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).

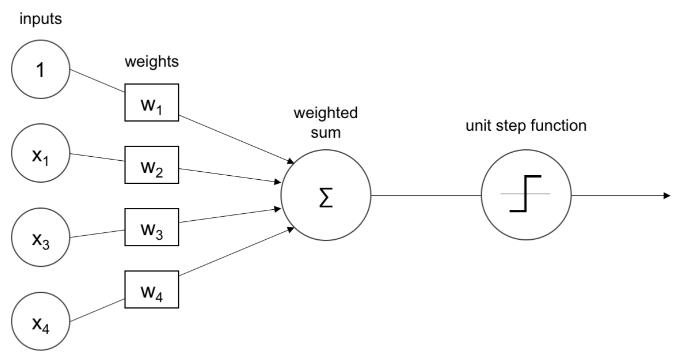


Figure 1: Schema of the Perceptron neuron

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:

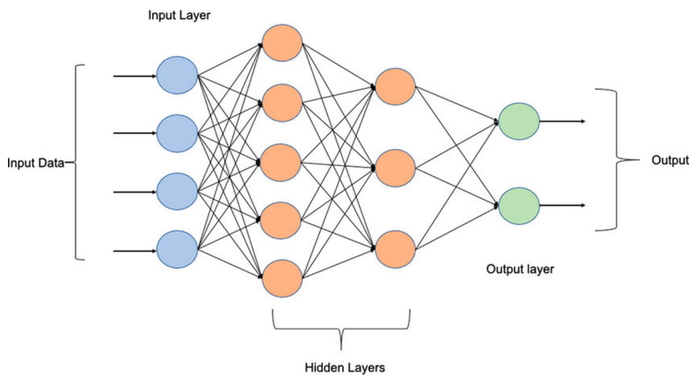


Figure 2: Schema of the Multilayer perceptron

- **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 $\forall(\vec{x_i}, \vec{y_i}) \in B, \vec{y_i} = f(\vec{x_i})$.

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

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

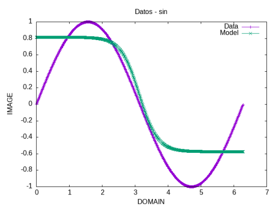
To start, lets assume the following axioms to simplify the study and give some feedback to make the correlations to werify or dimsmiss the proposal.

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.

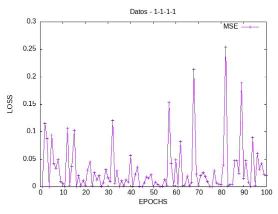
All point can be proven using *Taylor Series* and *Fourier Series*. Therefore, the study will be reduced into take patterns of the architectures used to approximate the functions $\sin(x), \sqrt(x), x^2$.
This paper use $I - H - N - O$ nomenclature to represent the architecture of the neural network, 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

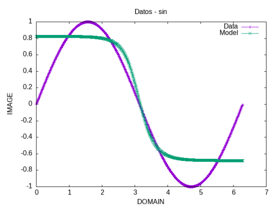
To test if exists an architecture that can approximated efficiently the sin function in the real plane, will be study to approximated the function in the interval $[0, 2\pi]$ using a fixed number of iterations with diferents architectures.



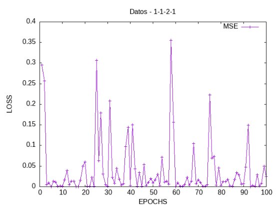
(a) Model 1-1-1-1



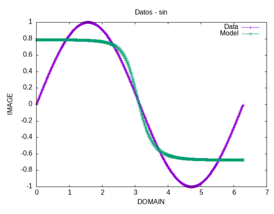
(b) MSE 1-1-1-1



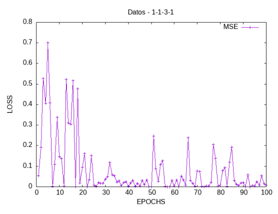
(c) Model 1-1-2-1



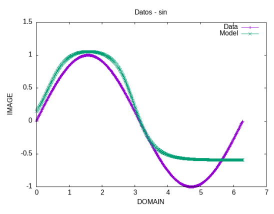
(d) MSE 1-1-2-1



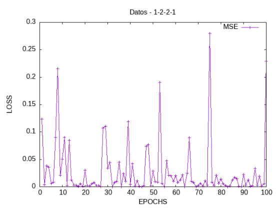
(e) Model 1-1-3-1



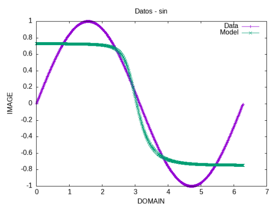
(f) MSE 1-1-3-1



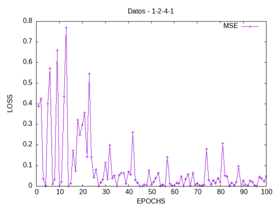
(g) Model 1-2-2-1



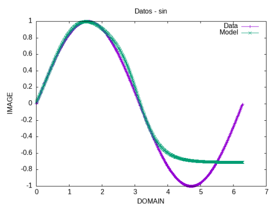
(h) MSE 1-2-2-1



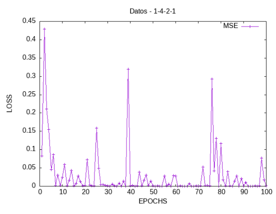
(i) Model 1-2-4-1



(j) MSE 1-2-4-1



(k) Model 1-4-2-1

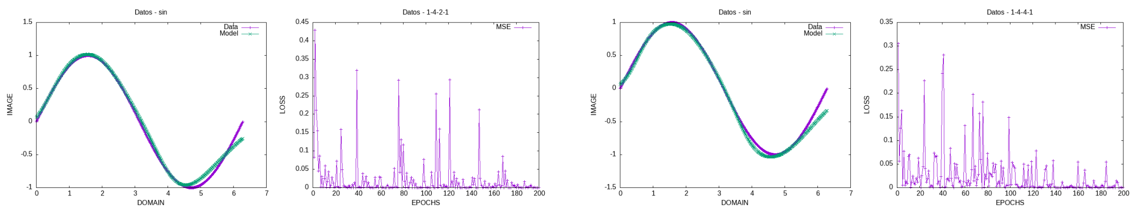


(l) MSE 1-4-2-1

Figure 3: Results of diferents architectures approximating the function *sin*

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*).

As can be seen in the Figure, almost all architectures does not approximate the sin function properly, despite the architectures *1-2-2-1* and *1-4-2-1*.
If the experiment is repeated changing the epoch number (insted of 100 change to 200) the results are almost similar (with more precission in some parts but no improvement of modelation).
However, the architectures *1-4-4-1* and *1-4-2-1* give an augment of precission significally important.



(a) Model 1-2-4-1 (b) MSE 1-2-4-1 (c) Model 1-4-4-1 (d) MSE 1-4-4-1

Figure 4: Architectures that improves the modelation of *sin* function

The approximation done by these architectures is so significant that allow to observe and conclut that the model that approximates the best the sin function if 1-2-4-1.
This conlusion is obtained because is the minimum architecture that can approximate the function, that can "learn"[5].
Moreover, it can be an evidence of the following proposal:

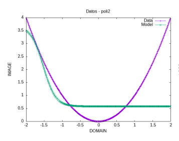
- **Proposal:** To approximate any periodical function in real plane needs minimum 2 neurons in each of the $2 \cdot K$ hiddden layers, where K is sum of relatives extrems and point of inflection of the function.

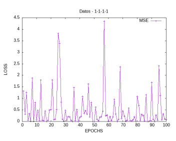
This proposal is obtained observing that every layer of the neural netwrok is modeling one interval (delimited by the extreams of the function and inflection points) and for perform that approximation nead at least 2 sigmoids.

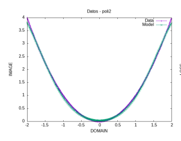
2.2.2 Study of the *parabolic* function

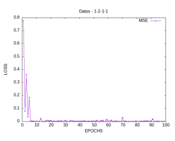
The same experiment have been done with the function x^2 (and also the x^3 to have more results to compare) with fixed epochs, learning rate and domain.
Concretly, the values of epochs, learning rate and domain used are, respectively, 100, 0.1 and $[-2, 2]$.
The results obtained for polinocmical functions are shown in **??**.
It shows how the most efficient architecture (which give the better performance as modeling as precission) is the $1-2-1-1$.
This architecture is similar obtained before with the periodical functions but is not the same, which was one of the axioms suposed in the proposal.
Therefore, a last study will be done to obtain the architecture need to model the irrational functions.

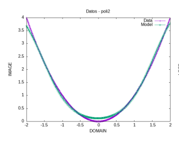
[5]Learn in this tearm refears as if the training process allow to reduce the loss (also named error) near 0 with a significally decrease throughout the epochs.

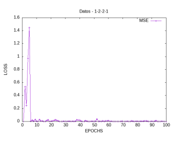

(a) Model 1-1-1-1

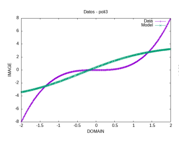

(b) MSE 1-1-1-1

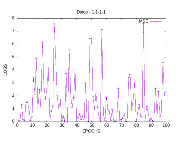

(c) Model 1-2-1-1

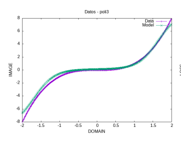

(d) MSE 1-2-1-1

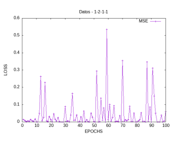

(e) Model 1-2-2-1

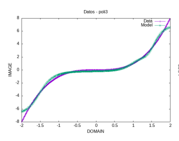

(f) MSE 1-2-2-1


(g) Model 1-1-1-1


(h) MSE 1-1-1-1


(i) Model 1-2-1-1


(j) MSE 1-2-1-1


(k) Model 1-2-2-1

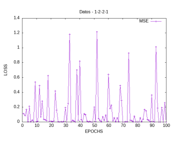
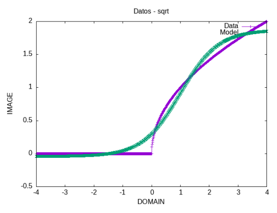

(l) MSE 1-2-2-1

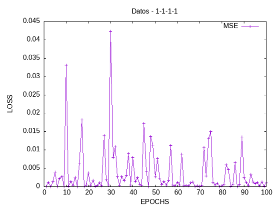
Figure 5: Results of diferents architectures approximating the functions x^2 and x^3

2.2.3 Study of the *squared root* function

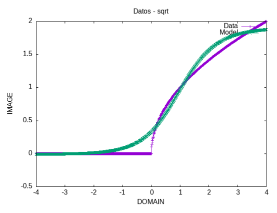
Finally, reproducing the experiment for the function $\sqrt{x}$, the results obtained are shown in **??**.



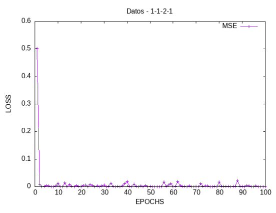
(a) Model 1-1-1-1



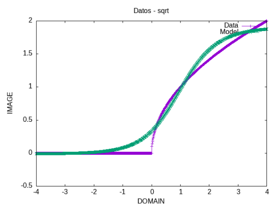
(b) MSE 1-1-1-1



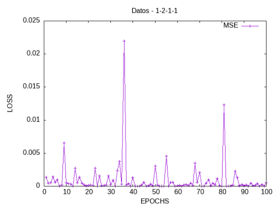
(c) Model 1-1-2-1



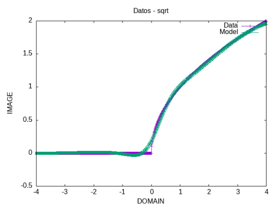
(d) MSE 1-1-2-1



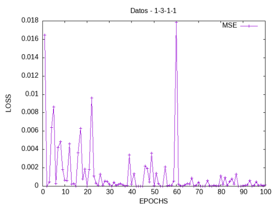
(e) Model 1-2-1-1



(f) MSE 1-2-1-1



(g) Model 1-3-1-1



(h) MSE 1-3-1-1

Figure 6: Results of diferents architectures approximating the function $\sqrt{x}$

In this study, the only parameters that where changed respect the previous experiments was the domain (which was increased to the interval $[-4, 4]$).
As it can be observed, the architecture that approximate the best the function is $1-3-1-1$. Moreover, comment that the intutition of the inclusion of irrational funccions such as the $\sqrt{x}$ in the axioms of the proposal was made to use the property of having a function similar to *ReLU*[6].
This inclusion of the function $\sqrt{(x)}$ in spite of the *ReLU* was made thinking that was a better option and the least "forced" option of any other function that could give that

[6]The *ReLU* function is widely use as an activation function which is deffined as $f(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases}$

similar property. However, this text between other similar functions was not made so could be interesting to expand this stuidy adding and experiment with other functions.

2.3 Viability of approximation discontinuous functions

It is important to have in mind that splitting functions in domains can develop in discontinuities which does not allow the use of the **Universal Approximation Theorem**. Nevertheless the application of neural networks in tascks such as classification or regression in many areas, with no information of the function that are approximating, using activation functions diferent from the *sigmoid* that give (apparently) correct results give the idea that possibly the theorem could be expanded and (in some cases) have no need of the continuity restriction.

- **Proposal:** If a function has discontinuities diferent from asimptotyc, the **Universal Approximation Theorem** can be used to give an approximation with error 0 when the number of sigmoid tends to infinity.

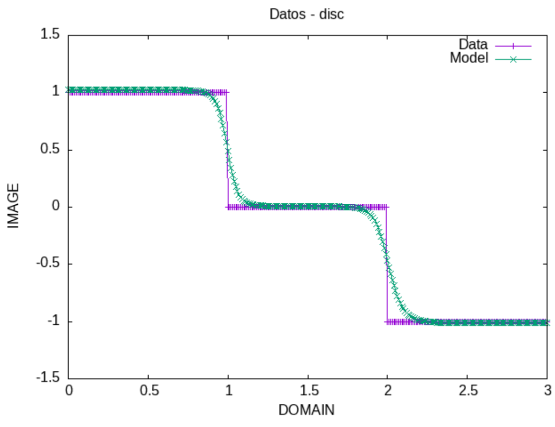
This proposal can be sustain by the ability of sigmoid functions to approximate jump discontinuities. However, an experiment with discontinuous functions have done to give evidence of this proposal.

2.3.1 Study of Jump discontinuity

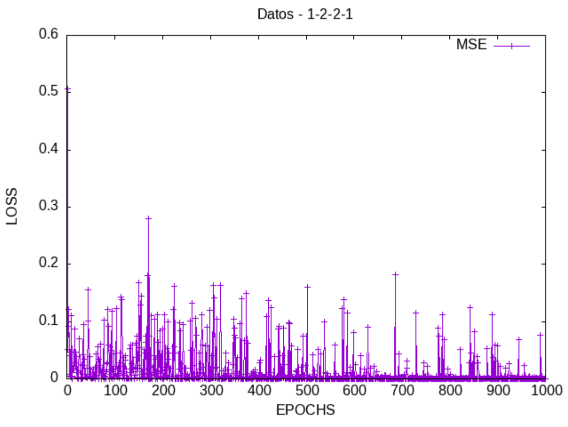
In this experiment a simple architecture of 1−2−2−1 with high amount of epochs (concretly 10000) and fixed learning rate of 0.1 have been used to approximated the function:

$$f(x) = \begin{cases} 1, x \leq 1 \\ 0, x \in (1, 2] \\ -1, x > 2 \end{cases}$$

The results of approximating the function in the interval $[0, 3]$ are shown in **??**.



(a) Model



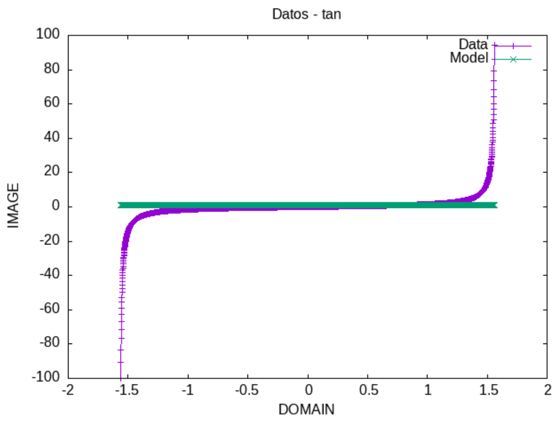
(b) MSE

Figure 7: Architecture $1-2-2-1$ approximation to a jump discontinuity function

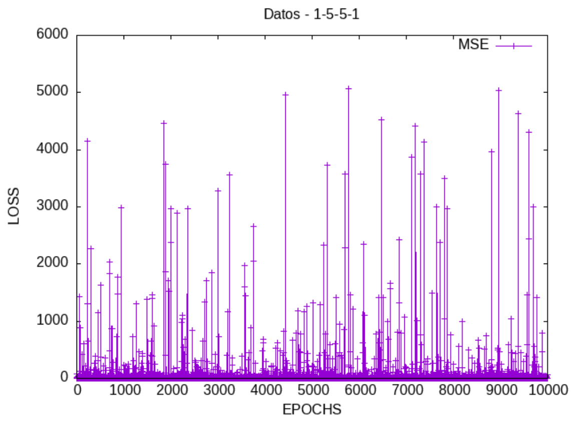
As can be seen in the figure **??**, using a limited model with enough number of epochs the approximation is quiet acceptable. Therefore the use of *sigmoid* functions in approximate splitted functions with jump discontinuities can be use at least in some cases.

2.3.2 Study of Asimptotyc discontinuity

However, the application of the **Universal Approximation Theorem** in discontinuities such as asimptotyc can give problems that impossibilates the modelation using sigmoids.
Repeating the experiment done with the jump discontinuity function, but with a more prowerfull architecture, to approximate the function $\tan x$ in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$ give us an idea of this problem.
The results obtained are shown in **??**.



(a) Model



(b) MSE

Figure 8: Architecture $1-5-5-1$ approximation to $\tan x$ function

As can be observed the neural network is limited and cannot approximate properly the function. However, the use of other activation function such as the *ReLU* mentioned before can give a better performance.
In case of using the *ReLU* function, it give another problem that can be dificult to handle, the **Universal Approximation Theorem** does not ensure that the resulting function approximates with a limited error the system.
Moreover, the use of the *BackPropagation* method and its dependency with the *Newtons Method* (the *Steepest Gradient Descend*) to obtain zeros of the *cost function* in the training process, does not ensure that the solution it arrive is the best or, even doe, if it is correct in all cases (and we can be blind in situations were the network was not tested).

2.4 Observations of the results

To sump up all the results and conlusions obtained in the studies we can conclud if theres enough evidence of the proposal.

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

To verify the viability of the proposal needed to acomplish the following conditions:

1. All periodically functions can be approximated as a convination of sinus functions.
The first contidion can be proven usen *Fourier series*
However, its important to mention that a periodical function such as $\tan x$ can not be approximated due to the asymptoties. Therefore could exist a periodical function that can be modeled with sin functions because it requeres that can hadle asymptoties.

2. All polinomical functions can be approximated as a convination of lineal functions.
Such as the first condition, the second one can be proven using *Taylor Series*, not having a possible counter part that do not apply.

3. All irrational functions can be approximated as a convination of irrational functions.
The point 3 can be explained using the condition that we are tryiung to approximate a function un a certain interval. Therefore iy can be obtained the following deduction:

- **Deduction:** Approximation of irration functions.
Being $f_n : \mathbb{R} \to \mathbb{R}$, $g : \mathbb{R} \to \mathbb{R}$ where $f_n(x) = \sqrt[n]{x}, \forall n \in \mathbb{N}/n > 1$ and $g(x) = \sqrt{x}$.
$$\exists c, d \in \mathbb{R}/\forall x \in [a, b], f_n(x) - g(c \cdot x + d) < \epsilon$$
Where $\epsilon \in \mathbb{R}^+$ is the error and $\{c, d\}$ are parameters that ajust g to approximate f_n in the interval $[a, b] \cup \mathbb{R}$.

Nevertheless, mention that when n (the degree of the root) is even the interval needs to be in the set $\mathbb{R}^+$. Moreover, this deduction can handle situations when the set of functions f_n are defined as $f(x)_n = \alpha \sqrt[n]{x} + \beta$ where the parameters $\alpha, \beta \in \mathbb{R}$
This deduction can also be demostrated using tangency:

- **Demostration:** Approximation of irration functions.
Being $f_n : \mathbb{R} \to \mathbb{R}$, $g : \mathbb{R} \to \mathbb{R}$ where $f_n(x) = \sqrt[n]{x}, \forall n \in \mathbb{N}/n > 1$ and $g(x) = \sqrt{x}$.
Being $P = (p_0, p_1)/p_0 = \frac{a+b}{2}, p_1 = f_n(p_0)$, $\exists c, d \in \mathbb{R}/g(c \cdot p_0 + d) = p_1$. Therefore,
$$\exists(\alpha, \beta) \subset [a, b]/p_0 = \frac{\alpha + \beta}{2} \text{ where } \forall x \in (\alpha, \beta), f_n(x) - g(c \cdot x + d) < \epsilon$$
$\epsilon \in \mathbb{R}^+$ the error.

These demostration could be use for every function, not limiting them only for irrational functions.

4. The mencioned functions can be modeled using a particular architecture.
The studies about the differents architectures of the diferents functions ($x^2, \sin x, \sqrt{x}$) shows that exist a diferenciable minimum architecture that approximate the best each function.

5. Exists a determined minimum number of data to train the neural network.
   That point is controversial. In all the studies perfom the number of data available where diferents with at least 200 registers. However, the rellevant part about this is that it really needs a deterministic process that generates the dataset.
   Te minim information that is required is the determinmistic process intrinsecly. If there is no deterministic process the information needed will not be enough despite how many it could recap.
   This is consecuence of the backpropagation method and its dependency in the *Steepest Gradient Descend*. If there is no "correct" model to compare and interpretate the functionability of the network it is not possible to "extract theoretically the best architecture".
   Hence, there is no "minim number of samples" despite there is a necessary existance of a (deterministic) *non-black box*[7] method.

Concluding the observations, the studies give the enough evidence of the proposal.
To give more evidence of that proof, an experimental process will be done to extract th best minimal architecture that approximate a deterministic process.
Moreover, a efficiency test will be also perfomed to ensure that neural networks in colaboration with the information that gives the deterministic process improves the computation time it needs.

3 Neural Network for arabic and roman numbers

The idea about that topic is the following, any natural number (normally written in the arabic form) can be expressed as a roman number. This process is a sequence of fixed rules that repeated in steeps give the result (it is a deterministic process).
As is a deterministic function ensure the existence of a function that models the process, but this function is unkown.
The main objective is, with the observation and the proposal done before, create a neural netqwork that approximates this unkown function.

3.1 Preprocess of data and reduction of variables

The main problem when tracting with roman numbers is that is a sequence of diferents characters that can not be handle easily. Therefore a simplification was perfomed.

- **Application:** Interpretation of Roman numbers.
  Being the set $C_n = \{c_0, \cdots, c_{n-1}\}$ where $\forall c \in C_n$ is a roman number.
  Lets define that each $\forall c \in C_n$ are formet by two characters: $c = (a, b)$, where b is a letter of the roman number system, the set $\{I, V, X, C, \cdots\}$, and a the element in the roman number system just one or two positions before which is b or $\varnothing$.
  Using this asumption we can write very needed number, for example the element
  $c = (I, V) \equiv 4$ and the sequence $C_3 = \{(\varnothing, X), (\varnothing, X), (I, V)\} \equiv 24$.

With this interpretation and definition of the roman numbers set and its properties, another application can be done.

[7]A *black box* model is a process where the internal operations and functionability of it can not be interpretated partially or at all, giving the aspect of being a non undertanable or/and verified model.

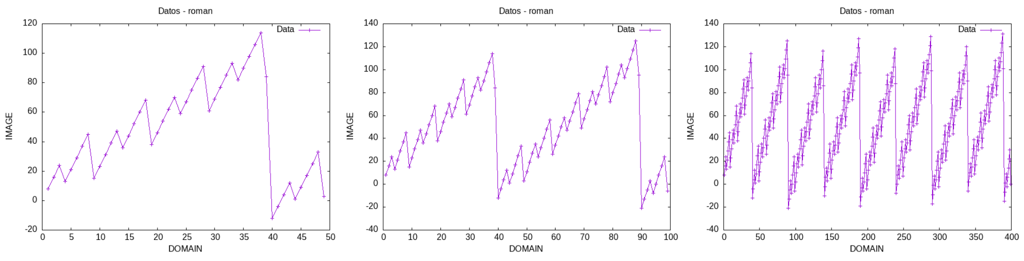
- **Application:** Reduction of liberty degrees function.
  Using the previous definition of the *Roman numbers set*, it is possible to create a function $f : \{C_n\} \to \mathbb{Z}, \forall n \in \mathbb{N}$.
  Before defining this function its esentially assign to each letter of the Roman number system a number (for example its position in the alphabet starting with $A \equiv 1$ and defining $\varnothing \equiv 0$).
  This method allow us to represent the example sequence used before as a set of numbers:

$$C_3 = \{(\varnothing, X), (\varnothing, X), (I, V)\} \equiv \{(0, 24), (0, 24), (9, 22)\} \equiv 24$$

  Now we can create the function $f : C_n \to \mathbb{Z}, \forall n \in \mathbb{N}$ as:

$$f(C_n) = \sum_{i=0}^{n-1} b_i - a_i, \text{ where } c_i = (a_i, b_i)$$

This application allow to represent each roman number in one arabioc number (but does not ensure that this process satisfy the injection property).
The important about this process is that we can represent each natural number expresed in arabic with another number (in the integers) in arabic which is equivalent as, at least one, roman number.
In consecuence, it can be represented graphically in $\mathbb{R}^2$. This representation is shown in ??.



(a) 1 to 49 (b) 1 to 99 (c) 1 to 399

Figure 9: Distribution of the associated arabic number to the roman natural number

The function distribution that appears is the result of applying the function defined to the deterministic rules of roman numbers, therefore there is no knowledge of which is the function behind this process.
However is an ideal case to use the experimental observation and try to obtain with the dataset given by the deterministic method a function that approximates the distribution of the dataset.

3.2 Analisis of the dataset and creation of the neural network

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.