2 Introduction

Artificial neural networks are biologically-inspired computer models that enable a computer to learn from "observation". We indeed have a complex neural network in our brains; around 80 billion neurons, each connected to several thousand other neurons. That is what makes recognizing objects, voices, hand writings, formulating sentences, and performing everyday tasks seem so effortless.

On the other hand, it is extremely difficult to come up with a precise algorithm (step-by-step set of instructions) for a computer to recognize hand-written digits; you can imagine because of the infinite variations, the chances of getting it right most of the time is very low!

In a neural network approach, you create a system for the computer to learn by "observing", for example, by observing a large number of handwritten digits we can call "training samples". You let the computer infer the rules for the recognition. But, how is that actually done?

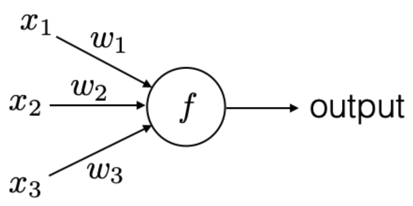

Figure 1: A Perceptron: The building block of an artificial neural network. The output is a function of the weighted sum of all input values (denoted here by x's). The weights are denoted by w's and the function by f.

The building block of the system you create is called the perceptron (see Fig. 1). It takes several (sometimes binary) inputs, and gives a single (sometimes binary) output. The rule for generating the output for a perceptron is that each input "receptor" has a *weight* associated with it, and the perceptron uses these weights to computes a weighted sum of all inputs. It then passes the sum, usually after adding a *bias* term (often denoted by b), through a function to determine the output. This function is called the *activation function* and can be a range of things, including a step function, a *sigmoid* function, $f(x) = 1/(1 + e^{-x})$, which can be thought of as the smooth version of the step function, a "rectified linear unit" (ReLU), or others. ReLU means the function will return x if $x > 0$ and 0 of $x \leq 0$.

The perceptron is the basic element of decision making. If the activation function is a step function and the output is a binary number (0 for no and 1 for yes), its process of generating the output is exactly the same as the weighing process we humans do for making binary decisions in life. Imagine a scenario in which your high school friend is in town and you are deciding whether to see him/her tonight. There are several factors contributing to your decision making: You have a lot of assignments and seeing him/her means you have to stay up later tonight to finish your homework. This factor carries a large negative weight! You really miss him/her and this may be your only chance to see him/her in a long time. This factor carries a positive weight. It is raining and the weather is not nice so you cannot really hang out. This also carries a negative weight. You weigh all these factors and decide whether or not to go (1 or 0).

Of course, you are using your entire brain to make that decision and that is very different from the job of a simple perceptron. However, the idea is that by creating a network of these perceptrons in a computer model (like the network of neurons in our brain), adding them in *layers*, we should be able to make more sophisticated decisions with our artificial neural network (ANN).

We can program such a network using linear operations and conditions, so that given some input x, there is an output, or a set of outputs, but that is not the main point! "The idea is that we can devise a learning algorithm in which weights and biases [of our ANN] are automatically *tuned* to make better decisions."[3] This implies that the network is made aware of the correct decision for every set of input factors and that it is given the chance to adjust itself, i.e., its weights and biases. It is the job of the "supervised" (you, the programmer) to make sure the correct answers are provided to the model during this process, and hence, the name "supervised learning". However, the supervisor does not directly intervene in the the adjusting/optimizing/tuning of parameters. Neural networks can "learn" to solve problems, sometimes problems that are extremely difficult to directly program a solution for. The process of optimizing weights and biases, which you also implement, is referred to as "training".

Before we get started, I strongly recommend that you watch the amazingly useful video by 3Blue1Brown on what a neural network is at `https://www.youtube.com/watch?v=aircAruvnKk`, and perhaps read Chapter 1 (at least up to the first set of exercises) of Neural Networks and Deep Learning by Michael Nielsen, which is available for free in an online format at `http://neuralnetworksanddeeplearning.com/`.