, the linear boundary struggles to classify non-linearly separable data fully. The bias term (-0.13) adds flexibility, but error rates of 19% to 23% indicate the model's inherent limitations.

Delta Batch Learning: The Delta Batch algorithm generated a decision boundary, with a final MSE of 0.6701 (Figure 3). Batch updates average weight adjustments over all samples, resulting in a smoother but less adaptive learning process. While slightly less accurate than sequential updates, this approach also struggles with the data's non-linear structure.



(a) Sequential Perceptron Decision Boundary. (b) Sequential Delta Rule Decision Boundary. (c) Batch Delta Rule Decision Boundary.

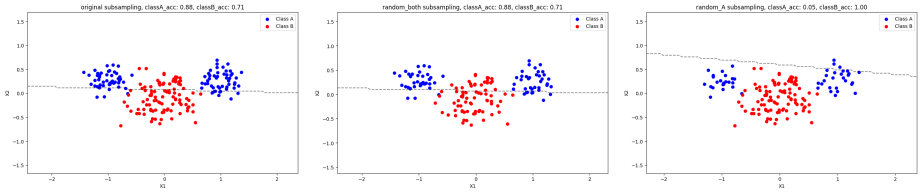
Figur 5: Comparison of decision boundaries and metrics for different learning algorithms. Sequential Perceptron struggles with non-linearly separable data, while Delta Rule (sequential and batch) shows slight improvement in alignment, but still lacks flexibility for complex data patterns.

3.2.2 Batch Delta Rule on sub-sampled not linear-separable data

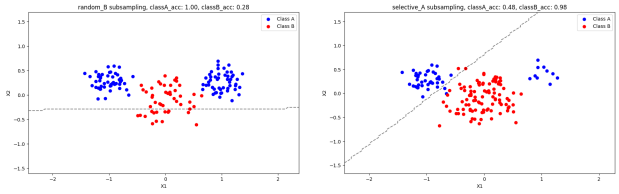
Though the instruction suggests using the dropped samples from down-sampling as test data, we opted for the full undivided dataset for testing. The classification boundaries and accuracy per class under different sub-sampling policies are shown in Figure 6.

Figure 6.(a) demonstrates that the one-layer perceptron struggles with non-linearly separable data, resulting in a compromised approximate decision bounda-

ry. Similarly, Figure 6.(b) shows that downsampling both classes does not significantly affect the boundary, as the perceptron remains unable to separate the classes effectively. This limitation is consistent across these scenarios, emphasizing the perceptron's inability to handle non-linear separability.



(a) Original samples (b) Both classes subsampled (c) class A subsampled



(d) only class B being subsampled (e) right fraction of class A subsampled

Figur 6: Batch Delta Rule Boundary under different subsampling policies.

In 6.(c) and 6.(d) it is clear that perceptron tends to preserve the integrity of one class (by deviating the boundary away from its samples) while the importance of the other class is underrated. The same applies to 6.(e), where the model ignores the right fraction of class A and yields the separating boundary between class B and left fraction of class A.
Finally, this graphical description is also reflected in accuracy per class, in which perceptron tends to achieve better performance on class that is not downsampled.

4 Final remarks *(ca. 0.5 page)*

In this lab we explored in detail some features of perceptrons in experiences. However, some further clarification of the instructions may be beneficial, for example the comparison standard between sequential and batch processing. Also, the instruction advises to apply dropped samples in sub-sampling as test dataset, this also sounds not reasonable enough in that the class distribution within it can be highly unbalanced and biased from the train dataset.