Neural Networks
Practical Assignment I: Perceptron training

Rick van Veen (s1883933) Laura Baakman (s1869140)

November 25, 2014

Neural Smithing (p21) heeft een formule die de kans geeft dat een randomly gekozen dichotomy l.s.

Het lijkt alsof hij met de remark aan het einde van de slides wil zeggen dat patterns uit getrokken uit een standaard normaal distribution altijd in general position zijn.

Use compactitem environment for lists

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, 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 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 perceptron is any feed-forward network of noes with responses like $f(\vec{w}^T\vec{x})$ where $\vec{w}$ is the vector of weights, $\vec{x}$ is the pattern and f is a sigmoid-like squashing function[1]. This squashing function ensures the binary output of the perceptron.

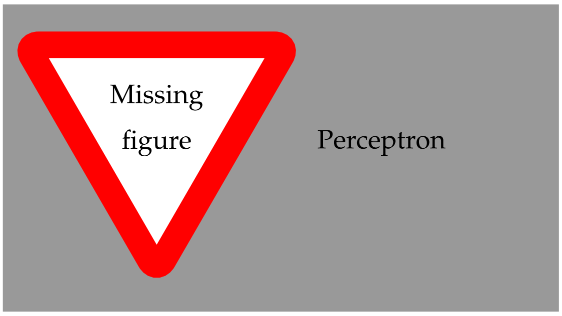

Figure 1: *A perceptron.*

The perceptron is a linear binary classifier which means that it can only classify data sets that are homogeneously separable, which means that one can separate the data points of different classes with a vector through the origin. Inhomogeneously separable datasets are linearly separable when the separating vector is moved with some offset with respect to the origin.

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.

II. Method

Given a dataset $\mathcal{D}$ with N patterns $\xi \in \mathcal{R}^d$ each with a label S: $\mathcal{D} = \{\xi^\mu, S^\mu\}_\mu^N$, a percptron can be trained using the Rosenblatt algorithm which updates the weights each time step =1, 2, . . .:

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

Where $\mu(t) = 1, 2, \ldots, N, 1, 2, \ldots$ denotes the present pattern. $E(\cdot)$ the energy function is defined as:

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

The energy function indicates if the perceptron gives the correct output for the given input pattern $\xi^{\mu(t)}$.

The update set defined in Equation 1 is executed until $E^v > 0$ for $v \in [1, N]$ in this cases the algorithm has converged. If the dataset is not linearly separable it will never converge, thus it is generally a good idea to set a maximum number of time steps.

The update method (1) only works for homogeneously separable data sets. To classify a inhomogenously separable dataset one needs to add a one extra input to all patterns, namely minus one and one extra weight, associated with the extra input.

References

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