A Study of the Rosenblatt Perceptron Algorithm

Practical Assignment I for the course *"Neural Networks and Computational Intelligence"* at the University of Groningen

E. Argiolas, M.J. Havinga
(Group 59)

January 30, 2020

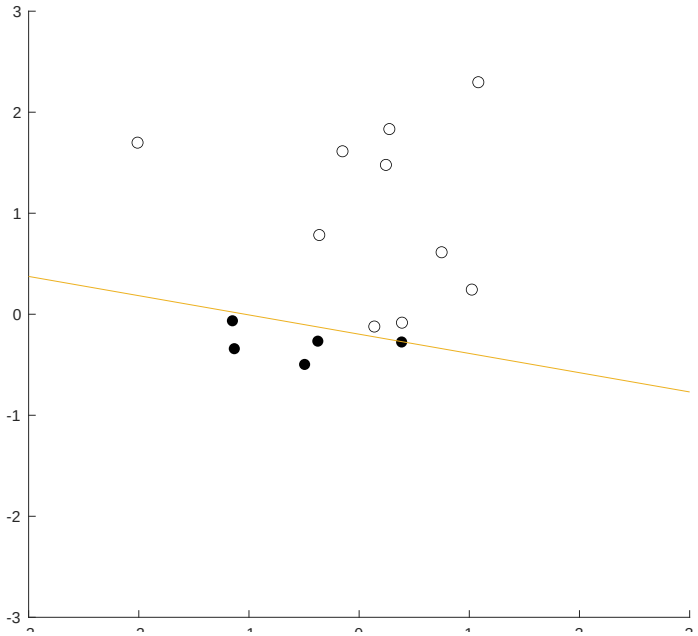

Figure 1: An illustration showing a linear separation rule in a two-class problem, as found by the Rosenblatt perceptron.

1 Introduction

In this report we investigate and discuss the perceptron algorithm as proposed by Rosenblatt [Ros58; Bie19].
The Rosenblatt perceptron is a primitive yet powerful neural network, dating back from 1958, that addresses the problem of binary classification of data. It is a *supervised* machine learning algorithm, that is to say any new (unlabelled) given data point is assigned to either class A or B based on a (labelled) pre-existing set of points, called the *training set*. Although the perceptron is most ultimately adopted to make predictions on a *test set* whose points originally lack class labels, in this document we focus more on discussing certain details of the model itself, concerning its mathematical framework.
Throughout the report, we will describe the data points as vectors $\vec{\xi} \in \mathbb{R}^N$ where N is the number of dimensions (or features); their label (i.e. the class they belong to) will be either $+1$ or -1. Strictly speaking, the goal of the perceptron is to learn from examples a weight vector $\vec{w}$ that defines a (N-1)-dimensional hyperplane - referred to as *decision boundary* - which separates the data set classwise in the feature space. Given such a vector, at any given learning step the output of the perceptron is determined by the following:

$$S_{\vec{w}}(\vec{\xi^\mu}) = \begin{cases} 1, & \text{if } \vec{w} \cdot \vec{\xi^\mu} \geq \theta \\ -1, & \text{if } \vec{w} \cdot \vec{\xi^\mu} < \theta \end{cases} = \text{sign}(\vec{w} \cdot \vec{\xi^\mu} - \theta) \tag{1}$$

Here, $S_{\vec{w}}(\vec{\xi^\mu})$ references the label of a specific feature vector $\vec{\xi^\mu}$ as predicted by $\vec{w}$. For simplicity of notation we will define $S_{\vec{w}}(\vec{\xi^\mu}) \equiv S^\mu_{\vec{w}}$. On the other hand, the feature vector $\vec{\xi^\mu}$ has an actual label S^μ_T we are aware of. These target labels combined with the feature vectors themselves make up the "input data set" $\mathbb{D} = \{\xi^\mu, S^\mu_T\}^P_{\mu=1}$, where P is the number of feature vectors (or patterns). We will refer to a certain weight vector as *correct* provided that the following holds:

$$\forall_\mu \; 1 \leq \mu \leq P : S^\mu_{\vec{w}} = S^\mu_T \tag{2}$$

If such a correct vector $\vec{w}$ exists (regardless of whether the perceptron actually finds it or not), the data set is said to be *homogeneously* or *inhomogeneously linearly separable* depending on $\theta = 0$ or $\theta \neq 0$ respectively. Indeed, since the algorithm is linear, a set of points can be correctly separated iff it is possible to do so in a linear fashion: a perceptron can't determine curved nor piecewise linear decision boundaries. It is noteworthy that a non-zero shift, that makes the hypothesis inhomogeneous in N dimensions, can be embedded in the weight vector by means of adding an extra "clamped" dimension. The input vector shall be augmented accordingly, its (N+1)-th value being fixed at -1, so that the dot product in eq. 1 is preserved in form. This way, the same hypothesis is, by definition, homogeneous in $N + 1$ dimensions.

An interesting property, closely related to the perceptron, is the probability that a certain given data set is homogeneously linearly separable. In order to estimate it, we will assume the data set to be normally distributed with mean = 0. An insightful way of seeing this probability is as the fraction of linearly separable dichotomies[1] over all possible dichotomies for the given set of points. A counting argument analytically leads to the result (see [Bie19]):

$$P_{ls}(P, N) = \begin{cases} 1, & \text{for } P \leq N \\ 2^{1-P} \sum_{i=0}^{N-1} \binom{P-1}{i}, & \text{for } P > N \end{cases} \tag{3}$$

Alternatively, we can empirically compute this fraction by running the Rosenblatt perceptron algorithm on n_D separate, randomly drawn data sets and recording whether or not a correct weight vector is found. This fraction will be denoted as

$$Q_{ls}(P, N) = \frac{n_{ls}}{n_D} \tag{4}$$

[1] A *dichotomy* is a split of a set in two mutually exclusive subsets whose union is the original set. In this case, each of the subsets contains only objects belonging to one of the two classes. All possible permutations one can think of clearly total to 2^P.

where n_{ls} is the number of random data sets $\mathbb{D}_i^{P,N}$ for which we find a correct solution vector $\vec{w_i}$ within n_{max} perceptron *epochs*[2], where $i = 1, ..., n_D$.

There are various variants of the Rosenblatt perceptron algorithm in the literature. Our experiments will make use of the following formulation of the learning step.
At most n_{max} times, we go through every feature vector in the data set and update the current solution vector $\vec{w}$ accordingly:

$$\vec{w'} = \begin{cases} \vec{w} + \frac{1}{N}\vec{\xi^\mu}S^\mu, & \text{if } \vec{w} \cdot \vec{\xi^\mu}S^\mu \leq c \\ \vec{w}, & \text{otherwise} \end{cases} \tag{5}$$

Here, $\vec{w'}$ will be the solution vector in the next step and μ represents the currently selected feature vector along with its label. Generally, we will select $c = 0$ as the threshold value. What eq. 5 does, together with eq. 1, is *learn from mistakes*: the weight vector is modified iff the presented pattern is currently being misclassified by the perceptron. If the weight vector does not change at all throughout a full epoch, the algorithm stops because there are no more misclassified points, i.e. the dataset has been correctly separated. If, however, a number of n_{max} epochs have been run through, the algorithm also terminates and we conclude that the current dichotomy cannot be found by the perceptron. In reality this isn't necessarily the case as the upper bound for the number of required epochs to reach convergence might very well be, albeit *finite*[3], very large. Therefore, we choose to set n_{max} to a reasonably high value to reduce the systematic underestimate of $Q_{ls}(P, N)$ with respect to $P_{ls}(P, N)$.
A useful way of expressing the general result for the target vector in iterative algorithms that involve sequential presentation of data is (under the assumption that $\vec{w}(0) = 0$)

$$\vec{w}(t) = \frac{1}{N}\sum_{\mu=1}^{P} x^\mu(t)\vec{\xi^\mu}S^\mu \tag{6}$$

at learning step t. Here $\xi^\mu(t)$ is the *embedding strength* of the μ-th example. In the case of the perceptron its value is integer, increasing by 1 each time that example is presented and triggers a nonzero learning step due to misclassification.

In the following sections our experiments will be outlined (section 2) and the main results thereof will be presented (section 3). Besides a from-scratch coding of the perceptron architecture, this work will mainly address the problem of its *capacity* to realize dichotomies as data sets are presented, thus leading to some genuine insight concerning usefulness and limitations of the algorithm.

2 Methods

To interact with the Rosenblatt perceptron, we implemented the algorithm as described in [Bie19] in MATLAB. In the coming paragraphs we describe what our program does and how we will use it to conduct our experiments.
First off, a data set is drawn from a normal distribution $\mathcal{N}(0, 1)$ as a set of P N-dimensional vectors. Upon zero-initialization of the weight vector, the core perceptron step (see Equation 5) is nested in a loop that scans all of the P patterns until either convergence (dichotomy has been realized) or a fixed maximum number of epochs n_{max} is reached. This is repeated for a number n_D of random data sets. To sanity check the consistency of the algorithm, we would set $N = 2$ and visualize some of the drawn data along with the discovered decision boundary in a scatter plot. An example of such a plot serves as the picture on the cover sheet of this report (Figure 1).
In order to compare Equation 3 and Equation 4, the above procedure was nested once more in a loop to vary the value of P, such that $\alpha = \frac{P}{N} \in (0, 4]$. In the end, we would then inspect a stair plot of the analytic results next to a point-wise plot of the experiment results within the

[2] A learning epoch is one complete presentation of the data set that is to be separated.

[3] The statement that a perceptron is bound to linearly separate any given linearly separable set in a finite number of epochs is known as the *Perceptron convergence theorem.*

same picture. This will be our first experiment (referred to as Experiment I). On a side note, the choice of the interval for α has theoretical roots: One expects the fraction of linearly separable dichotomies to drop as α crosses the value 2, where this drop becomes increasingly more abrupt with higher dimensionality of the feature space N.[Bie19] As a result, a small interval centered in $\alpha = 2$ happens to be the most interesting region in the parameter space to be inspected.
On top of varying α, we can explore the parameters further. For instance, increasing the parameter n_D decreases variance and draws the empirical frequency nearer to an actual probability. Increasing n_{max} makes the perceptron more "stubborn" in its task of finding a linear separation of each of the data sets and therefore more trustworthy. Because we hypothesize that the ability to approximate the actual probability (as represented by P_{ls} in Equation 3) using our perceptron depends on both the complexity of the data points themselves, i.e. the number of dimensions N, as well as the programs' own resolution to find a solution as represented by the n_{max} parameter, we investigate the relative discrepancies between Q_{ls} and P_{ls} for various values of those parameters in Experiment II.
Changing the threshold c to a (small) positive nonzero value would (slightly) relax the conditions under which a learning step is triggered, i.e. make more strict the requirements for a data point to be considered as correctly classified - potentially affecting the convergence time.
As a side note, our program has the capability to augment feature vectors with an N+1^{th} dimension in order to reduce inhomogeneously linearly separable problems to homogeneously linearly separable problems. However, in our experiments we will assume the homogeneous case by default and this augmentation functionality will not be used.

3 Results

Experiment I: Exploring Q_{ls} versus P_{ls}

As described in section 1, we generate a value of Q_{ls} as an experimental approximation of the theoretical fraction of linear dichotomies in an N-dimensional data set of P data points. First, we select $N = 20$ as our dimensionality and run our program with $n_D = 50$ (attempt to linearly separate 50 independently drawn random data sets) and $n_{max} = 100$ (the maximum number of epochs in the Rosenblatt algorithm). We can increase the parameters for potentially better results to $n_D = 500, n_{max} = 1000$. The results of this experiment with $N = 20$ are shown in Figure 2. To compare with higher dimensions, we run the program for $N = 100$ with the higher parameters ($n_D = 500, n_{max} = 1000$) and even higher parameters ($n_D = 500, n_{max} = 10000$). The result from this experiment are shown in Figure 3. We also run the program on a lower dimensionality of $N = 4$ with $n_D = 500, n_{max} = 1000$, as seen in Figure 4.

Experiment II: Exploring the systematic underestimation bias of Q_{ls} compared to P_{ls}

Because the ability of the Rosenblatt algorithm to find the linear dichotomy in a data set if it exists is limited by the n_{max} parameter, we investigated the discrepancies between Q_{ls} and P_{ls} for different values of n_{max} and N. Our metric for the discrepancy is the relative underestimate or error of Q_{ls} compared to P_{ls}, computed as $1 - \frac{Q_{ls}(P,N)}{P_{ls}(P,N)}$. For each combination of N and n_{max}, the Rosenblatt algorithm is ran 1000 times on a independently drawn data set, as in Experiment I (thus, $n_D = 1000$). In this experiment however, we only consider a single value for α. The setting $\alpha = 2$ is arguably the most interesting configuration and the results of this experiment are shown in Figure 5.

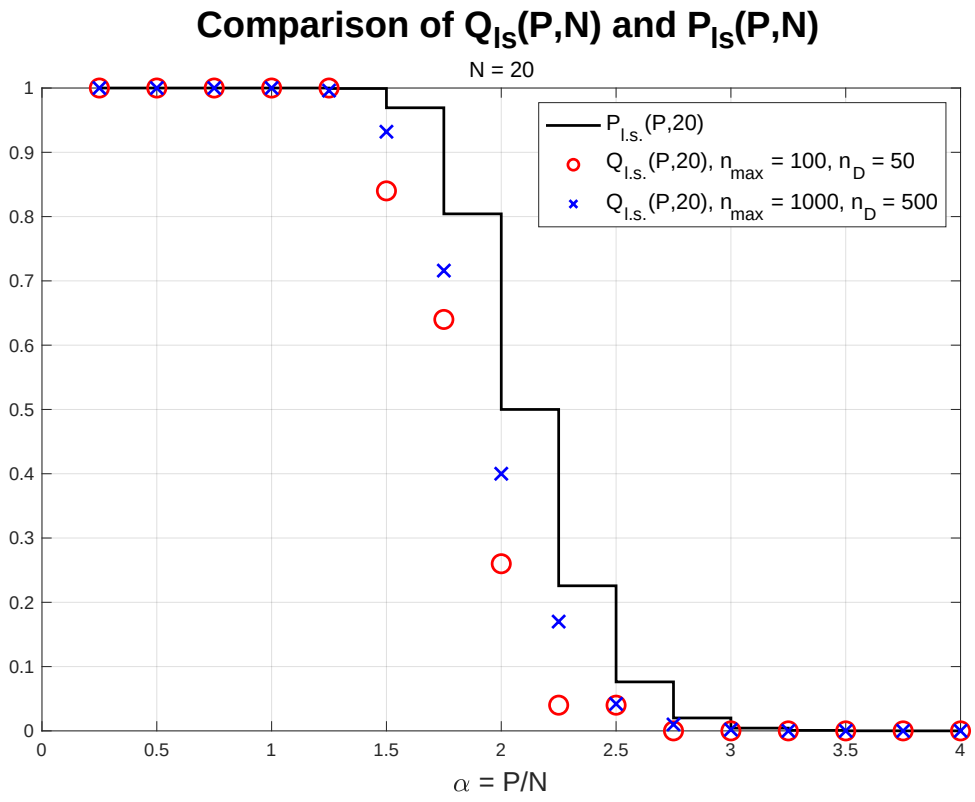

Figure 2: Found values for $Q_{ls}(P, N)$ compared to calculated values for $P_{ls}(P, N)$ for $N = 20$.

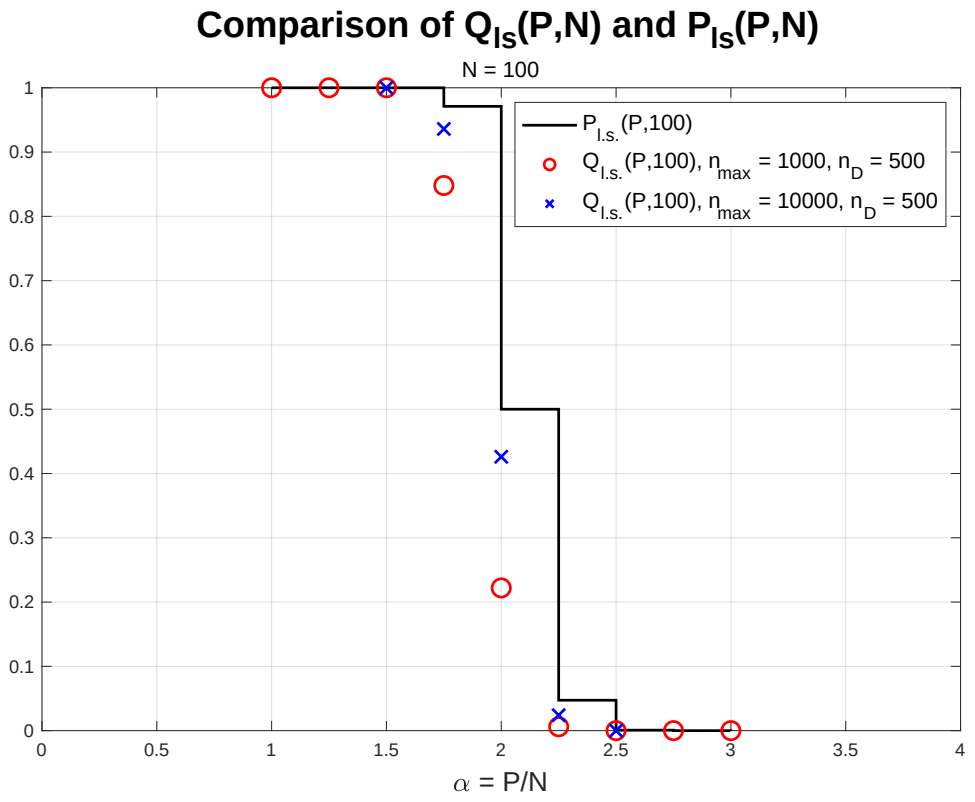

Figure 3: Found values for $Q_{ls}(P, N)$ compared to calculated values for $P_{ls}(P, N)$ for $N = 100$.

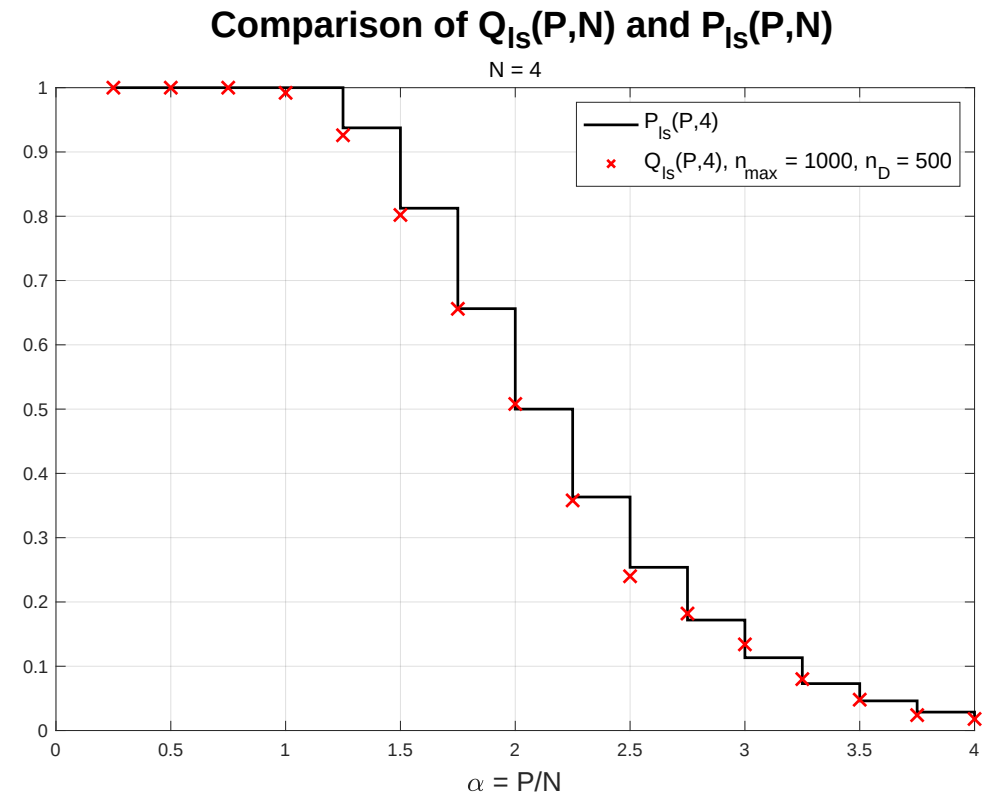

Figure 4: Found values for $Q_{ls}(P, N)$ compared to calculated values for $P_{ls}(P, N)$ for $N = 4$.

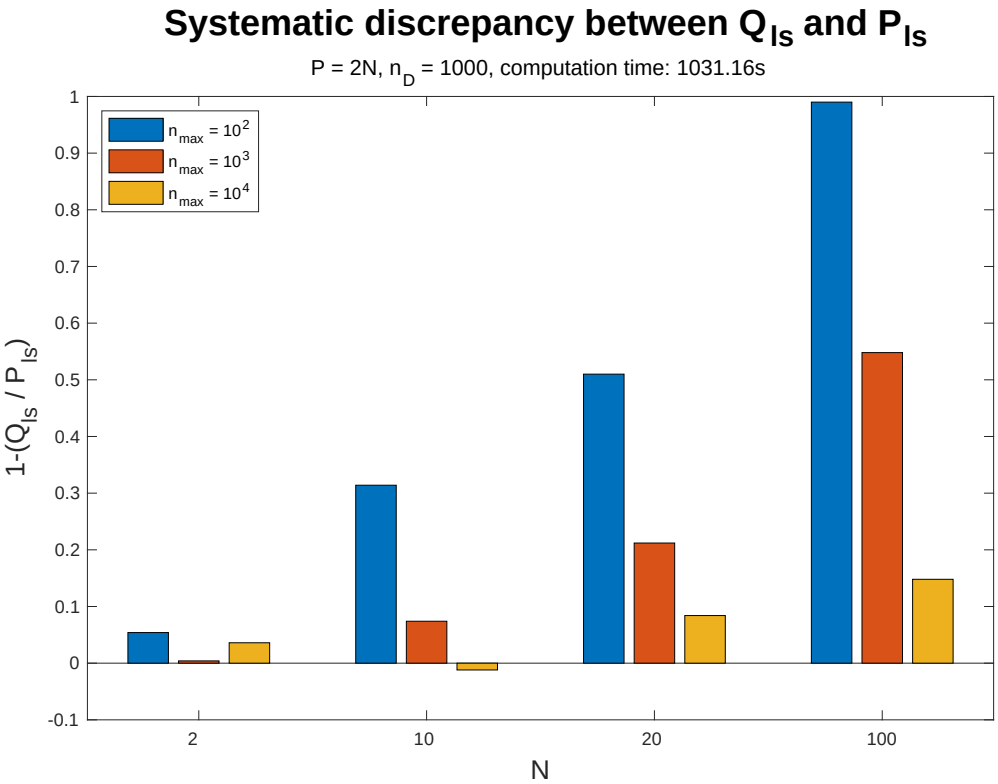

Figure 5: Typical discrepancies between Q_{ls} and P_{ls} at $\alpha = \frac{P}{N} = 2$ for different values of N and n_{max}.

4 Discussion

Experiment I: Exploring Q_{ls} versus P_{ls}

In this experiment we gained several insights on the influence of the various parameters with relation to the found value for $Q_{l.s.}$, as compared to $P_{l.s.}$. In general, we observed a gradual decline in the fraction of linear dichotomies as α increases, in a very similar fashion to the theoretical P_{ls} values. Theory predicts that for higher dimensions the perceptron's separation abilities decline increasingly steeply around $\alpha = 2$ or $P = 2N$. Comparing this theoretical insight, represented by the P_{ls} values, to the experimental Q_{ls} values, we see that they clearly follow a similar pattern. It's clear that our experimental results consistently underestimate the theoretical values, which is what was expected because the actual perceptron is never guaranteed to find a solution even if it exists. This effect can be larger or smaller depending on the relation between the complexity of the problem (in terms of N and P) and the resolution of the algorithm in terms of maximum epochs (n_{max}). For example as we saw in Figure 2, the values for Q_{ls} become a better estimation of P_{ls} for a higher value of n_{max}. Similarly, a higher dimensionality of $N = 100$ was shown to result in poorer results for the same value of n_{max}. Especially the latter insight is noteworthy in our opinion, as it implies that a higher dimensional problem inherently requires more epochs to achieve a good result.

Experiment II: Exploring the systematic underestimation bias of Q_{ls} compared to P_{ls}

In this experiment we further explore the underestimation bias of Q_{ls} compared to P_{ls} which we first observed in Experiment I. As is in line with our expectations, the found discrepancies indeed increase with higher dimensions N and decrease with higher values of n_{max}. For $N = 2$, there is no noticeable effect of increasing n_{max}. Presumably this is because $n_{max} = 100$ is a sufficient value in almost every case of such a data set and any found discrepancies can be attributed to random (noise) errors. For higher dimensions, we see a convergence behavior in the decline of the discrepancy that illustrates the inability of the perception to reach theoretical perfection. In other words, the improvement made with increasing n_{max} with another order of magnitude becomes increasingly smaller. For $N = 10$, we achieve a very good result for $n_{max} = 10^4$, while for that same n_{max} the results for $N = 20$ and $N = 100$ imply an increasing inability of the perceptron to find solutions at $\alpha = 2$.

5 Conclusions

In this report we explored the capabilities of the Rosenblatt perceptron at different parameters. Primarily, we showed that a sufficient maximum number of epochs is essential for finding correct linear separations in a data set if one exists. We saw that for a higher number of dimensions, the perceptron was increasingly unlikely to find correct linear separation rules, which could be partly addressed by increasing n_{max}. However, we clearly showed that the relationship between n_{max} and the perceptron's ability to find correct linear separations is not linear. Rather, it is suggested that the perceptron's success rate converges to the theoretical maximum, but will never reach it.

References

[Bie19] Michael Biehl. *Learning from Examples - An Introduction to Neural Networks and Computational Intelligence*. Nov. 2019.

[Ros58] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

A Appendix: Distribution of work

In this section, we describe the distribution of the individual workload for this project.
The initial MATLAB code implementing the perceptron was written by Thijs and perfected by Edoardo who then conducted the first experiments with computing the ratio of linearly separable data sets. The report was also collaboratively established by means of a shared Overleaf[4] document. Thijs wrote a first version of each chapter (section 1), which Edoardo expanded and perfected.

[4] `https://overleaf.com`