Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 1a

Classification with a single-layer perceptron

Nouhaila AGOUZAL, Abdessamad BADAOUI and
Nasr Allah AGHELIAS

September 1, 2023

1 Main objectives and scope of the assignment

Our major goals in the assignment were :

- to create a single layer perceptron and train it using perceptron learning rule and delta rule to separate linearly separable data.
- to compare the performance and accuracy of the created perceptron for different learning rules and hyperparameters on both lineraly and nonlinearly separable data.
- to evaluate single-layer perceptrons on subsamples of linearly non-separable dataset and demonstrate how data imbalance and non-representativeness can impact generalization.

2 Methods

The programming was done in python. We used the `numpy` library for basic mathematical calculations and `matplotlib.pyplot` to generate graphs.

3 Results and discussion

3.1 Classification with a single-layer perceptron

We have worked with data generated using a normal distribution with mean `mA` set to $[1.5, 1.5]$ and a standard deviation σ of 0.5 for class A, and mean `mB` set to $[-1.5, -1.5]$ and a standard deviation σ of 0.5 for class B.

The perceptron learning rule and delta rule both offer solutions to the problem, but the perceptron's solution can be less intuitive (with a narrower margin) because it stops learning as soon as it achieves 100% accuracy in classification. In contrast, the delta rule continues learning until it reaches the minimum of the loss function, which may be a local minimum and is entirely independent from the misclassification rate. This approach provides a more intuitive and robust solution.

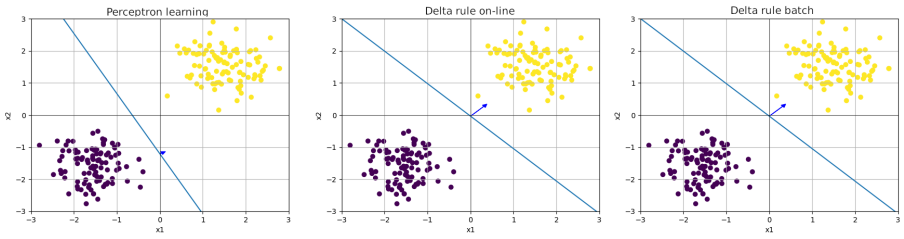


Figure 1: The final classifier of the perceptron across different learning rules

As the learned models are influenced by the initial weight settings, we conducted 200 distinct experiments to determine the required number of epochs for the classifier using both the perceptron learning rule and the delta rule across various learning rates.

Perceptron learning typically requires more training epochs to achieve convergence compared to the delta rule while keeping in mind that their definitions of convergence differ (perceptron: zero misclassification rate, delta rule: minimizing MSE). Moreover, when considering the delta rule with its two approaches (online vs. batch mode), the sequential update method notably outperforms batch mode, particularly when using small learning rates. See Figure 2.

The learning of our perceptron is sensitive to the random initialization of weights. If the initial weights happen to be very far from the potential solution, the process of learning will take much longer. This sensitivity is further amplified by the choice of learning rate. If the latter happens to be too small, the convergence of our model will become even slower. The smaller the learning rate the more sensitive our model becomes to random initialization.

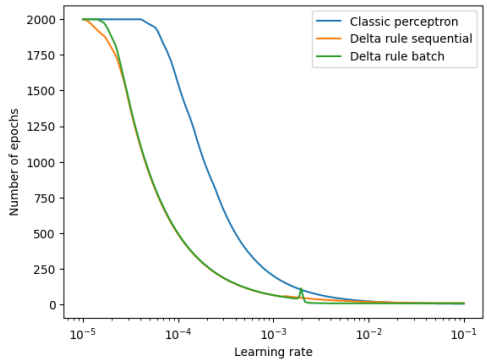


Figure 2: Number of necessary epochs for perceptron convergence depending on the learning rate for the three learning rules : classical perceptron learning, delta rule sequential and delta rule batch

Both the sequential and batch modes of the delta rule produce the same outcomes. This is expected because the Mean Squared Error (MSE) follows a simple curved shape with only one lowest point. However, it's important to highlight that the sequential update method doesn't exactly implement the true gradient descent. As a result, the MSE curve for sequential updates appears higher than the curve for batch mode, which does employ actual gradient descent, as shown in the figure 2 :

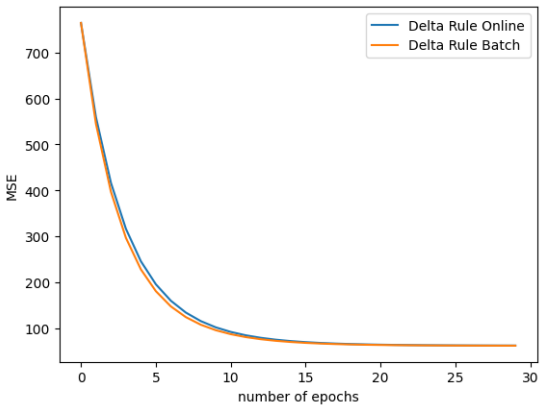


Figure 3: The mean squared error of the perceptron as a function of the number of epochs for the delta rule online and batch

The perceptron without bias converges and classify correctly all data samples when there is a solution that passes through the origin of the cartesian coordinate system.

3.2 Classification of data that are not linearly separable

We have worked with data generated using a normal distribution with mean `mA` set to $[1, 1]$ and a standard deviation σ of 0.8 for class A, and mean `mB` set to $[-1, -1]$ and a standard deviation σ of 0.8 for class B.

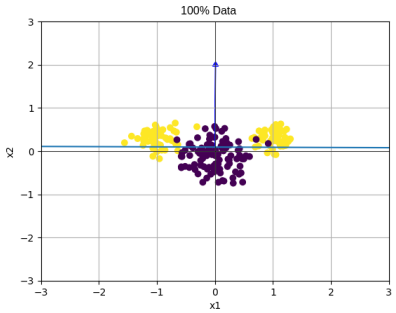
Primarily, perceptron learning fails to converge when dealing with non-linearly separable data because there will always be at least one data point that remains misclassified, causing the algorithm to continuously learn without stopping. In contrast, the delta rule consistently converges, provided that an appropriate learning rate is employed. This reliability arises from the fact that the learning process revolves around minimizing Mean Squared Error (MSE), which means that the algorithm is primarily concerned with reducing overall prediction errors and isn't overly concerned with individual data points that might remain misclassified.

When we decrease the amount of data in the first scenario when we randomly remove 25% from each class, we are actually preserving the same data distribution but with less data. Therefore, we should expect that the training error will decrease because with less data, there is less information available to adjust the model weights. In essence, we are increasing the variance of the model. Consequently, this increase in variance leads to a larger generalization gap, which negatively impacts the classifier's ability to generalize effectively. See Figure 5.

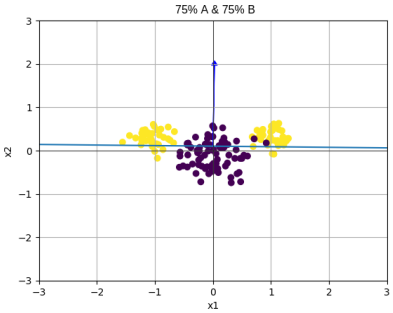
In the second and third scenarios, we create an imbalanced dataset when we remove 50% of one class. Consequently, the classifier tends to favor the more present class, which we can notice in the table displaying specificity and sensitivity. See Table 1.

In the fourth scenario, the data used for training no longer follows the actual data distribution (for class A). Consequently, the model will not be able to accurately capture the true underlying patterns and relationships present in the data. As a result, even if the model performs well during training, it will encounter difficulties when presented with new or unseen data, resulting in poor generalization performance.

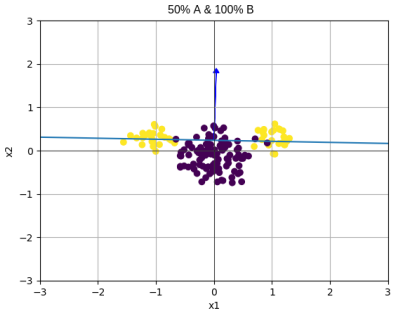
Here are the results obtained from resampling the data across all scenarios :



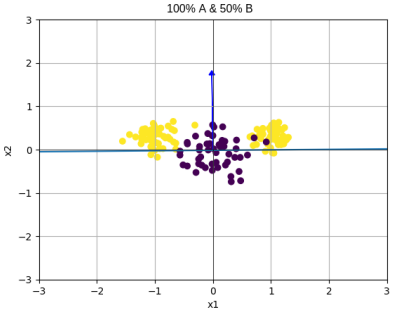
(a) 100% Data



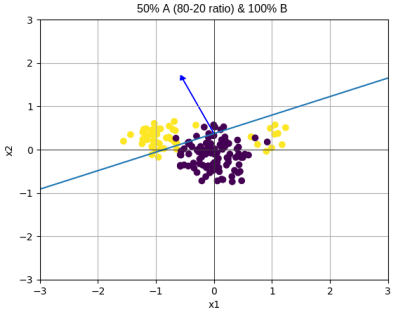
(b) 75% A & 75% B



(c) 50% A & 100% B



(d) 100% A & 50% B



(e) 50% A (80-20 ratio) & 100% B

Figure 4: The final classifier for the original data and its multiple subsamples

Subsamples	100% Data	75% A & 75% B	50% A & 100% B	100% A & 50% B	50% A (80-20 ratio) & 100% B
Sensitivity	0.88	0.92	0.62	0.95	0.74
Specificity	0.74	0.71	0.87	0.5	0.92

Table 1: Values of sensitivity and specificity for multiple subsamples of a non-linearly separable data

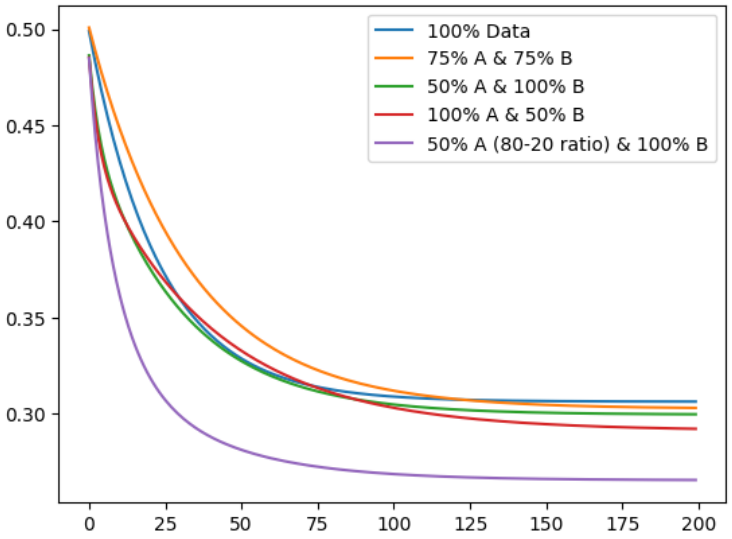


Figure 5: The mean squared error calculated across all the scenarios.

4 Final remarks

The main objective of the lab is to master the single layer perceptron and its multiple learning rules. Furthermore, we learned how these influence its convergence and performance.

The only confusing part about the assignment was the instruction describing the last scenario of subsampling. We understood at the end that one side should have most of the data compared to the other within the same class A but we took some time to understand the semantics.

We should note that the method we used to estimate the convergence of the perceptron trained using the delta rule (online or batch) is to calculate the difference of the mean squared error of 6 consecutive epochs and when all these differences are small enough we determine that the perceptron has converged.