Neural Networks
Practical Assignment I: Perceptron training

Rick van Veen (s1883933) Laura Baakman (s1869140)

December 2, 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[1]. 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.

Reference to paper/book that shows that there is no distinction between homogeneously and inhomogeneously separable datasets.

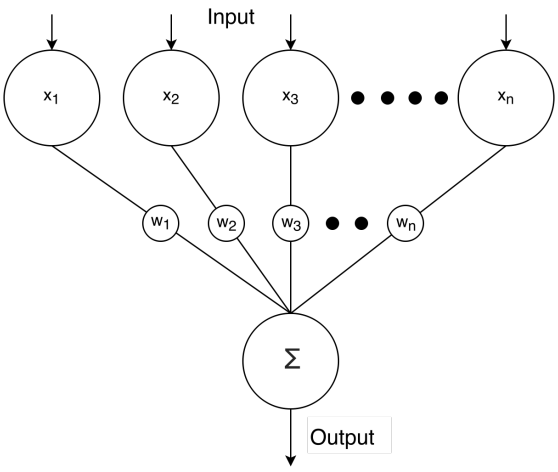

Figure 1: *A perceptron.*

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.

Reference to book with the proof

In this paper we will consider the Rosenblatt algorithm for perceptron training.

Referentie naar het experiment

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

Add a reference to the Rosenblatt algorithm

which updates the weights each time step $t = 1, 2, \ldots$:

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

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:

$$E^{\mu(t)} = \mathbf{w}(t) \cdot \xi^{\mu(t)} S^{\mu(t)}. \tag{2}$$

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 $\mathbf{w} = 0$.

The update defined in Equation 1 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 $d_{max} \cdot N$.

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

$$f(N, d) = \begin{cases} 1 & N \leq d + 1 \\ \frac{2}{2^N} \sum_{k=0}^{d} \binom{N-1}{k} & \text{otherwise} \end{cases} \tag{3}$$

To verify this theoretical result we have performed two experiments.

References

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