Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 1

Classification with a single-layer perceptron

Tristan Perrot, Mathis Pernin and Romain Darous

September 1, 2018 (*fix the date)*

Please be aware of the constraints for this document. The main intention here is that you learn how to select and organise the most relevant information into a concise and coherent report. The upper limit for the number of pages is 6 pages for Lab 1a (more for other labs) with fonts and margins comparable to those in this template and no appendices are allowed.
All short lab reports should be submitted to Canvas by the authors as a team before the lab presentation is made. To claim bonus points the authors should uploaded their short report a day before the bonus point deadline. The report can serve as a support for your lab presentation, though you may put emphasis on different aspects in your oral demonstration in the lab. Below you find extra instructions in italics. Please remove them and use normal font for your text.

1 Main objectives and scope of the assignment *(ca. 0.4 page)*

Our major goals in the assignment were

- to generate linearly separable and non linearly separable 2D datasets,
- to implement single-layer perceptron classification, using perceptron rule and delta rule,
- to implement the delta rule using batch learning, with and without bias,
- to test the implementaion on linearly separable data and non linearly separable data with low and high variance.

The scope of this first lab concerns implementing single-layer perceptrons with focus on classical perceptron and Widrow-Hoff delta rule learning algorithms". As said before, we will use 2D data with binary labels for the perceptron rule and symmetric labels for the delta rule. We will see how this approach is mostly efficient for linearly separable data by looking at the results when it is not the case.

2 Methods *(ca. 1 page)*

We used the programming language **Python** and the programming environment **Jupyter Notebook**.
We used the python library **numpy** to perform the matrix computations and generate data.
Concerning the display of the plots such as dataset visualization and learning curves, we used the libraries **matplotlib, tqdm** and **IPython.display**. This allowed us to have a dynamic display of the algorithms, with a progress bar to observe the convergence of the separating line step by step.

3 Results and discussion

*Make effort to be **concise and to the point** in your story of what you have done, what you have observed and demonstrated, and in your responses to specific questions in the assignment. You should skip less important details and explanations. In addition, you are requested to add a **discussion** about your interpretations/predictions or other thoughts concerned with specific tasks in the assignment. This can boil down to just a few bullet points or a couple of sentences for each section of your results. Overall, structure each Results section as you like, e.g. in points. Analogously, feel free to group and combine answers to the questions, even between different experiments, e.g. with linearly separable and non-separable data, if it makes your story easier to convey.*

*Plan your sections and consider making combined figures with subplots rather than a set of separate figures. **Figures** have to condense information, e.g. there is no point showing a separate plot for generated data and then for a decision boundary, this information can be contained in a single plot. Always carefully describe the axes, legends and add meaningful captions. Keep in mind that figures serve as a support for your description of the key findings (it is like storytelling but in technical format and academic style.*

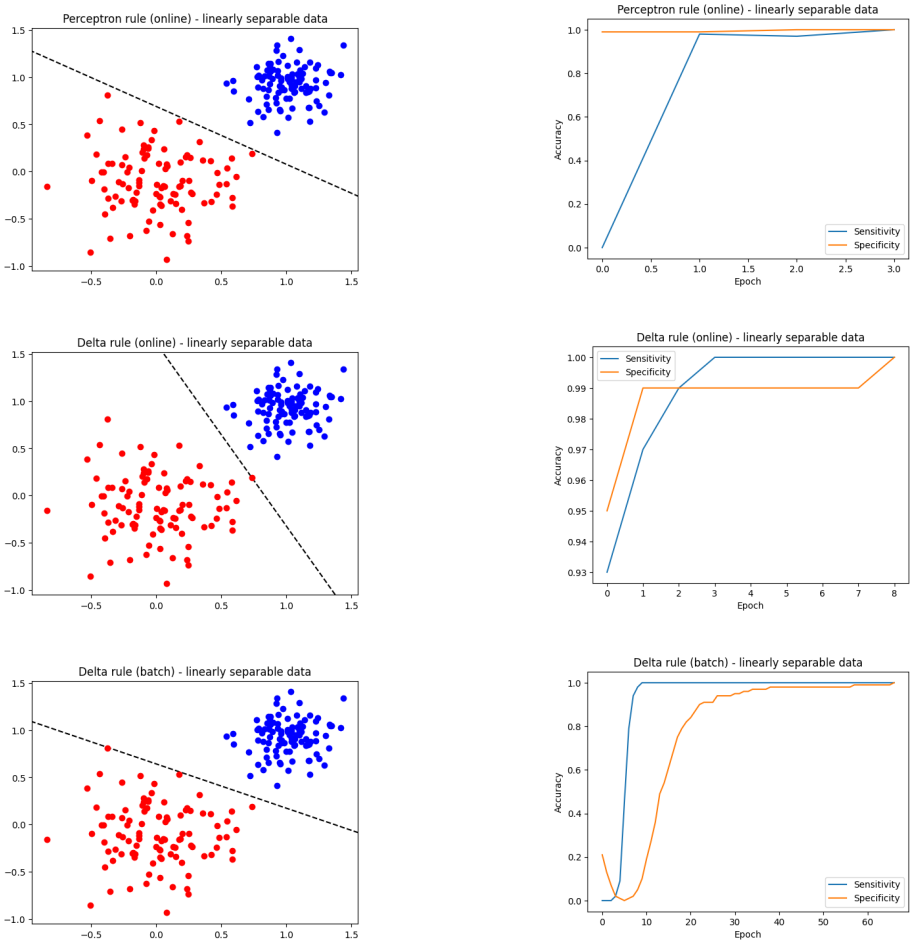
*Similarly, use **tables** to group relevant results for easier communication but focus on key aspects, do not overdo it. All figures and tables attached in your report must be accompanied by captions and referred to in the text, e.g. "in Fig.X or Table Y one can see ....".*

*When you report quantities such as errors or other performance measures, round numbers to a reasonable number of decimal digits (usually 2 or 3 max). Apart from the estimated mean values, obtained as a result of averaging over multiple simulations, always include also **the second moment**, e.g. standard deviation (S.D.). The same applies to some selected plots where **error bars** would provide valuable information, especially*

where conclusive comparisons are drawn.

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

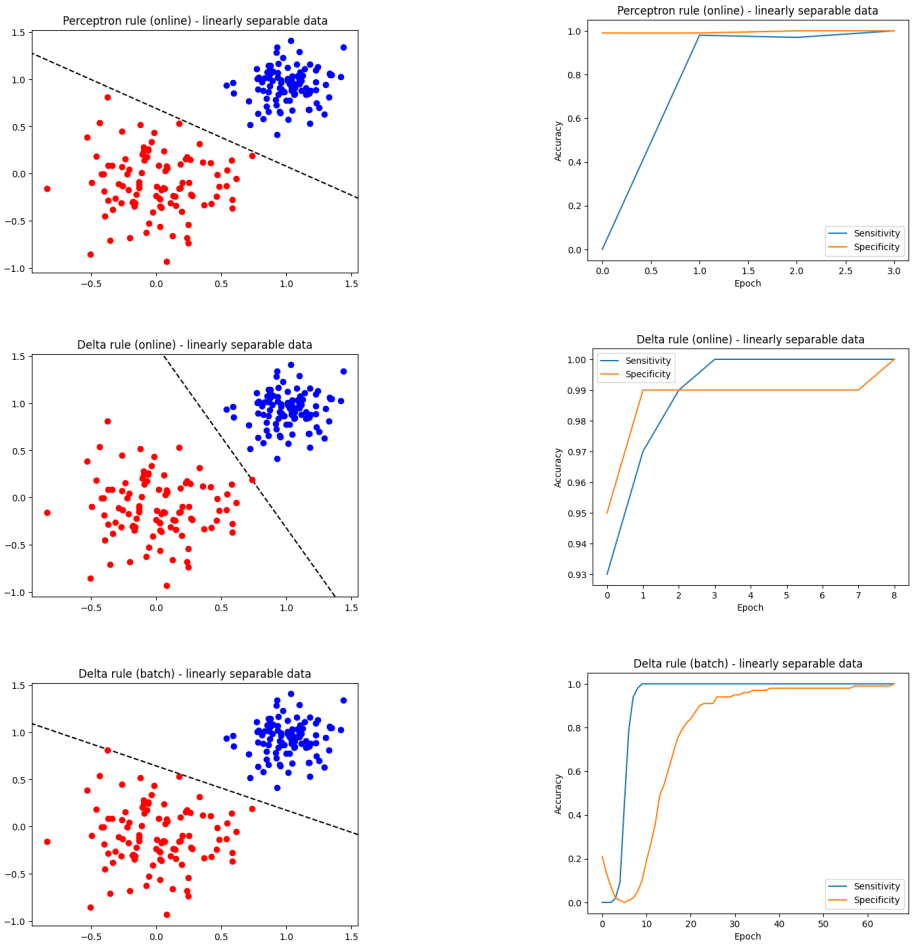
The next page shows the results of perceptron simulations using the perceptron rule, delta rule and delta rule using batch learning on linearly separable data generated according to the lab requirements :



Figur 1: Classification of linearly separable data

3.1.1 Convergence rate and speed

The perceptron rule converges after 4 epochs and the delta rule after 9 epochs. Using batch learning, number of epochs required to converge reaches 67, which is significantly higher. This can be explaine by the fact that for the batch learning method, the weights are updated only once per epoch.
The three methods reach the value of 1 for both sensitivity and specificity, which makes sense as the data are linearly separable.
One may notice that the convergence rate is higher for the perceptron rule and the online delta rule. It is smaller when using batch learning as we struggled to make it converge.



Figur 2: Classification of linearly separable data

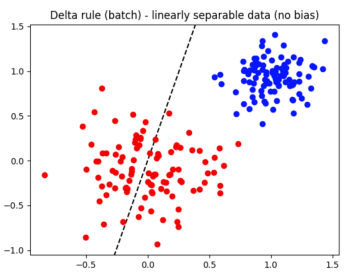
3.1.2 Results

The three methods provide a 100% accuracy over the training data set. However, the two obtained regions differ when implementing the delta rule, which can impact the generalization of the model. Indeed, perceptron rule and delta rule using batch learning have a rather similar separation line, but using batch makes the region of the red class smaller.
A test set would be required to evaluate which one of them generalizes the best.

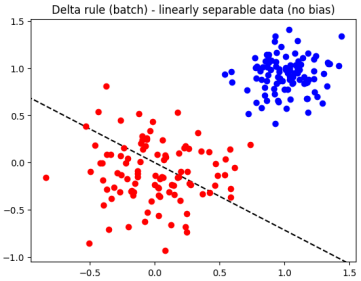
So far, delta rule using batch learning seems to be less efficient than the two other methods, without . We will see what happens when we use those algorithms on non linearly separable data.

3.1.3 No bias case

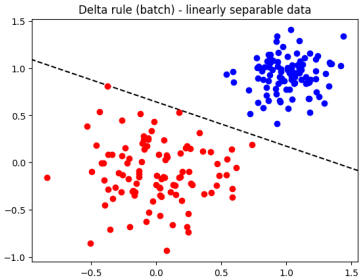
An interesting case to study is the case where we remove the bias of the parameters. Then, the algorithms allow to determine the orientation of the separating line, but it results in a linear line passing through zero, parallel to the result with bias. It fails to split the data as it is not centered around 0. The graphs below show the convergence of the no bias case, and a comparison with the bias case.



(a) First step



(b) Last step



(c) Batch rule with bias

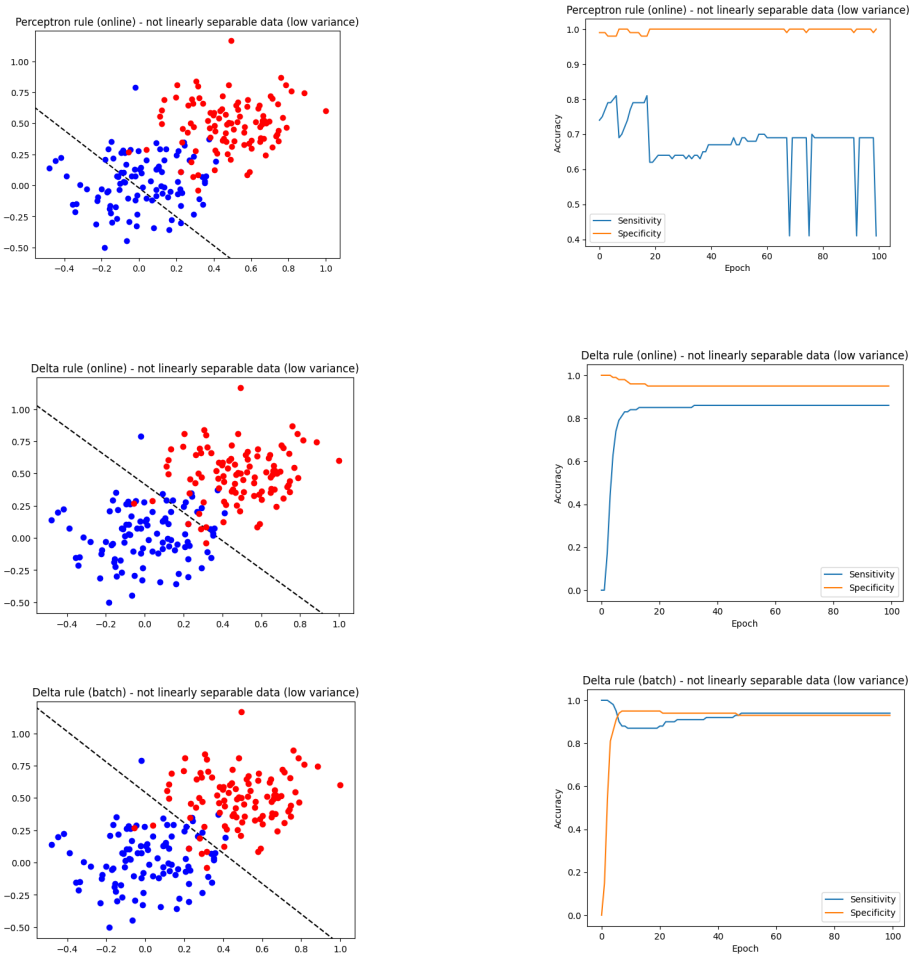
Figur 3: Classification of linearly separable data

3.2 Classification of data that are not linearly separable *(ca.1.5-2 pages)*

Here are displayed the results for three other datasets. One with a low variance, which makes it almost linearly separable, but with some outliers. Increasing the variance makes it less linearly separable. Finally, we will study a dataset made of split clusters, which makes it irrelevant to be considered as linearly separable.

3.2.1 Low variance data

Low variance data means that we generate a dataset which is mostly linearly separable, but with some outliers.



Figur 4: Classification of non linearly separable data with low variance

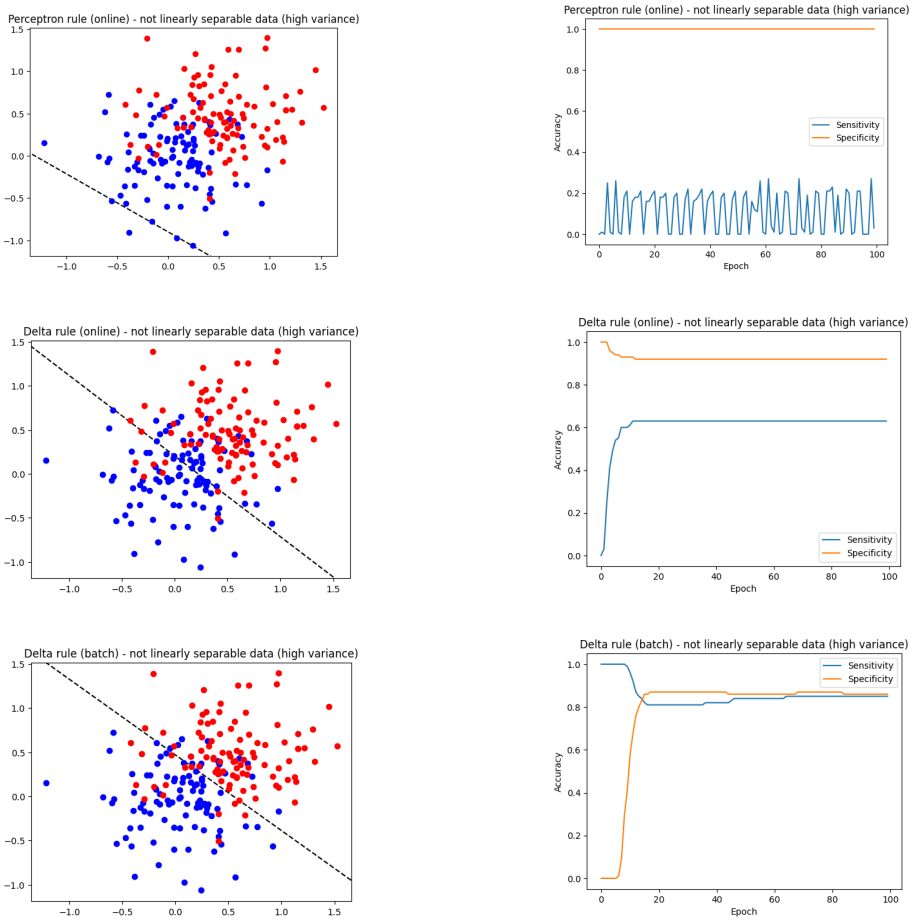
We can observe that using the perceptron rule, one of the two classes is fully on one side of the sepration line. It means that if a point is classified as being from the blue class, we will have confidence in the fact that it is indeed the case. It will be less accurate when a point is classified as red class, as even regarding the training data, there is a significant classifying error on this side of the separating line.

The delta rule leads to a more balanced distribution of the outliers. Most of the points are well classified and the separation line seems to distribute with more relevance the outliers between the two sides, splitting the plane in two areas where we can clearly identify the two clusters that we observe. The outliers are centered around the separation line.

The difference between the two methods is even more important when taking a look at high variance data.

3.2.2 High variance data

Increasing the variance of the data provides a dataset whose the separation is not clear anymore.



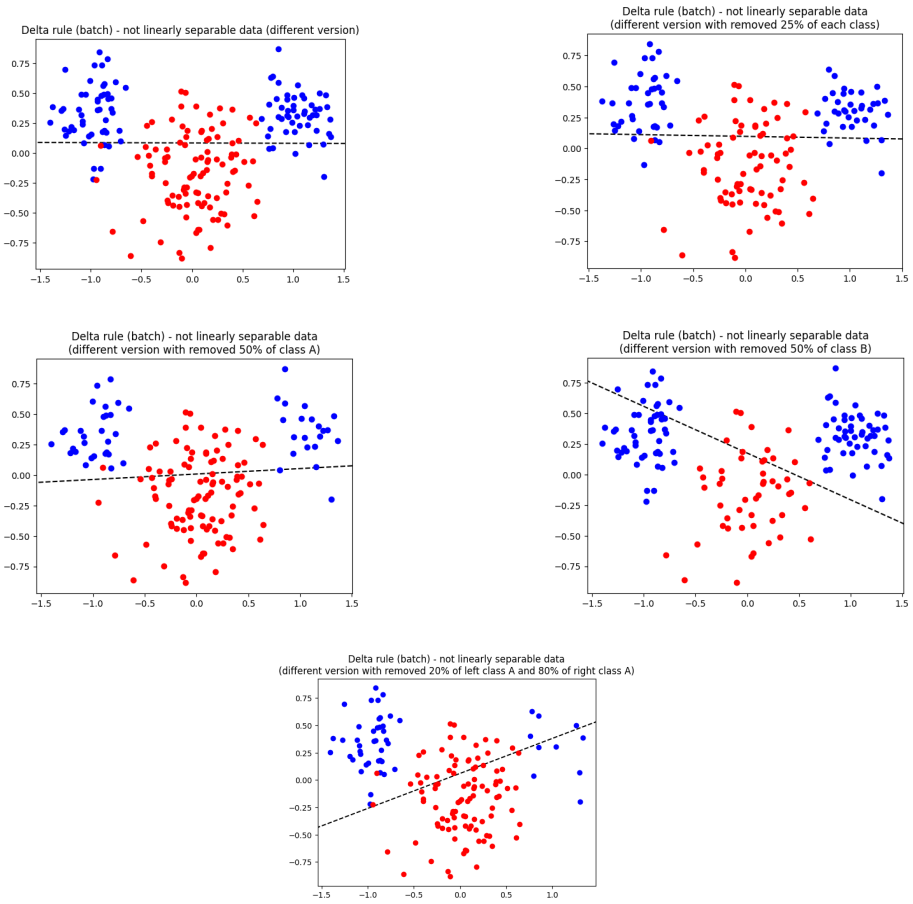
Figur 5: Classification of non linearly separable data with high variance

We observe the same pattern for the perceptron rule. The red class is on only one side of the separating line, with more blue datapoints than before. The rest of the blue class datapoints are on the other side. It makes the separating line irrelevant, as most of the datapoints are on the same side of the separating line. The algorithm will almost always fail calssifying blue data points.

The delta rule seems to deal a lot better with outliers, as most of the blue and red class datapoints are well separated.
We can also observe the using batch allows a better trade-off between sensitivity and specificity, as both indicators reach the same value. Without using batch learning, the sensitivity is significantly lower (an absolute difference of 0.2). Delta rule using batch learning shows better results in this case.

3.2.3 Non linearly separable data

The last dataset on which we wanted to train the model is a non linearly separable model as follows :



Figur 6: Classification of non linearly separable data

Here, the results provided by the delta rule are not relevant. All the algorithms fail to classify the data with a separating line, which is normal as there is no linear separation possible, even after removing some outliers.

Still, it is interesting to study the decision boundary when modifying the data, for instance for non balanced datapoints in each class.
When removing the same amount of datapoints in each class, as expected, it does not change much the separating line.
What is interesting to notice is what happens when we remove data from one of the two blue clusters. It makes rotate the separating line. It gets closer to act like a split line between the red class and the biggest remaining blue cluster. The other one ends up being considered as outlier data.
This demonstrates how unbalanced data samples from the true distribution can lead to a misunderstanding of the underlying structure. The consequences can be a bad choice of model (two simple here, as the data is actually not linearly separable) which poorly generalizes.

4 Final remarks *(ca. 0.4 page)*

The lab starts by the simple ideal example of linearly separable data, which allows to check the efficiency of the algorithms in their most relevant use case. It was also interesting to implement a more efficient algorithm of the delta rule using batch learning, to compare its efficiency, especially with more realistic datasets.
It is interesting to also study the limits of those algorithms by studying non linearly separable data, and to see how the outliers are managed by all of the algorithms, as in practice, it is rare to have perfectly linearly separable dataset. We could also observe that unbalanced data can lead to a wrong separating line that fails capturing the information of the data.