HOCHSCHULE ESSLINGEN

DEEP LEARNING DEMONSTRATION

# Traffic Sign Recognition

*Author:*
Sergio IZQUIERDO

*Supervisor:*
Dr. Dominik SCHOOP

December 6, 2017

Hochschule Esslingen
University of Applied Sciences

## Abstract

caca

# Contents

From the begining of computing, the machines have solve probems incredibly difficult for a human, problems based on mathematicals rules. But what has been a really difficult task for computer was to solve some problems that are simple and easy for humans such as images or speech recognition. Tasks that a person perform easily but that are difficult to describe formally.

A person needs a big amount of knowledge to live and to understand, and many times this knowledge cannot be described in a formal way. That means that if we want the computers to performs the same tasks as an human, that computer should acquire that knowledge, by learning.

If it is said that a computer learns that means that it understands a complex concept as a hierachy of simpler concepts. The hierachy of concepts is deep, with many layers composed of concepts, and hence the term of deep learning.

To achieve this approach Neural Networks are going to be used. They are copying the behaviour of a brain neural network. Multiple layers of neurons that are connected to each others and pass information through those connections. This model, allows the computer to learn since the neurons adapt the output for a given input depending on the expected answer. That means that with a big amount of data to learn, the network could predict a correct answer for the input. [1]

# Chapter 1

# Neural Networks

The brain has been studied deeply during the last century. The researchers wanted to use the power of the brain in a mathematical or computational model. The first step in the Deep Learning was to copy the behaviour of the brain neurons to a circuit (1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts). The poor computing capacity of the time stopped the studies on the topic.

In the 80s the field became again interesting with the inclusion of multiple layered neural networks. In 1986 researchers modeled the idea of back propagation. This idea helps the network to distribute pattern recognition errors throughout the whole network. The problem is that with this algorithm the net learn slowly, so many iterations are needed.

Nowadays Neural Networks are incredibly important. The big data and the computing capacity are a perfect base for the networks. Also with new types of models such as Convolutional Neural Networks or Recurrent Neural Networks they are solving amazing problems beyond the image recognition. [6] [7]

## 1.1 What is a Neural Network

You could think of a Neural Network as a box with a certain number of inputs and outputs. For a given input the box answer with a output, that is the network prediction.

Let us imagine that we have one box of this kind, and if you input a image of a handwritten digit, the output of the box will be the digit of the picture. You could think that if you create one box like this you have all your work done, but the box by itself is stupid. If you create a box and feed it with the picture of a '1' the output could be whatever.

What we should do is to teach the net so it could answer correctly. To achieve it our box has several knobs that we can regulate in order to tell the box how bad was the prediction it said. In that way, after correcting the knobs several times, the box has learned, what means that it is ready for answer right. Now we have a box with the knowledge to make good predictions, but since it is a box we cannot outcome any knowledge for us. We cannot get any method to create an algorythm that solve our task more efficiently. We just can use our box, and get our answer from our box.

## 1.2 Under the Hood

Now that we have understood our box it is time to start knowing what is inside it. The neural network is composed by several layers. At least we have the input layer (receive the data from outside) and the output layer (give us the prediction). Between those two could be more hidden layers. Each of them has a certain number of neurons $n_1, n_2, .., n_m$ where $m$ is the number of layers. That means that the input layer has $n_1$ neurons while the output layer has $n_m$ neurons. Each neuron of the first layer is connected with each neuron of the second layer, each neuron of the second layer is connected with each neuron of the third layer and so on (See Figure 1.1).

Each of those connections has a weight $w$. To denote these weights we are going to use the form $w_{ij}^{(k)}$ where $k$ means which layer the connection departs from, $i$ means which neuron of layer $k$ the connection departs from and $j$ means which neuron of layer $k+1$ the connections arrives to (See Figure 1.1). Therefore $w$ is:

$$\begin{aligned} & w_{ij}^{(k)} \\ & i = 1, \ldots, n_k \\ & j = 1, \ldots, n_{k+1} \\ & k = 1, \ldots, m \end{aligned} \tag{1.1}$$

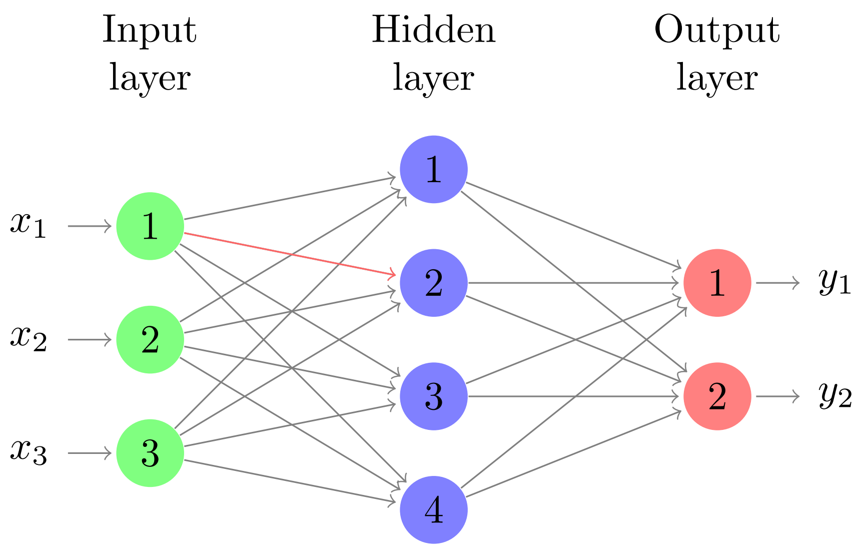

Figure 1.1: Example of Neural Network with 3 layers. Here $n_1 = 3$, $n_2 = 4$ and $n_3 = 2$. The connection in red is denoted as $w_{12}^{(1)}$ (according to Equation 1.1)

Each neuron has a bias that allow set how excited is a neuron. A really excited neuron always output '1' while a non-excited neuron always give us '0'. The bias helps the network to adjust some neurons as important since not every neuron output is equally interesting. The work of a single neuron is as simple as add the bias and the weighted inputs and apply a function to the result [3, Chapter 27]. The output $o$ of a neuron is:

$$o_i^{(k)} = \begin{cases} x_i, & k = 1 \\ f\left(u_i^{(k)} + \sum_{j=1}^{n_{k-1}} w_{ji}^{(k-1)} \cdot o_j^{(k-1)}\right), & k > 1 \end{cases} \tag{1.2}$$

For example, for the neuron 3 in blue of Figure 1.1 the output will be $o_3^{(2)} = f(u_3^{(2)} + w_{13}^{(1)} \cdot o_1^{(1)} + w_{23}^{(1)} \cdot o_2^{(1)} + w_{33}^{(1)} \cdot o_3^{(1)})$. For the same neuron the scheme is showed in Figure 1.2.

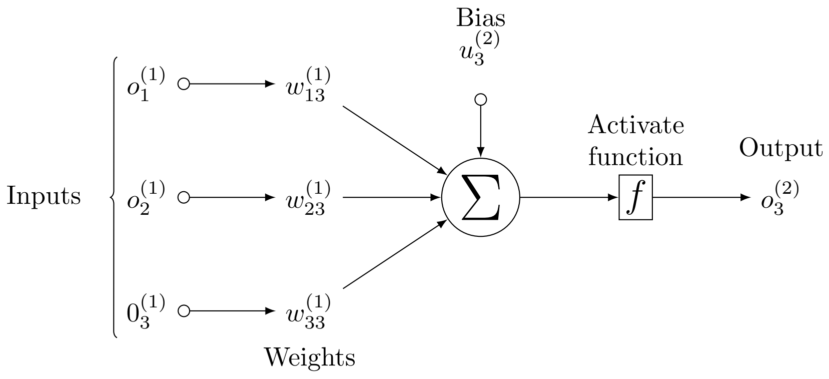

Figure 1.2: Example of the task of a single neuron [2]

## 1.3 Back Propagation

Now a Neural Network could be programmed with all showed on 1.2, a program that output $y_1$, and $y_2$ could be developed. But, where is the "learning" of deep learning here? Nowhere, yet. Is the moment of teaching the net with examples, giving it some inputs and showing it the expected outputs. The *Back Propagation Algorithm* is going to achieve it. The goal of this method is to minimize the error using the method of gradient descent.

### 1.3.1 Activation Function

As explained before each neuron has to apply one function to the result of its computing. One of the most popular functions used for this purpose is the sigmoid:

$$s(x) = \frac{1}{1 + e^{-cx}} \tag{1.3}$$

The shape of this function depends on the value of $c$, but for simplicity $c = 1$ is going to be used. The derivate of the sigmoid function respect of x is $s(x)' = s(x)(1 - s(x))$. One of the reasons why this function is used is the simplicity of the derivate, because it allows to make all the derivates later more easily [5, Chapter 7].

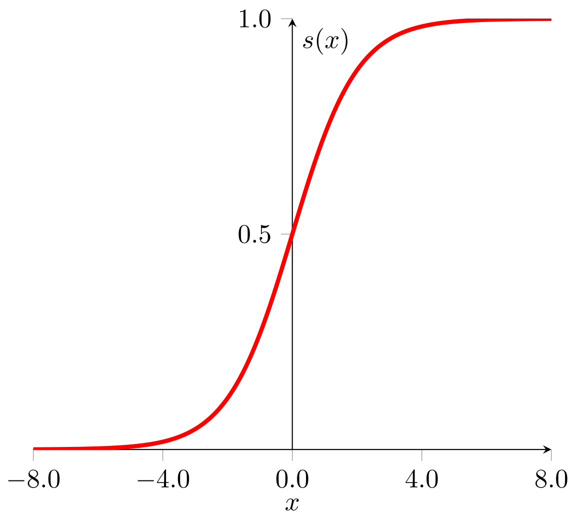

Figure 1.3: Shape of the sigmoid with $c = 1$

### 1.3.2 The Error

The error could be measured in differents ways. The most common and apparently most effective method is to calculate the Euclidean distance. It allows to measure the error in different dimensions, for example, the error in $\mathbb{R}^1$ between 4 and 1 is $E = 4-3 = 1$. For $\mathbb{R}^2$ you should use the Pythagorean theorem, and for $\mathbb{R}^{\mathrm{n}}$ the distance is:

$$E_{(p,q)} = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} \quad (1.4)$$

Is important to know how to size the error that the net has because you have to tell the network how far was the result from the expected output. And only once you have the distance between the target and the output you can adjust the knobs to achieve a better result.

How much the knob must be changed? It depends on how important is that knob for the result. For example it might be possible that changing one knob will get the same result. In this case it does not matter if we adjust the knob, but, what about one knob that when adjusted change the result drastically? Here the new position of the knob is realy important. Hence we need the partial error with respect a specific weight or bias [4, Chapter 2]:

$$\frac{\partial e}{\partial w_{ij}^{(k)}} \quad \text{or} \quad \frac{\partial e}{\partial u_i^{(k)}} \quad (1.5)$$

### 1.3.3 The Correction

Once we can calculate the error depending on one parameter we are ready to make the correction to the network. The process is simple, first you feed the input layer with some data, then you get the result and compare it with the target and finally you make this adjustment for each weight and bias [5, Chapter 7]:

$$\begin{aligned} w_{ij}^{(k)} &= w_{ij}^{(k)} - \alpha \frac{\partial e}{\partial w_{ij}^{(k)}} \\ u_i^{(k)} &= u_i^{(k)} - \alpha \frac{\partial e}{\partial u_i^{(k)}} \end{aligned} \tag{1.6}$$

In the adjustment the $\alpha$ is the learning constant and represent how far is the next step in the gradient descent.

# Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[2] Gonzalo Medina (https://tex.stackexchange.com/users/3954/gonzalo-medina). *Diagram of an artificial neural network*. Tex Stack Exchange. URL:https://tex.stackexchange.c (version: 2017-11-30).

[3] Janusz Kacprzyk and Witold Pedrycz. *Springer handbook of computational intelligence*. Springer, 2015.

[4] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[5] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[6] Standford. *Neural Networks*. URL: `https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/index.html` (visited on 16/11/2017).

[7] Neha Yadav, Anupam Yadav, and Manoj Kumar. *An introduction to neural network methods for differential equations*. Springer, 2015.