Good language, good discussion, no bonus.
sufficient. HK

Neural Networks
Practical Assignment I: Perceptron training

Rick van Veen (s1883933) Laura Baakman (s1869140)

December 12, 2014

I. Introduction

Artificial neural networks are non-linear mapping systems inspired by biological nervous systems. The most import parts of a biological neuron include a dendrite that receives signals from other neurons, and a soma that integrates the signals and generates a response that is distributed via a branching axon.

An artificial neural network consists of a large numbers of simple processors linked by weighted connections, analogously the neurons. Each processor receives inputs from many other nodes and generates a single scalar output that depends on locally available information. This scalar is distributed as input to other nodes.

The most simple case of a neural network is the single-layer perceptron, see Figure 1. This network consists of one layer of input nodes connected to a processing unit through a single layer of weights, which determine the result of the output node. Mathematically a single-layer perceptron is a feed-forward network of nodes with a response function $f(\mathbf{w}^T\mathbf{x})$ where $\mathbf{w}$ is the vector of weights, $\mathbf{x}$ is the pattern and f is a sigmoid squashing function[3]. This squashing function ensures the binary output of the perceptron.

The perceptron is a linear binary classifier, which means that is separates classes with a hyperplane, as a consequence perceptrons can only separate linearly separable datasets. We identify two different types of linearly separable datasets: homogeneously separable data sets can be separated by a hyperplane through the origin, inhomogeneously separable data sets cannot be separated by a hyperplane through the origin but they can be separated by a hyperplane if it is offset with a certain bias with respect to the origin. This bias can be incorporated into the network by adding a fixed input of minus one and the bias as the weight connected to this fixed input. Thus we will make no distinction between homogeneously and inhomogeneously separable datasets and talk about linearly separable sets [4].

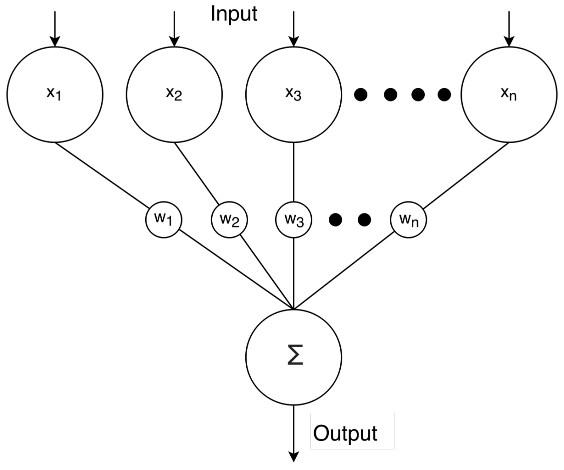

Figure 1: *A perceptron.*

It can be shown that the perceptron converges, i.e. it separates positive examples for the negatives, if the input data set is linearly separable.

Cover [1] showed that the probability that a randomly chosen dichotomy in general position is linearly separable can be proven to be:

$$f(N,d) = \begin{cases} 1 & N \leq d+1 \\ \frac{2}{2^N} \sum_{k=0}^{d} \binom{N-1}{k} & \text{otherwise} \end{cases} \quad (1)$$

Based on this presented probability one can show that a d-input linear separator has a high probability of being able to learn a random dichotomy on up to $2d$ patterns. When $N < 2d$ generalization may be poor because the network is underconstrained and the solution is unlikely to generalize well to new patterns [3].

In this paper we consider the Rosenblatt algorithm for perceptron training, furthermore we show with two experiments that our implementaion exhibits the mentioned theoretical properties.

II. Method

Given a dichotomy $\mathcal{D} = \{\xi^i, S^i\}_i^N$ with N d-dimensional patterns $\xi \in \mathcal{R}^d$, each with a label $S \in \{-1, +1\}$, a perceptron can be trained using the Rosenblatt algorithm

$$\mathbf{w}(t+1) = \begin{cases} \mathbf{w}(t) + \frac{1}{d}\xi^{\mu(t)}S^{\mu(t)} & \text{if } E^{\mu(t)} \leq 0 \\ \mathbf{w}(t) & \text{otherwise} \end{cases} \quad (2)$$

Where $\mu(t) = 1, 2, \ldots, N, 1, 2, \ldots$ is used to select the next pattern to train with. The energy function $E(\cdot)$, anagolous to the local action potential in a biological neuron, is defined as:

$$E^{\mu(t)} = \mathbf{w}(t) \cdot \xi^{\mu(t)} S^{\mu(t)}. \quad (3)$$

The energy function indicates if the perceptron gives the correct output for the given input pattern $\xi^{\mu(t)}$. Since we know that independent of the initial value of the weights the perceptron will converge, if the $\mathcal{D}$ is linearly separable, the initialization of the weights is irrelevant. We have opted for $\mathbf{w}(0) = 0$.

The update defined in Equation 2 is executed until $E^{\xi^i} > 0$ for $i \in [1, N]$, if this is the case the algorithm has converged. If the dataset is not linearly separable the perceptron never converges, thus it is generally a good idea to set a maximum number epochs, d_{max}, where each epoch consists of N steps. The total number of steps taken by the algorithm has thus the upper bound $d_{max} \cdot N$.

To test if our implementation of the perceptron exhibits this theoretical property we have performed two experiments. The first is concerned with the implications of Equation 1, the second with the approximation to the step function.

III. Experiments

Since Equation 1 only holds when the points are general position we sampled them from a Gaussian distribution for both experiments, since the points are then guaranteed to be linearly independent [2, Chapter 5]. In these experiments we have made the number of patterns dependent upon some parameter α according to:

$$N = \alpha d. \quad (4)$$

We define $Q_{l.s.}$ as the fraction of data sets where the perceptron found a hyperplane to separate the two classes within the d_{max} epochs.

Experiment I

For the first experiment we have sampled 500 50-dimensional from a standard normal distribution. The number of data points per data set is dependent upon (4), d_{max} was set to 1000. The ratio $Q_{l.s.}$ for different values of α is plotted in Figure 2.

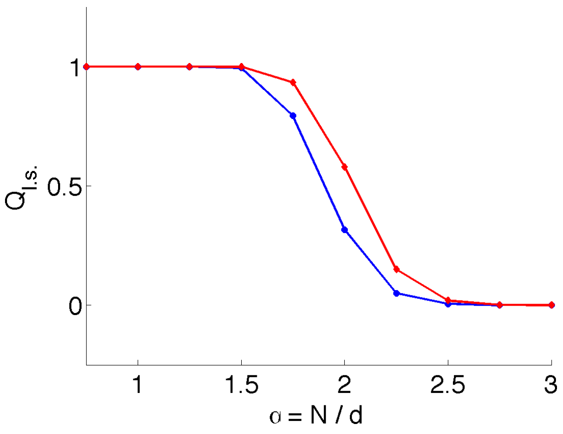

Figure 2: *The found relation between α and $Q_{l.s.}$, shown in blue, and the theoretical relation in red.*

Using that $d = 50$ and substituting (4) into the condition of the first case of (1) we find that $f(N,d)$ should be one for $\alpha \leq 1.02$. This fits with our results, which show that $Q_{l.s.}$ is one for all tested values of α that are smaller than or equal to 1.02.

For $\alpha \in [1.5, 2.5]$ we find that the empirical $Q_{l.s.}$ is a bit lower than the theoretical value. The first of two possible causes is that the parameter d_{max} was set too low. The perceptron convergence theorem states that the perceptron eventually converges, not necessarily after 500 steps. Another possible explanation is the fact that we are talking about chances, and that we just happened to get higher proportion of non-linearly separable datasets than theory predicts.

For $\alpha \geq 2.5$ we find that the theoretical and empirical results coincide once again, this is due to $f(N,d)$ approaching zero.

Experiment II

In this experiment we repeat the experiment I, with different values for d in order to find if $Q_{l.s.}$ approaches the step function. The resulting plots of this experiment and the step function are shown in fig. 3. We can see when the dimensionality increases and we keep the proportion $\alpha = N/d$, that N will approach ∞, and the plots shown in fig. 3 will approach the step function, this follows the theory from Cover [1] presented in the introduction. The reasons why the empirical $Q_{l.s.}$ are a bit lower than the theoretical values are the same as those stated for experiment one.

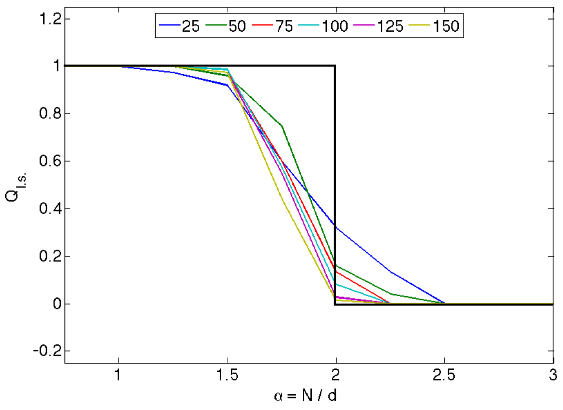

Figure 3: *Number of successful runs $Q_{l.s.}$ with different values for d and the step function drawn in black.*

IV. Conclusion

The behaviour of our implementation fits with the presented theory.

References

[1] Thomas M Cover. "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition". In: *Electronic Computers, IEEE Transactions on* 3 (1965), pp. 326–334.

[2] Simon JD Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[3] Russell D Reed and Robert J Marks. *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1998.

[4] Raúl Rojas. *Neural networks: a systematic introduction*. Springer, 1996.