UNIVERSITETET
I OSLO

Project 2: Building a Neural Network code

Applied Data Analysis and Machine Learning

UiO (FYS-STK4155)

Pietro PERRONE

pietrope@uio.no

November 10, 2025

GitHub link to the project repository:

https://github.com/p-perrone/UiO_MachineLearning/tree/main/ml_project1

Link to DeepSeek chat:

https://chat.deepseek.com/share/h2ifare1m1c31ud8vf

Abstract

content...

Contents

1 Introduction **2**

2 Theory and methods **2**

2.1 The universal approximation theorem . . . 3

2.2 Structure of a Neural Network . . . 3

2.2.1 Activation functions . . . 3

2.2.2 The Feedforward algorithm . . . 3

2.2.3 The Backpropagation algorithm . . . 3

2.2.4 Training the Neural Network . . . 3

2.3 Cost functions . . . 3

1 Introduction

An Artificial Neural Network is a computational model that emulates the functioning of human brain, in the way it can process several informations in parallel, resulting in a form of "intelligence" (Wang, 2003).

A typical Neural Network (NN) is represented in Figure 1. It generally consists of connected units, called, *nodes* or *neurons*. Every node receives a specific *signal*, *i.e.* a real number, from its connected node. Nodes can be regrouped in specific layers, and the signal travels from the input layer to the output layer, passing through an arbitrary number of *hidden layers* (Bishop, 2006). Before being transmitted from one layer to the following one, the signal is modulated by an non-linear function, called *activation function*, which can be specified for each layer. Formally, a NN transforms an input vector $\mathbf{x} \in \mathbb{R}^d$ into an output vector $\mathbf{y}$ through successive linear transformations, where every node is multiplied by *weights*, and activation functions (Goodfellow et al., 2016). The weights are the parameters used in a NN for approximating the target function or dataset. The multiplication of each node's value is usually followed by the addition of a real number (a *bias*), in order to avoid the transmission of zero-signals throughout the network. A cost function is usually computed at the end, in order to assess the agreement between the target and output prediction of the NN.

The simplest possible neural network is given by the *perceptron model*, conceived by Rosenblatt, 1958. It is a type of linear classifier, that takes binary inputs, weights these by real numbers and yields a binary output. More complicated neural networks can generally support a higher number of layers and nodes, accept a multidimensional input and yield a non-binary output. In this project, particularly, we focus on so called *multilayer perceptrons* (MLPs). An other type of neural network can be the *feed-forward* neural network (FFNN). In this model, the signal is transmitted always in a single direction, forward through the layers, from the input to the output. If each node in a layer is connected to all the

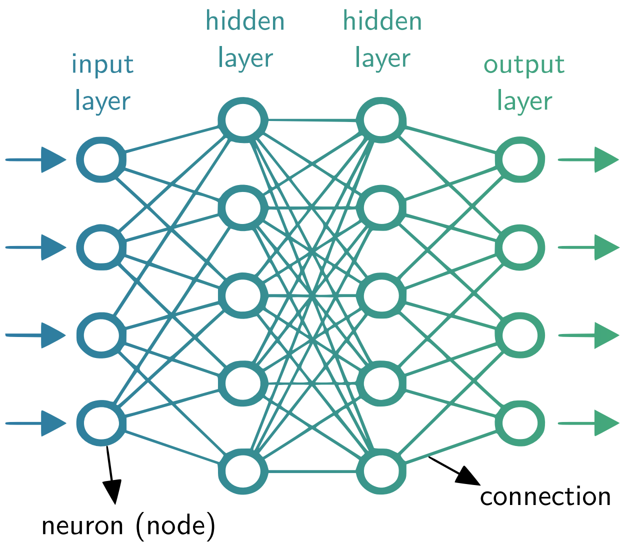

Figure 1: Neural Network | A typical structure.

other nodes in the following layer, this correspond to a *fully connected* FFNN (Haykin, 1994). The operation by which the weights and biases are optimized is called *backpropagation*, that consists of calculating the gradients of the cost function with respect to weights and biases starting from the last layer; a training can then be performed in order to minimize this gradients and find the optimal values of these parameters, using optimization algorithms such as gradient descent (Goodfellow et al., 2016).

The aim of this project is to build a NN code in Python, that could perform both forward feeding and backpropagation. We will test this model initially on a simple 1-dimensional dataset, given by a cluster of samples computed with the Runge function for an input array $\boldsymbol{x} \in [-1, 1]$. Several types of activation functions and layers depths will be tested.

A more complicated fit will be then performed on a dataset of handwritten digits, the MNIST-784. This consists of 70 000 digits samples, each of size 28×28 pixels. The objective will be to train the coded NN to make it recognize and classify correctly these digits. An analysis of the goodness of such classification will also be performed.

2 Theory and methods

2.1 The universal approximation theorem

The cornerstone of the Neural Networks theory in machine learning is the *universal approximation theorem*. The formulation by (Cybenko, 1989) states that:

Theorem 2.1. *Let σ be any continuous sigmoidal function such that:*

$$\sigma(z) = \begin{cases} 1 & z \rightarrow \infty \\ 0 & z \rightarrow -\infty \end{cases}$$

Given a continuous and deterministic function $F(\boldsymbol{x})$ on the unit cube in d-dimensions $F \in [0,1]^d$, $x \in [0,1]^d$ and a parameter $\epsilon > 0$, there is a one-(hidden) layer neural network $f(\boldsymbol{x};\boldsymbol{\Theta})$ with weights and biases $\boldsymbol{\Theta} = (\boldsymbol{W}, \boldsymbol{b})$ and $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, for which

$$|F(\boldsymbol{x}) - f(\boldsymbol{x};\boldsymbol{\Theta})| < \epsilon \; \forall \boldsymbol{x} \in [0,1]^d.$$

In other words,

2.2 Structure of a Neural Network

As stated previously, the most generalized structure of a typical neural network consists of an input layer, a given number of hidden layers and an output layer. In every layer l each node is modulated by the weights and biases, which corresponds to the parameters of this model:

$$\boldsymbol{\Theta}^{(l)} = (\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}) \tag{1}$$

and scaled by and activation function.

In other words, any continuous function $y = F(\boldsymbol{x})$ supported on the unit cube in d-dimensions can be approximated by a one-layer sigmoidal network to arbitrary accuracy and errors.

2.2.a Activation functions

2.2.b The Feedforward algorithm

2.2.c The Backpropagation algorithm

2.2.d Training the Neural Network

2.3 Cost functions