Chapter 1

Background theory

1.1 Feed-Forward neural networks

To understand the technique used in this report, it is necessary to understand basic neural networks functionning. Given a scenario with a training set of labeled data $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ denotes the training example composed of multiple features, say $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, and $\mathbf{y}$ the corresponding label. Let's introduce the idea of 'perceptron'.
Perceptrons are the building blocks of neural networks, and the best way to get stated is with an example. Assume at the university's admission office the students are evaluated with two pieces of information, the results of a test and their grades in school. Let's take a look at some sample students, see fig. 1.1.

Figure 1.1: test vs grades

The data on the figure can be nicely seperated with a line, where most students above the line get accepted and most students under the line get rejected, see fig. 1.2. Therefore this line is going to be our model.
The model makes a couple of mistakes since there are a few blue points that are under the

line and few over the line, but they are concidered as noise and add no new information to our model. Now, the natural question that arises: how do we find the line ?

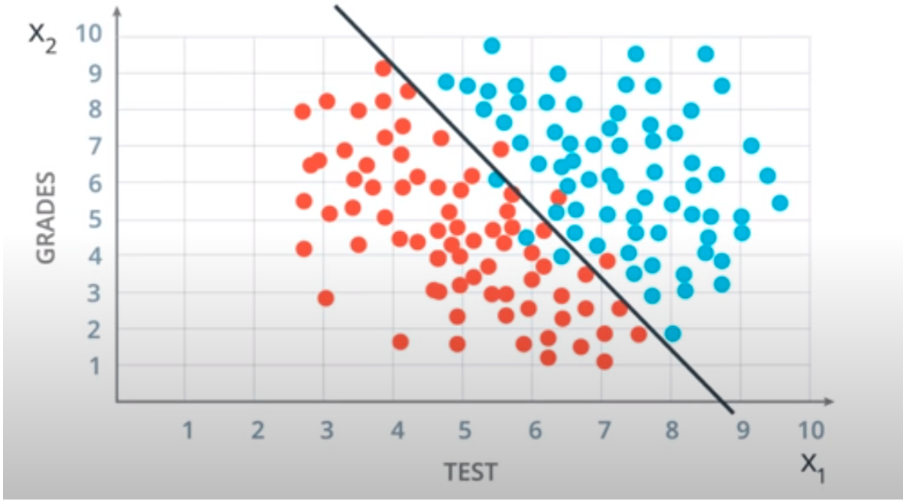

Figure 1.2: seperating line

We start by labeling the axies $\mathbf{x} = \{x_1, x_2\}$. The boundering line seperating the students has a linear equation specifically: $2x_1 + x_2 - 18 = 0$. Ploting the grades in the equation gives rise to a score, if the score is poitive –the student gets ploted in above the line–, the student gets accepted with otherwise not. This is called a prediction.
In a more general case, our boundary will be an equation of the following form: $w_1x_1 + w_2x_2 + b = 0$. Abbreviating this equation into vector notation:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{1.1}$$

Where $\mathbf{w} = \{w_1, w_2\}$. We refer to $\mathbf{x}$ as the input, $\mathbf{w}$ as the weights and b as the bias. Here $\mathbf{y} = \{0, 1\}$ is the label, where 0 indicates the student being rejected whereas 1 indicates the student being accepted. Finally, our prediction is going to be called $\hat{\boldsymbol{y}}$ and it will be what the algorithm predicts that the label will be, namely:

$$\hat{y} = \begin{cases} 1, & w \cdot x + b \geq 0 \\ 0, & w \cdot x + b < 0 \end{cases} \tag{1.2}$$

and the goal of the algorithm is to have $\hat{\boldsymbol{y}}$ resembling $\boldsymbol{y}$ as closely as possible. Reorgnizing the equations in a graph and generalizing, gives rise to fig. 1.3. Here the bias is consider as a dummy input with value 1 to the Perceptron with weight b.

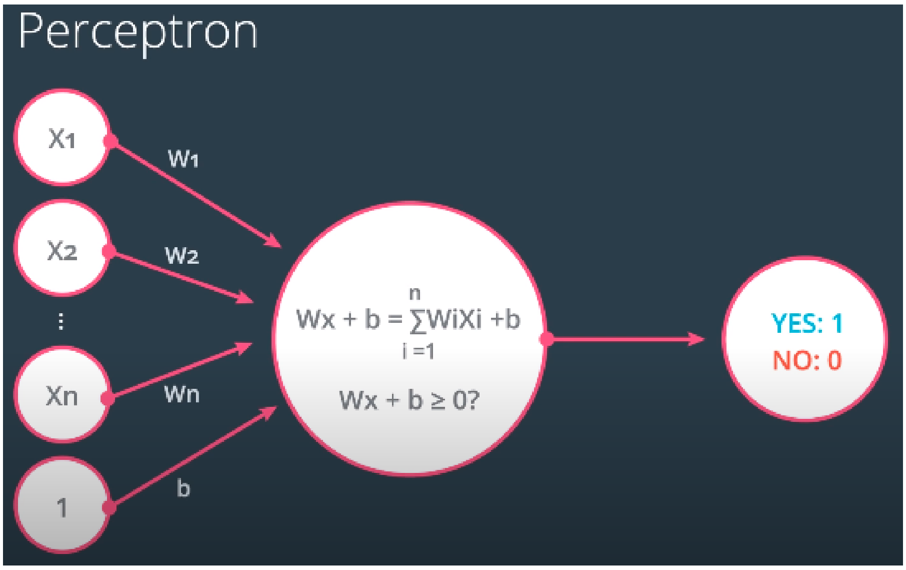

Figure 1.3: perceptron's graph representation

Giving a training set, the classification of the input vector is based upon **w**, therefor the goal of the algorithm is to determine those weights through an iterative process. The probelm given here is that the algorithm can only learn linear functions since the output of the perceptron is a simple linear combination of the input. In real world situation, the data is much more complex and highly non-linear, therefore to remedy this problem we shall introduce a non-linearty in our algorithm, namely the sigmoid function.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid function returns the probability of an event occuring, thus in our example the sigmoid function would output the probability of a student getting accepted or rejected, which can be expressed as the the conditional probability $p(y = 1|x) = \sigma(w \cdot x + b)$.

In order to estimate the accuracy of the algorithm, we take the joint probability of the entire training set, assuming the training examples being independent events:

$$p(y^{(1)}, y^{(2)}, \ldots, y^{(n)}|x^{(1)}, \ldots, x^{(n)}) = \prod_{i=1}^{n} p(y^{(i)}|x^{(i)}) \tag{1.3}$$

where $x^{(i)}$, $y^{(i)}$ represent the i^{th} training example and label respectively.

Bibliography

Symbols

CNN Convolutional neural networks

ANN Artifitial neural networks