Universitat Autònoma de Barcelona

Degree Thesis

Improvements of Deterministic Processes through Neural Networks

Author:
Gerard Lahuerta Martín

Supervisor:
Dr. Lluís Alsedà Soler

A thesis submitted in fulfilment of the requirements for the degree of Computational Mathematics and Data Analytics in the

SCIENCE FACULTY

June 2024

Declaration of Authorship

Abstract

Acknowledgements

I would like to express my gratitude to my family for their support throughout my thesis, degree, and my entire life.

I am especially grateful to my mother, Maria Montserrat, for her encouragement and guidance in facing the challenges of this thesis and my career.

Thanks to all the peoples that treatme as I was part of their family.

I also appreciate the encouragement and support of my colleagues.

Finally, I want to give a special mention to Dr. Lluís Alsedà for guiding me through the thesis and being an inspiration.

Thank you all for everything.

Contents

List of Figures

Preface

Introducción

1 Neural Networks

A neural Network is made of individual and independent elements connected between them, passing and managing the information through the network formed.
In this thesis we will focus on one of the simplest networks, a multilayer perceptron, to test the different methods of optimization.

1.1 Multilayer Perceptron and Perceptron neuron

One of the simple Neural Networks to analyse is the Multilayer Perceptron[1].
It was first proposed by Frank Rosenblat[2] in 1958 (nevertheless its approach did not learn either produce accurate results).

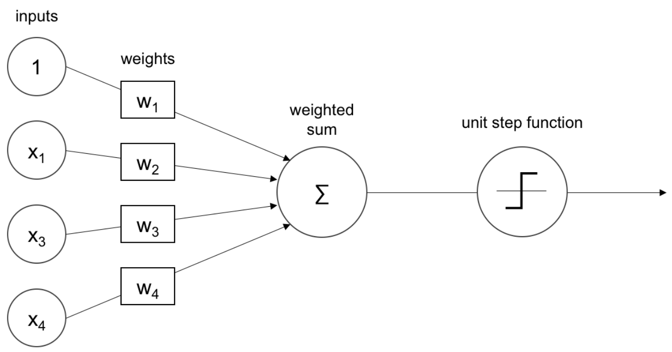


Figure 1: Schema of the Perceptron neuron

The Neural Network is made by individual neurons called *Perceptrons*. The neurons are splitted into input, weight and activation functions.
Nevertheless, the most important part of the neuron, and that determines significantly the capabilities of the neuron, is the activation function (which returns the output of the neuron).

The traditional activation function used in the Multilayer Perceptron is the Sigmoid:

$$f(x) = \frac{1}{1+e^{-w\cdot x}}, \text{ where: } x, w \in \mathbb{R}^n$$

The Multilayer Perceptron Topology can be splitted into layers of three types:

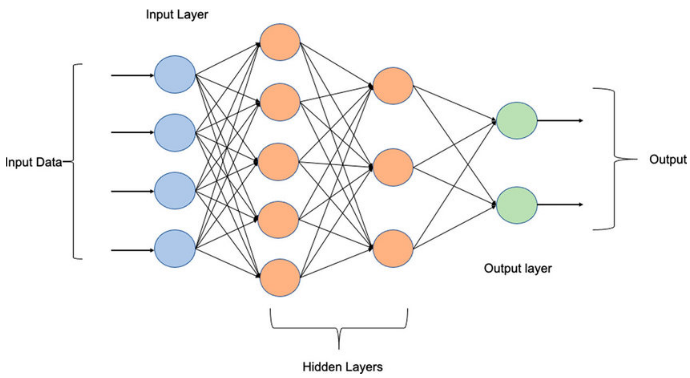


Figure 2: Schema of the Multilayer perceptron

- **Input Layer**: The initial set of neurons of the Multilayer Perceptron.
- **Output Layer**: The final set of neurons of the Multilayer Perceptron.
- **Hidden Layers**: The set of neurons (in layers) between the input and output layers.

[1] To obtain more information about Multilayer Perceptron functionability check the following link
[2] Frank Rosenblat, psychologist and father of deep learning, check its biografy.

1.2 Why Neural Networks works

It is important to remark that what gives the capacity of model any system is because we can express everyproblem as a function (no matter if its a classification tasck, probability function, or prediction, regression function).
The association between a tasck and a function allow to apply the **Universal Approximation Theorem**.

- **Definition:** Universal Approximation Theorem.

For any function $f : \mathbb{R}^n \to \mathbb{R}^m$, with $m, n \in \mathbb{N}$, and a subset $D \subset \mathbb{R}^n$ where f is continuous at all D, $\exists \{(w_i, b_i, c_i)\}_{i=0}^k$ that:

$$f(\vec{x}) - \lim_{k\to\infty} \sum_{i=0}^{k} c_i \sigma \left(w_i^T \cdot \vec{x} + b_i \right) = 0$$

Where $w_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, $c_i \in \mathbb{R}^m$, $\vec{x} \in D$ and σ the sigmoid function.

The parameters $\{w_i, b_i, c_i\}$ are associated (respectively) with the weights, bias and scale factor of the i-th neuron.
Neural Networks essentially use this theorem with a limit number of sigmoid (equal at de number of neurons), consecuently an error of approximation is given.

$$f(\vec{x}) - \sum_{i=0}^{k} c_i \sigma \left(w_i^T \cdot \vec{x} + b_i \right) = \epsilon, k \in \mathbb{N}$$

Modeling systems the information of the function shape/tendency is limited or none, making only possible study the problem using data values.
This cases hare handle using the *square-norm* metric (or *Mean Squared Error*).

- **Definition:** Square-norm:

Being $f : \mathbb{R}^n \to \mathbb{R}^m$ function followed by the system. Being $g : \mathbb{R}^n \to \mathbb{R}^m$, with $m, n \in \mathbb{N}$, as $g(\vec{x}) = \sum_{i=0}^{k} c_i \sigma \left(w_i^T \cdot \vec{x} + b_i \right)$.
Given a dataset $B = \{(\vec{x_i}, \vec{y_i})\}_{i=0}^N$, with $N \in \mathbb{N}$, where $\forall (\vec{x_i}, \vec{y_i}) \in B, \vec{y_i} = f(\vec{x_i})$.

$$\Delta^2 = \frac{1}{N} \sum_{i=0}^{N} \left(\vec{y_i} - g(\vec{x_i}) \right)^2$$

Decreassing the error Δ implies reducing the error ϵ because as Δ decrease the model improves its approximation whereas ϵ decrease.
As the more neurons are added into the Network, the approximation and the training time increases. This increase in training time for the model in cases is not worth it as it represents a minimum improvement of the model.
Moreover, the Backpropagation[3] method used to train Neural Networks have a complexity that increase the process time exponentially to the number of neurons.
Therefore, trying to obtain the ideal topology for solving a problem is a dificult tasck that in many cases resides about trial error.

[3]The Backpropagation method is an algorithm to modify the internal parameters of a Neural network by using the chain rule, more information in the following link

2 Neural Networks topology

The main objective of the study is obtaining a more efficient process using neural networks than the traditional ways that has been used at time.
Doing this requires, as has been explained before, a non undertand process to obtain an eficient neural network.
Therefore, an observation has done while approaching this problem that could partial solve that non-undertand of what really is doing the neural network and deduce the optimal architecture.

2.1 Hypotesis and Proposal

As has been said before, approaching this problem an intuitive idea apeared.

- **Proposal:** If there is enough information about the system to model, exists an eficient neural network which architecture can be obtained using rules.

The idea proposed is influenced by the following observations:

1. Every deterministic process is a convination of restriccions that can be represented as a function.
2. Every function can be splitted in domaind which its behavior can be categorized as: periodical, irrational, polinomical.
3. The periodical, polinomical and irrational functions are continuous in "all"[4] domain.
4. All continuous functions can be approximated using the **Universal Approximation Theorem**.

Moreover, if a function does not follow one of this types of functions, it can be splited by more intervals until each of them can be aproximated by one of this types or even can be approximated using *Fourier Series* or *Taylor series* allowing it to be modeled by periodical and polinomical functions.
Therefore, this proposal can be verified reducing it into corroborate the following points:

1. Each category of functions mencioned can be modeled using a particular architecture of neural network.
2. Exists a determined minimum number of data registers were the neural network can be trained.

To verify the proposal a topological, efficiency and preccision studies is required.

[4] In case of irrational functions such as $\sqrt{x}$, it can be interpretated as $f(x) = \begin{cases} 0, x < 0 \\ \sqrt{(x)}, x \geq 0 \end{cases}$

2.2 Study of deterministic process

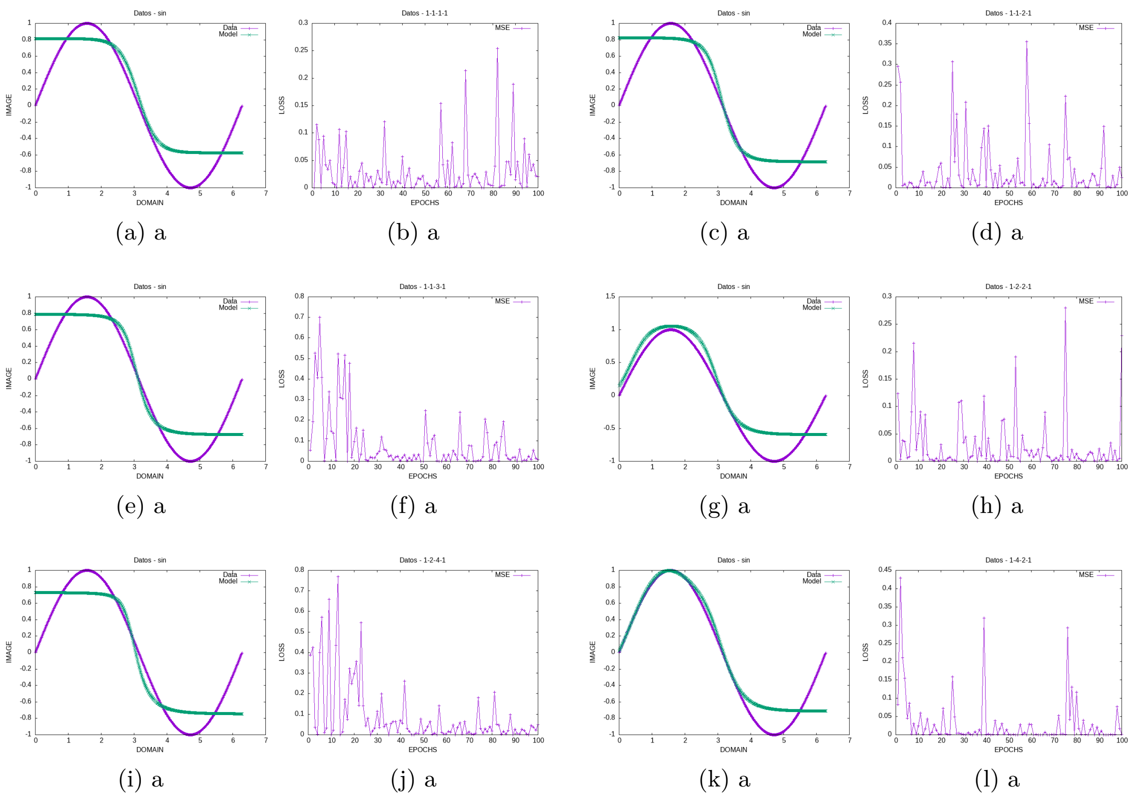
To start, lets assume the following axioms to simplify the study and give some feedback to make the correlations to werify or dimsmiss the proposal.

1. All periodically functions can be approximated as a convination of sinus functions.
2. All polinomical functions can be approximated as a convination of lineal functions.
3. All irrational functions can be approximated as a convination of irrational functions.

All point can be proven using *Taylor Series* and *Fourier Series*. Therefore, the study will be reduced into take patterns of the architectures used to approximate the functions $\sin(x), \sqrt(x), x^2$.
This paper use $I-H-N-O$ nomenclature to represent the architecture of the neural network, where the letters I, H, N, O represents the number of inputs, the hidden layers, the number of neurons for each hidden layer and the number of outputs respectively.

2.2.1 Study of the *sin* function

To test if exists an architecture that can approximated efficiently the sin function in the real plane, will be study to approximated the function in the interval $[0, 2\pi]$ using a fixed number of iterations with diferents architectures.



(a) a (b) a (c) a (d) a

(e) a (f) a (g) a (h) a

(i) a (j) a (k) a (l) a

Figure 3: q

The results shown in Figure **??** where obtained fixing 100 epochs and trained using the stochastic gradient descend method with a learning rate of 0.1 trying to minimice the *Square-norm* error (*MSE*).

2.2.2 Study of the *parabolic* function

2.2.3 Study of the *squared root* function

2.3 Observations of the results

2.4 Viability of approximation discontinuous functions

2.4.1 Study of Jump discontinuity

2.4.2 Study of Asimptotyc discontinuity

2.5 Conclusions

References

[1] Wikipedia, *Multilayer perceptron*,
https://en.wikipedia.org/wiki/Multilayer_perceptron

[2] Wikipedia, *Perceptrón*,
https://es.wikipedia.org/wiki/Perceptr%C3%B3n

[3] Wikipedia, *Activation function*,
https://en.wikipedia.org/wiki/Activation_function

[4] Wikipedia, *Frank Rosenblatt*,
https://es.wikipedia.org/wiki/Frank_Rosenblatt

[5] Wikipedia, *Activating Function*,
https://en.wikipedia.org/wiki/Activating_function

[6] Wikipedia, *Artificial neural network*,
https://en.wikipedia.org/wiki/Artificial_neural_network

[7] Wikipedia, *Neural network*,
https://en.wikipedia.org/wiki/Neural_network

[8] Wikipedia, *Universal Approximation Theorem*,
https://en.wikipedia.org/wiki/Universal_approximation_theorem

[9] Wikipedia, *Mean Squared Error*,
https://en.wikipedia.org/wiki/Mean_squared_error

[10] Wikipedia, *Mean Absolute Error*,
https://en.wikipedia.org/wiki/Mean_absolute_error

[11] Wikipedia, *Cross-Entropy*,
https://en.wikipedia.org/wiki/Cross-entropy

[12] Wikipedia, *BackPropagation*,
https://en.wikipedia.org/wiki/Backpropagation

[13] Dantzig, G.B. y P. Wolfe, «Decomposition principle for linear programs», *Operations Research*, **8**, págs. 101–111, 1960.