UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

A RADIAL BASIS FUNCTION NEURAL NETWORK USING BIOLOGICALLY PLAUSIBLE ACTIVATION FUNCTIONS

ALBERT TARRÉS RIBALTA

Thesis supervisor: LUIS ANTONIO BELANCHE MUÑOZ (Department of Computer Science)

Degree: Bachelor Degree in Informatics Engineering (Computing)

Thesis report

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

27/06/2023

Acknowledgements

The realization of this project could have not been possible without the support of my director, Luís Antonio Belanche Muñoz. Thanks for your help and suggestions during the development of the project.

I would also like to thank Joan Sardà Ferrer, who helped me with the planning and organization of the study.

Special thanks to my parents and my brother, who supported me at all times.

Abstract

This project focuses on the design, implementation and evaluation of Radial Basis Function Neural Networks (RBFNN), comparing the gaussian model with a new version using the Ricker Wavelet activation function. The shape of this wavelet has been observed in the signals of neurons from different parts of the human brain, often producing a negative (inhibitory) signal known as *lateral inhibition*. Two RBFNN models have been developed, incorporating Machine Learning (ML) and statistical techniques such as L2 regularization and the sigest algorithm for improved performance. Additional techniques, such as estimating an oversized k parameter and using AIC backward selection, are implemented to enhance efficiency.

In this study, the developed models are tested with datasets of different nature, evaluating their performance with synthetic and realistic data and measuring their results with problems of various levels of noise and difficulty. Furthermore, a comparison of the models is also made in order to observe which RBFNN performs better on certain conditions, as well as to analyze the difference in the number of neurons and the estimated smoothing parameter.

The experimental evaluation confirms the effectiveness of the RBFNN models, yielding accurate estimations and demonstrating their adaptability to problems of varying difficulty. Comparative analysis reveals that the Ricker model tends to exhibit superior performance in the presence of high levels of noise, while both models perform similarly under low noise conditions. These results suggest the potential influence of lateral inhibition, which could be explored further in future studies.

Keywords: *Radial Basis Function Neural Networks, Ricker Wavelet, regularization, sigest algorithm, noise, lateral inhibition, performance evaluation.*

Resumen

Este proyecto se centra en el diseño, la implementación y la evaluación de Redes Neuronales de Función de Base Radial (RBFNN), comparando el modelo gaussiano con una nueva versión que utiliza la función de activación Ricker. La forma de esta función ha sido observada en las señales de neuronas de distintas partes del cerebro humano, a menudo produciendo una señal negativa (inhibitoria) conocida como *inhibición lateral*. Se han desarrollado dos modelos de RBFNN, incorporando técnicas de Machine Learning (ML) y estadística como la regularización L2 y el algoritmo sigest para mejorar su rendimiento. También se implementan técnicas adicionales, como la estimación de un parámetro k sobredimensionado y la *AIC backward selection*, para mejorar la eficiencia.

En este estudio, los modelos desarrollados se prueban con conjuntos de datos de diferente naturaleza, evaluando su rendimiento con datos sintéticos y realistas, y midiendo sus resultados con problemas de varios niveles de ruido y dificultad. Además, también se realiza una comparación de los modelos para observar qué RBFNN funciona mejor en determinadas condiciones, así como para analizar la diferencia en el número de neuronas y el parámetro de suavizado estimado.

La evaluación experimental confirma la eficacia de los modelos RBFNN, proporcionando estimaciones precisas y demostrando su adaptabilidad con problemas de dificultad variable. El análisis comparativo revela que el modelo Ricker tiende a exhibir un rendimiento superior en presencia de altos niveles de ruido, mientras que ambos modelos tienen un rendimiento similar en condiciones de bajo ruido. Estos resultados sugieren la potencial influencia de la inhibición lateral, que podría ser explorada en más profundidad en futuros estudios.

Palabras clave: *Redes Neuronales de Función de Base Radial, Ricker Wavelet, regularización, algoritmo sigest, ruido, inhibición lateral, evaluación del rendimiento.*

Resum

Aquest projecte es centra en el disseny, la implementació i l'avaluació de Xarxes Neuronals de Funció de Base Radial (RBFNN), comparant el model gaussià amb una nova versió que utilitza la funció d'activació Ricker. La forma d'aquesta funció ha estat observada en les senyals de neurones de diverses parts del cervell humà, sovint produint una senyal negativa (inhibitòria) coneguda com *inhibició lateral*. S'han desenvolupat dos models de RBFNN, incorporant tècniques de Machine Learning (ML) i estadística com la regularització L2 i l'algoritme sigest per millorar el seu rendiment. També s'implementen tècniques addicionals, com l'estimació d'un paràmetre k sobredimensionat i la *AIC backward selection*, per millorar l'eficiència.

En aquest estudi, els models desenvolupats es posen a prova amb conjunts de dades de diferent naturalesa, avaluant el seu rendiment amb dades sintètiques i realistes, i mesurant els seus resultats amb problemes de diversos nivells de soroll i dificultat. A més, també es realitza una comparació dels models per observar quina RBFNN funciona millor en determinades condicions, així com per analitzar la diferència en el nombre de neurones i el paràmetre de suavitzat estimat.

L'avaluació experimental confirma l'eficàcia dels models RBFNN, proporcionant estimacions precises i demostrant la seva adaptabilitat amb problemes de dificultat variable. L'anàlisi comparatiu revela que el model Ricker tendeix a exhibir un rendiment superior en presència d'alts nivells de soroll, mentre que tots dos models tenen un rendiment similar en condicions de baix soroll. Aquests resultats suggereixen la potencial influència de la inhibició lateral, que podria ser explorada amb més profunditat en futurs estudis.

Paraules clau: *Xarxes Neuronals de Funció de Base Radial, Ricker Wavelet, regularització, algoritme sigest, soroll, inhibició lateral, avaluació del rendiment.*

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation of this project

The nodes of an Artificial Neural Network model the output results using an activation function [3]. Currently, several types of activation functions are used for different problems and circumstances. The most commonly used nowadays are the Ridge Activation Functions, which include the Rectified Linear Unit (ReLU) [27] and the Sigmoid functions [29]. However, Radial Basis Function Neural Networks (RBFNNs) make use of a different group of functions called Radial Basis Functions (RBFs). These are real-valued functions whose output is determined by the norm of the difference between the input and a given center [10]. At the present time, the most employed RBF is the Gaussian, although Multiquadric functions and other polyharmonic splines are often used as well [8].

In contrast, biological neurons exhibit a more complex radial pattern of activation [4] which makes them more flexible than the frequently used basis functions. Therefore, this project aims to design and train RBFNNs using this new observed radial pattern, exploring their potential in real data-science problems and conducting a comparison of their performance with the traditional models.

1.2 Objectives

As mentioned in the previous section, the main goal of this project is to design and train RBFN Networks using more biologically plausible activation functions, as well as developing a comparison study of their behavior in real data science problems.

In order to accomplish this objective, the project has been divided into different sub-objectives:

- Study the existing RBF Neural Networks and the different methods associated with this topic, which will be used throughout the project.
- Research and select the different biologically plausible activation functions that will be evaluated in this study.
- For each selected activation function, develop a high-performing and optimized Python implementation of a RBFNN based on state-of-the-art concepts.

- For each RBFNN model, analyze its behaviour with synthetic and real-world data, measuring the model's accuracy, its final number of hidden neurons and execution time, and other metrics of great relevance (explained in Section 6.2).
- Extract meaningful conclusions from the comparison and metrics obtained during the study. Specifically, determine the model that achieves superior performance, identify the model that requires fewer neurons to achieve comparable results, and explore other pertinent findings that contribute to the understanding and evaluation of the RBFNN models under investigation.

1.3 Structure of the document

Content in this project is organized in different sections. The first section serves as an introduction to the study, presenting the study's motivation followed by its main objectives and structure. In Section 2 Linear Models are introduced along with their different selection criteria and regularization strategies. In Section 3 an explanation about the main concepts of Radial Basis Function Neural Networks is given. This section includes a detailed explanation about the Training Phase in RBFNNs. Section 4 presents the biological neuron's phenomenon which inspired this study and links previous studies about this biological behaviour. In Section 5 the designed RBFNN architecture which will be used in this project is explained. Section 6 focuses on the conducted experiments, explaining the employed data and the selected metrics, and including the obtained results. The project's conclusions and further investigation proposals are given in Section 7.

2 Linear Models

A commonly used solution to approximate a function from a given set of noisy input-output pairs is a Linear Model. A Linear Model is a linear combination of a number of fixed parameterized basis functions (BF), where the coefficients are known as *weights* in the neural networks community. This model has a wide flexibility, and can be used to approximate many distinct functions by adjusting the function weights differently.

2.1 Supervised Learning

One of the most common problems in statistics is to determine the shape of a function using only some known input-output pairs without having any knowledge of its form. In Machine Learning and Artificial Intelligence this problem is known as Supervised Learning.

To learn the relationship between input features (also known as predictor variables) and an output variable, the input-output pairs are divided into two datasets which will be used to train the model and evaluate its ability to accurately predict the output values for unseen data, respectively.

The training set, containing p input-output pairs can be represented by:

$$T = \{(x_i, \hat{y}_i)\}_{i=1}^{p}$$

The predictor variables are represented by the vector x_i, which contains all input values of the function. The output values are shown as $\hat{y}_i$, and usually contain some noise corruption, hence the correct output value (y_i) is unknown.

2.1.1 Regression Problems

Regression problems can be mainly classified into three subgroups [19]: *parametric*, *semi* and *nonparametric* regression.

In parametric regression, the general form of the relationship between the dependent variable and the independent variables is predefined before the learning starts. However, this relationship involves certain parameters whose precise values are unknown and need to be estimated using the available training data.

An example of parametric regression would be any Truncated Fourier series. This problem would be *parametric*, because the specific form of the relationship between the dependent variable (y) and the independent variable (x) is known, which is a simple equation of the form

$$f(x) = \sum_{i=1}^{k} a_i \cos(ixT) + b_i \sin(ixT)$$

where scalars $a_i, b_i \in \mathbb{R}^k$ and $T \in \mathbb{R}$ are the unknown parameters needed to be estimated. Once T is known or estimated, the function can be expressed as follows by using feature extraction:

$$f(x) = \sum_{i=1}^{k} a_i z_i + b_i v_i$$

where $z_i = \cos(ixT)$ and $v_i = \sin(ixT)$.

Nonparametric regression differs from parametric regression in that it does not rely on having much prior knowledge about the specific form of the true underlying function being estimated. To estimate the underlying distributions, the Kernel Density Estimation (KDE) is used [6]:

$$\hat{p}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

where K is a smooth function named as the *kernel function*, $h > 0$ is the *bandwidth*, which controls the amount of smoothing, and X is an identically distributed random sample from an unknown distribution with density function p.

Unlike parametric regression, where the number of parameters is typically big and may have clear physical interpretations, nonparametric models often use a low number of free parameters, and lack such direct relationships between their weights and parameters and the specific problem being solved.

Semiparametric problems combine elements of both parametric and nonparametric approaches, achieving a balance between model interpretability and capturing the complexity of real-life data. This allows for a better representation of the inherent variability and noise present in real-world scenarios. Neural network models are an example of semiparametric models. These include the RBFNNs, which are the topic of study in this project.

2.2 Basis Functions

A Linear Model can be described as follows:

$$f(x) = \sum_{i=1}^{n} w_i h_i(x)$$

The fixed functions $h_i(x)$ constitute the base of a Linear Model, and therefore will limit the range of functions that can be obtained by giving different values to the weights w_i. For this reason, they are usually called basis functions. If the basis functions are not fixed and can change during the learning stage of a model, then this model is nonlinear.

There are several types of basis functions, which are used in a wide range of fields and have different properties and applications. Some of the most used are the following types:

- **Polynomial Basis Functions**: These basis functions are often used for curve fitting and regression analysis [22]. The polynomial basis functions of degree 1, 2 and 3 (x^1, x^2 and x^3 respectively) are all examples of this type.
- **Fourier Basis Functions**: Based on the Fourier series, these basis functions are sinusoids with different frequencies and phases, and are especially used in signal processing and image analysis [18].
- **Wavelet Basis Functions**: With applications in denoising and time-frequency analysis, wavelet basis functions are localized in both time and frequency domains [9]. The Ricker Wavelet, which will be explained in more detail in Section 4.1.1, is an example of this type.
- **Radial Basis Functions**: Unlike the previous type, this type of functions model the output in terms of a fixed center and a given point [10]. They are used in RBFNNs, and will be explained in more depth in Section 3.1.
- **Spline Basis Functions**: Spline basis functions are piecewise-defined polynomials that are smoothly joined together at certain points called knots [14]. These functions are commonly used in interpolation and curve fitting.

2.3 Model Selection Criteria

The model selection criteria discussed in this project are all prediction error estimations, indicating how well the trained model is expected to perform on future data. The optimal model is the one with the lowest estimated prediction error. Cross-validation is one of the most commonly used strategy for measuring prediction error, and there exist different statistical learning methods which make use of it.

2.3.1 The Validation Set Approach

The validation set approach is a simple strategy for model selection. As illustrated in Figure 1, it consists in randomly dividing the input-output pairs into two main subsets: the *training* and *validation* subsets. The model is first fitted with the training subset, and is then used to make predictions for the validation subset inputs. The error rate obtained with the validation subset, often measured using the Mean Squared Error (MSE), serves as an estimate of the test error rate [14].

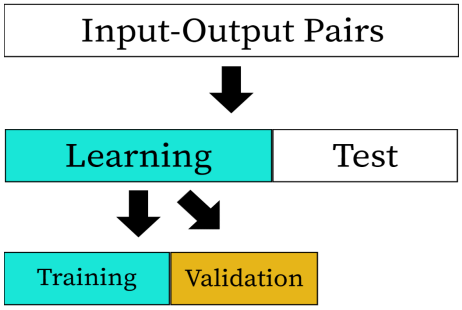

Figure 1: Schematic image of the validation set approach

2.3.2 Cross-validation

Cross-validation is a technique used in machine learning to assess the performance and generalization ability of a model. As seen in Figure 2, it involves partitioning the learning data into k multiple subsets named *folds*, with one subset used as a validation set while the rest are used for training. In order to predict the resulting MSE, the following formula is used:

$$\text{Validation MSE} = \frac{1}{k}\sum_{i=1}^{k}\text{MSE}_i$$

where MSE_i corresponds to the Mean Squared Error of the i-th validation subset.

Leave-one-out cross-validation is a special case of k-fold cross-validation where the validation subset consists of only one sample, and the training subset is created with the remaining $n-1$ samples [3]. This process can be repeated n times, calculating the MSE_i for the input-output pair (x_i, y_i).

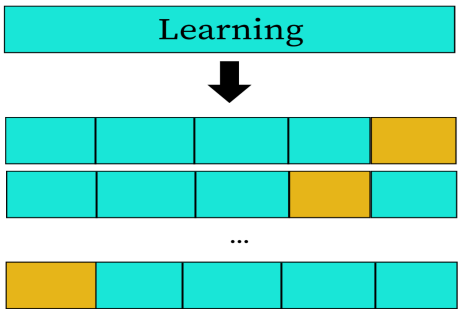

Figure 2: Schematic image of the 5 cross-validation, where cyan squares represent the training subsets, and orange squares represent the validation subsets

2.4 Regularization

One of the major problems in machine learning models is avoiding *overfitting*. This phenomenon occurs when a machine learning model becomes excessively complex and specific to the training set, to the point where it performs poorly on unseen data. In other words, the model "memorizes" the training data instead of learning general patterns and fails to generalize well to new examples.

To avoid *overfitting*, multiple methods have been developed, including cross-validation and regularization, amongst others. Unlike cross-validation, regularization consists of any method used to prevent machine learning models from minimize the error function as much as it could, in order to reduce its dependency on the specific training set. Its primary objective is to strike a balance between model complexity and generalization performance, thereby mitigating the risks of overfitting or underfitting.

For this study, two classes of regularization methods will be used: *weight shrinkage* and *subset selection* methods.

2.4.1 Weight Shrinkage

Weight shrinkage regularization works by introducing one or more penalty terms to the cost function, which *shrink* or limit the size of the coefficients towards zero. The main effect of this addition is a reduction in the model's variance, although depending on the type of shrinkage applied, it can lead to some coefficients being equal to zero. Therefore, some shrinkage methods can also be used for subset selection (Section 2.4.3).

LASSO Regularization

Least Absolute Shrinkage and Selection Operator (LASSO) is a regularization strategy particularly useful when dealing with high-dimensional datasets, where the number of predictors is large compared to the number of observations. This method results in the following expression:

$$C = \sum_{i=1}^{M} \left(\hat{y_i} - \sum_{j=0}^{p} w_j x_{ij} \right) + \lambda \sum_{j=1}^{p} |w_j|$$

where λ is the *L1* penalty term.

Ridge Regression

Ridge regression is a method of regularization used to deal with *multicollinearity*, which occurs when the predictor variables in a regression model are highly correlated with each other. Unlike the last explained algorithm, this method performs what is known as *L2 regularization*:

$$C = \sum_{i=1}^{M} \left(\hat{y_i} - \sum_{j=0}^{p} w_j x_{ij} \right) + \lambda \sum_{j=1}^{p} w_j^2$$

Thus, the optimization function is penalized when the coefficients w take large values.

As seen in the presented cost function, the ridge regression's effect over the cost function can be regulated by a *lambda* parameter $\lambda > 0$. The selected value for the lambda parameter is of great importance, as it will directly influence the model's predictions. As stated in [24], A small value of lambda means the data can be fit tightly without causing a large penalty, while a large value of lambda means a tight fit has to be sacrificed if it requires large weights.

2.4.2 Dimension Reduction

Dimension reduction methods involve expressing all the model's predictors (n) in a m-dimensional space, being $m < n$. By calculating m linear combinations of the variables, it is possible to fit a ML model having this m projections as its predictors. This strategy can help reducing the dimensionality of an oversized model, even removing redundant predictors.

2.4.3 Subset Selection

The subset selection methods are centered in identifying a subset of predictors which are closely related with the model's response. When this selection is done, a new model is fitted with a reduced set of variables, keeping the selected subset of related predictors. Reducing the quantity of used predictors can improve significantly the model's computational time.

Forward Stepwise Selection

Forward stepwise selection is an efficient subset selection algorithm, and it is widely used as a regularization method. It works by beginning with a model which contains no predictors, and then it starts adding the best predictors to the model, one at each iteration until every predictor has been added. The forward stepwise selection algorithm can be expressed as follows:

Algorithm 1 Forward Stepwise Selection Algorithm [14]

1: let M_0 be a model with no predictors
2: **for** $k = 0, ..., p-1$ **do**
3: Select all $p-k$ models which augment the predictors in M_k by one additional predictor.
4: Adjust the $p-k$ models with the training data and calculate each model's R^2.
5: Choose the model having the smallest NMSE and let it be M_{k+1}.
6: **end for**
7: Select the best model among $M_0, ..., M_p$ as the final model.

It is important to note that the selected model at each iteration can be chosen by using different techniques, including the cross-validated prediction error, the Akaike's Information Criterion (explained in detail in Section 5.3), the Bayesian Information Criterion or the adjusted R^2.

Backward Stepwise Selection

Backward stepwise selection is another efficient subset selection algorithm which, unlike forward stepwise selection, works by starting with a model composed by all the predictors, and then being reduced by substracting a predictor one at each iteration. This process is repeated until the model has no predictors. Its algorithm can be expressed as follows:

Algorithm 2 Backward Stepwise Selection Algorithm [14]

1: let M_p be a model with all the predictors
2: **for** $k = p, p-1, ..., 1$ **do**
3: Select all k models which contain all the predictors in M_k except one.
4: Adjust the k models with the training data and calculate each model's R^2.
5: Choose the model having the smallest NMSE and let it be M_{k-1}.
6: **end for**
7: Select the best model among $M_0, ..., M_p$ as the final model.

3 Radial Basis Function Neural Networks

This project focuses on the design, implementation and evaluation of Radial Basis Function Neural Networks (RBFNNs), which are a type of Artificial Neural Networks that use radial basis functions as the activation function. They are widely used to solve different types of problems, including function approximation, classification, time series predictions and pattern recognition. RBFN Networks are also known for their effectiveness and efficiency when approximating complex functions, as they normally use a relatively small number of hidden units.

3.1 Radial Basis Functions

A Radial Basis Function (RBF) is a real-valued function that exhibits a monotonically decreasing or increasing response as the distance from a central point changes. RBFs often are defined as a function of the Euclidean distance between the input point and the center point, although it is not the only used metric.

When being expressed as a linear combination, RBFs are typically used for function approximation, and can be used as a *kernel* for Support Vector Machines (SVM) [28] and RBFNNs. Different radial basis functions are used in ML [26], but some of the most common are the following functions:

- **Gaussian**: The gaussian radial function is the most used RBF in ML, and can be expressed by this function:

$$\phi_i(\boldsymbol{x}) = \exp\left[-\frac{||\boldsymbol{x} - \boldsymbol{c_i}||^2}{2\sigma_i^2}\right]$$

  where σ_i is a *shape parameter*, ϕ_i is the result of the i-th neuron, and $\boldsymbol{c_i}$ its centroid.

- **Multiquadratic**: The multiquadratic radial function is an alternative to the gaussian function which is also used in ML [24]. Its function can be expressed as follows:

$$\phi_i(\boldsymbol{x}) = \sqrt{1 + (\sigma_i||\boldsymbol{x} - \boldsymbol{c_i}||)^2}$$

  There exist some variations of the multiquadratic function, which include the inverse quadratic and the inverse multiquadratic functions.

- **Polyharmonic Splines**: Polyharmonic splines are a type of radial functions which have the characteristic of lacking a *shape parameter*, thereby avoiding the computational time required for tuning this parameter. Its general formula has the following form:

$$\phi_i(\boldsymbol{x}) = \begin{cases} ||\boldsymbol{x} - \boldsymbol{c_i}||^k, & k \notin 2\mathbb{N}, \\ ||\boldsymbol{x} - \boldsymbol{c_i}||^{2k} \ln ||\boldsymbol{x} - \boldsymbol{c_i}||, & k \in 2\mathbb{N}, \end{cases}$$

  These functions are also known as *surface splines* (or *thin-plate splines* in the case of $\phi_i(\boldsymbol{x}) = |\boldsymbol{x} - \boldsymbol{c_i}||^{2k} \ln ||\boldsymbol{x} - \boldsymbol{c_i}||$), and often exhibit good convergence rates in approximation.

3.2 Network Architecture

Following the traditional architecture first purposed by D. Broomhead and D. Lowe in 1988 [5] (seen in Figure 3), the general architecture of RBFN Networks consists of an input layer, a hidden layer and an output layer.

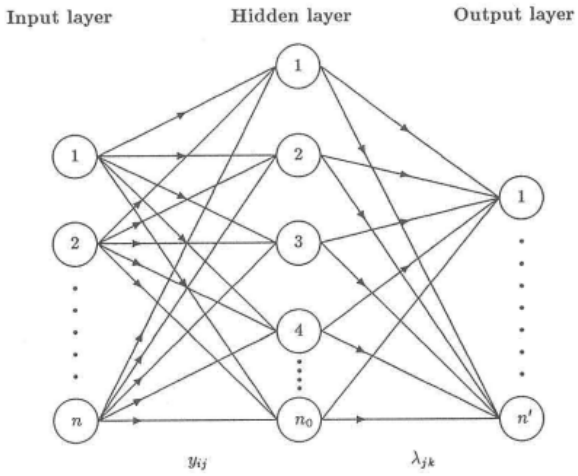

Figure 3: Radial Basis Function Neural Network Architecture

The input layer can be expressed as an input vector of dimension n, and is passed on each neuron of the hidden layer. The neurons of the hidden layer then compute a RBF (typically the Gaussian) as its activation function, using a neuron centroid which has been computed from previous training data. When all images of the RBFs have been calculated, all results are linearly combined with the network's weights to generate the RBFNN's output, forming the network's output layer.

3.3 Training Phase

Radial Basis Function Neural Networks generally feature multiple parameters. The most common are the weights or coefficients for the hidden layer (w), the neurons centroids c_i, the *shape parameter* of the RBF σ (if it has one or more) and the regularization parameter λ, if regularization is employed.

All these parameters are tuned during the training phase, when the hidden layer is generated along with each neuron's center and shape parameter. This tuning process can be divided into two main steps, the first one being the hidden layer generation by estimating the neurons centroids and calculating the shape parameters, followed by a second step in which the neuron weights are calculated.

3.3.1 RBF Centroid Selection

The centroid selection is an important step when building a RBFNN, as it will directly impact the model's performance. The most usual methods in centroid selection are clustering algorithms, which include the k-means algorithm among others.

The k-means algorithm is a simple iterative strategy used to divide given data into a predetermined number of clusters, denoted as k. This value is of great importance, as it determines the number of centroids the model will use. If there are too few centers, the network may struggle to generalize well, resulting in underfitting. On the contrary, if there are too many centers, the network may overfit by capturing irrelevant information from noisy data.

K-means minimizes the sum of distances from each sample data to the nearest centroid. This equation can be expressed as follows:

$$S(x) = \sum_{i=1}^{k} \sum_{\boldsymbol{x_j} \in \boldsymbol{S_i}} ||\boldsymbol{x_j} - c_i||^2$$

where $\boldsymbol{S}$ contains the different data subgroups ($\boldsymbol{x_j}$), and c_i is the centroid corresponding to the i-th of the k clusters.

This minimization is achieved by iterating between two steps until the first-step assignments no longer change, indicating convergence has been achieved. In the first step, the data is partitioned in different groups, assigning each sample to its closest centroid. There are different strategies to select the first set of centroids, which range from a random selection to more complex and better-performing algorithms, such as k-means++. In the second step, all clusters are recalculated, now being the mean of all samples assigned to each centroid. When no samples have changed their centroid, the iterative process has ended and the final centroid array is returned.

3.3.2 RBF Shape Parameter

Another important variable in RBFNNs is the shape (or *smoothing)* parameter. An excessive smoothing can lead to the loss of important details and nuances in the estimated true probability density. If it is too small, there is a risk of overfitting to the specific training set. It can be estimated separately for each neuron, or one estimation can be used for all neurons, which is proven to have universal approximation capability. There exist multiple alternative methods to calculate the shape parameter, such as computing and using the standard deviation of each cluster or using the average distances between each centroid and its m nearest centroids.

3.3.3 Output Layer Regularization

For regression problems in a radial basis function neural network (RBFNN), the output layer can be represented as a linear combination of the results from the activation functions. To properly minimize the output error, the weights can be calculated by using a linear pseudo-inverse solution. The equation to calculate the new weights, is [24]:

$$\hat{w} = (H^T H + \Lambda)^{-1} H^T \hat{y}$$

where $\hat{y}$ is a vector with the labels of the training set, Λ is a diagonal matrix with the regularization parameters (which can be null if no regularization is applied), and H is the *design matrix* which contains all the RBF results and can be expressed as [24]:

$$H = \begin{bmatrix} h_1(x_1) & h_2(x_1) & ... & h_n(x_1) \\ h_1(x_2) & h_2(x_2) & ... & h_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(x_p) & h_2(x_p) & ... & h_m(x_p) \end{bmatrix}$$

where each h_i corresponds to a hidden layer neuron output.

However, for classification problems, linear outputs are less suitable because they do not represent class probabilities. More appropriate alternatives derived from Generalized Linear Models (GLM) can be used for this problem types. The algorithm commonly used for training RBFNNs in classification problems is called Iteratively Reweighted Least Squares (IRLS) [20]. Thanks to the use of the Hessian matrix, which is negative semi-definite, it is ensured that only one maximum exists.

Alternatively, gradient descent can also be used as a training algorithm for RBFNNs in classification problems. The gradient descent algorithm finds a minimum by computing the gradient of the objective function at each iteration and adjusting the coefficients in order to get closer to a function's minimum. Although the two presented methods are equally valid to optimize the RBFNNs weights, it is often preferable to use Generalized Linear Models, as gradient descent generally require more computational time.

4 Biological Neuron Inspiration

In Artificial Intelligence, as well as in a large group of different fields, the emulation of biological systems has often proven to be a good source of inspiration. Among the numerous aspects of biological systems, neurons stand as fundamental units of computation, showing remarkable adaptability, fault tolerance, and parallel processing capabilities. Making use of the intrinsic power of biological neurons has driven numerous advancements in machine learning and neural network research.

In this project, the biological study is centered around an observed property named as *lateral inhibition* (Section 4.1), which is produced in the cortical networks of the human brain. For the experimental study, this property is employed to build alternative RBFNNs to explore its applications and limitations.

4.1 Lateral Inhibition

Lateral inhibition is a neuron behaviour observed in cortical networks located in the cerebral cortex of the brain, which constitutes the outermost layer of the brain. The cerebral cortex is a highly folded region consisting of neural tissue that covers the cerebral hemispheres. It is responsible for many cognitive functions, such as sensory perception, motor control, language processing, memory, attention, and decision-making.

Neurons situated in the cortical networks exhibit an excitation state when a stimuli is produced in their region (e.g. light falling in a particular region of the visual field). This state is translated as an electrical signal generated in response to the incoming stimuli. Along with this action, the excited networks send an additional inhibitory signal to their neighbours, thereby producing a *lateral inhibition* which augments the contrast between the stimulated and the non-stimulated neurons. As an example of a biological usage, lateral inhibition allows the perception of fine details and the extraction of important features from the sensory input in the visual system [17].

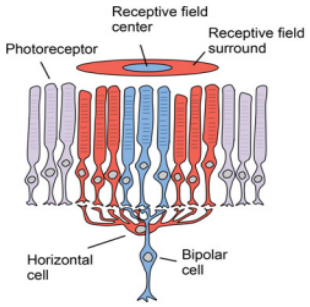

Figure 4: Photoreceptor interactions to produce lateral inhibition (from [17])

4.1.1 The Ricker Wavelet

It has been observed that lateral inhibition decays with the increasing distance between neurons, resulting in a signal relationship which resembles a "Mexican hat" [33]. This *mexican-hat* shape (Figure 5) can be modeled by a wavelet known as the *Ricker* Wavelet, which contains a shape parameter to control its peak frequency [25] and can be expressed as follows:

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{1/4}} \left(1 - \left(\frac{t}{\sigma}\right)^2\right) e^{-\frac{t^2}{2\sigma^2}}$$

By estimating the shape parameter σ and expressing the wavelet time t as the Euclidean distance from a given sample to a neuron's center $t = ||\boldsymbol{x} - \boldsymbol{c_i}||$, the Ricker wavelet can be used as a radial basis function on a RBFN Network.

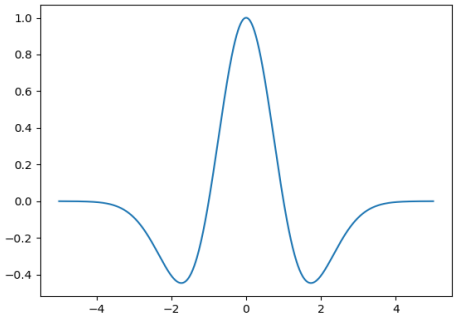

Figure 5: Ricker Wavelet function obtained with a shape parameter $\sigma = 1$

4.2 Biological Context

In the past years, different studies have been conducted in order to find new biologically plausible activation functions and develop an implementation for real data science problems.

Gidon et al. [11] studied the dendrites in human layers two and three cortical networks. Dendrites are a fundamental part of neurons that shape a neuron's electrical input and output. In this study it was discovered a class of calcium-mediated dendritic action potentials (dCaAPs) with different waveforms and effects on a neuron's output. These new waveforms observed in neurons can be used as radial basis functions for RBFN Networks.

Gardave S. Bhumbra [2] introduced a new alternative to the generally used Rectified Linear Unit (ReLU) and Exponential Linear Unit (ELU) activation functions, named Bionodal Root Unit (BRU). This alternative exhibit input-output non-linearities substantially more biologically plausible since their behavior is based on known properties of neuronal cells. This study shows the possibility of well-performing biologically plausible activation functions.

Based on the first recent study presented, Muthiah-Nakarajan et al. [21] presented and studied four different oscillating activation functions inspired by behaviors observed in biological neurons from layers two and three of the human cortex, which allowed individual neurons to learn the XOR operation without manual engineering. The goal of this project was to explore the possibilities of the presented oscillating activation functions to solve classification problems with fewer neurons and training time.

Kukjin Kang et al. [15] realized a study centered on the effects of the Mexican hat waveforms found in the neuronal cortical circuitry which originate in local groups of cells that form network models in frontal cortex, parietal cortex and other neuronal circuits. This waveform (explained in Section 4.1.1) will be used as an activation function in this study.

5 Training of the RBF Neural Network

To conduct the experimentation part of this study, a Radial Basis Function Neural Network has been designed for the Python programming language. Despite the existence of several public modules, it has been decided to develop a new implementation. The decision to build a self-implementation is motivated by the limited flexibility offered by the existing modules in terms of designing the architecture of a RBFNN. For instance, these modules may not allow the selection of a centroid calculation strategy or lack regularization options, which are essential requirements for this study.

The RBFNN model designed in this project maintains the RBFNN's traditional architecture (Section 3.2), which consists of an input layer, a hidden layer and an output layer. The hidden layer is generated by using the k-means clustering algorithm, while the output layer is obtained by fitting a Generalized Linear Model. Lastly, the GLM is regularized by using both backward selection strategy and Ridge regression.

5.1 K-Means Clustering for Centroid Calculation

The developed model's hidden layer is built by using the k-means algorithm along with the k-means++ initial centroid strategy, thus obtaining the neurons centers, which play an essential role in the design matrix calculation. As stated in Section 3.3.1, k-means comes with a tunable parameter (k) which is of great importance, as it will determine the number of neurons used by the model.

To avoid a computationally complex estimation for the k parameter, it has been chosen to select an oversized value, which will generate a larger quantity of neurons in relation to the number of samples. The exceeding amount of neurons will produce *overfitting*, which will be treated by the execution of the regularization techniques described in Section 5.3.

5.2 Generalized Linear Model as the Output Layer

After the neuron centers have been estimated, the design matrix of the model can be calculated by applying the corresponding radial basis function. In this project, two distinct radial basis functions are covered: the Gaussian and Ricker. Both of these functions have a shape parameter, denoted as σ, which needs to be estimated. To determine an appropriate value for σ, the *sigest* algorithm has been chosen due to its computational simplicity and fast performance time, among other options.

The *sigest* algorithm is used to estimate the shape parameter of a Gaussian RBF employed in Support Vector Machines. It is based on calculating the 0.1 and 0.9 quantiles of the squared Euclidean distances between data points. As stated by the author of the algorithm's R implementation [16], any value between these two boundaries will produce good results. However, a public *sigest* implementation for Python does not currently exist. Consequently, a self implementation of the *sigest* algorithm has been developed based on the R version, which works by performing the following steps:

Algorithm 3 Sigest Algorithm

1: Prepare the input data X by reshaping and scaling the array, if needed.
2: Randomly select two subsets of samples from X.
3: Extract the samples (x) from each subset and calculate their pairwise distances.
4: Compute the squared Euclidean distances between the samples and sum them along to obtain a 1-dimensional array M.
5: Calculate the 0.1 and 0.9 quantiles of M to estimate the range of σ.

Once all neuron RBFs have been calculated and the design matrix has been constructed, the design matrix can be fed into a Generalized Linear Model, which will represent the model's output layer. However, this is not the definitive output layer, as the RBFNN may be still overfitted and needs the application of regularization.

5.3 Subset Selection and Regularization

The last process needed to build a RBFNN model with the designed decisions consists in applying regularization to the output layer. For the designed RBFNN model, two regularization strategies are applied in order to suppress its overfitting.

The first regularization algorithm applied to the model is a subset selection method known as *backward stepwise selection* (Explained in Section 2.4.3). This algorithm reduces the model's neurons iteratively, selecting the reduced model which yield the best results by the chosen quality metric. This process is repeated until a better-performing reduced model is no longer found.

Akaike's Information Criterion (AIC) has been the selected metric to estimate the model's quality during the backward stepwise selection process.

Firstly purposed by Akaike [1], AIC estimates the relative amount of information lost by the radial basis functions, and can be represented with the following log-likelihood formula:

$$AIC = 2k - 2 \cdot \log(L)$$

where k is the number of free parameters and L is the maximum likelihood of the model.

Ridge regression is also applied to the RBFNN as a second step of regularization. As detailed in Section 2.4.1, this *L2* regularization method adds a penalty term to the cost function. The addition of the mentioned term influences the weights estimation, specially affecting the large-valued coefficients.

To correctly select an adequate regularization parameter for each problem, a wide group of candidates planned by the researcher will be evaluated. The selected *L2* candidates (λ) are:

$$\lambda = \left[10^{-3}, 10^{-2,5}, 10^{-2}, 10^{-1,5}, 10^{-1}, 10^{-0,5}, 10^{0}, 10^{0,5}, 10^{1}\right]$$

Each of these candidates repeat 10 times the execution of 5 different 2-fold cross-validation evaluations (10x5x2). The 5x2 cross-validation is performed in order to obtain the model's Normalized Mean Square Error (NMSE), as well as the t statistic [7]. These two quality metrics are explained in Section 6.2. All this process is executed 10 times for each parameter to compare different k-means groupings and select the best. If this last selection wasn't done, the RBFNN's error would greatly vary depending on the k-means grouping.

By making use of these two steps of regularization the model is ensured to have a robust way to deal with overfitting and reduce the model's complexity.

The process of training the designed RBFNN models with an estimated *L2* regularization parameter can be summarized in in Algorithm 4.

Algorithm 4 RBF Neural Network Training

1: **Build the Hidden Layer of the model**
2: Estimate an overfitted k from the number of samples.
3: Perform the k-means algorithm with the estimated value of k.
4: **Estimate the smoothing parameter**
5: Calculate the smoothing parameter using the *sigest* algorithm.
6: **Build the Output Layer of the model**
7: Build the design matrix H by calculating the model's RBF output for each hidden neuron.
8: Fit a Generalized Linear Model with the design matrix H.
9: Perform AIC Backward Stepwise Selection to reduce the number of neurons of the model and adjust the GLM.
10: Apply *L2* Regularization with Ridge Regression to correct the remaining overfitting.

6 Experimental Work

In order to correctly study the performance of a biologically more plausible RBFNN design, and compare its effectiveness with the commonly used gaussian RBFNNs, different experiments have been conducted. The main goal of the performed experiments is to gather descriptive metrics which help to identify patterns of functioning for each RBFNN. These patterns can prove the correct functioning of the developed model, as well as identify for which type of problems it scores better predictions.

6.1 Employed Training Data

The performed studies have been divided into two main experiments. These experiments differ from each other by the nature of the used data. The first experiment has been performed using *synthetic* data, generated by the researcher, while the second experiment has been performed with realistic data.

6.1.1 Synthetic Experiment

The synthetic experiment has been performed with the intention of evaluating each RBFNN's correctness, and to observe each network's behaviour when the difficulty of the problem changes through different factors.

The synthetic experiment's data has been generated by obtaining the image for multiple points (x) in the $\mathbb{R}$ space using the following function:

$$f(x) = \frac{sin(bx)}{bx} + \mathcal{N}(0, \sigma^2)$$

which is commonly used in the ML field to test the efficiency of a model. This function contains two parameters (b and σ^2) which regulate the shape of the function. The b parameter controls the "frequency" of the *sinc* function. Consequently, a larger value of b results in a more difficult function for the RBFNN models.

This effect can be seen in Figure 6:

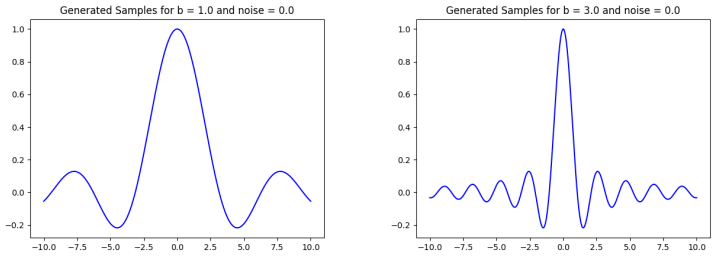

Figure 6: Comparison of the *sinc* function using different values of b. $b = 1$ on the left, and $b = 3$ on the right

The second parameter σ^2 also affects the resulting function's difficulty. It is used to add noise following a normal (gaussian) distribution. Like the b parameter, a larger value for σ^2 results in a more difficult problem. Its effect can be seen in Figure 7:

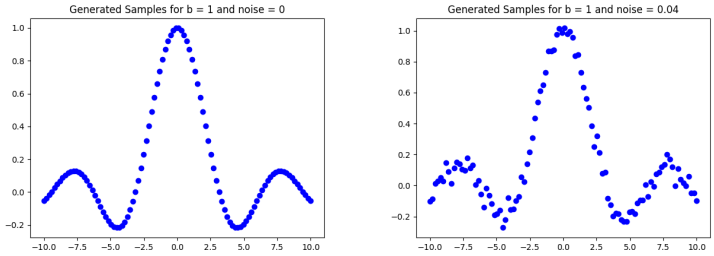

Figure 7: Comparison of the *sinc* function using different values of σ^2. $\sigma^2 = 0$ on the left, and $\sigma^2 = 0.4$ on the right

A third parameter which also affects this experiment is the number of examples used to train the model. It is of great importance to study how good a model can perform with a low number of training data. Unlike the two previously mentioned parameters, a greater number of training examples will generally improve a model's predictions.

6.1.2 Realistic Experiment

When studying a RBFNN it is important to evaluate its performance with realistic data distributions, which may contain noise and have a low number of samples. For this study, a one-dimensional dataset from *Statistics Online Computational Resource* (SOCR) has been used to test the models performance with realistic data. It consists of 25.000 records of the heights (in inches) and the weights (in pounds) of 18 years old children [30], and it is based on a Growth Survey taken in Hong Kong's Maternal and Child Health Centres in 1993. This dataset is a good option to test the developed models, as it is a difficult real scenario in which there may be people with a great value of height but a low weight, and vice versa.

6.2 Metrics Selection and Measurement

There are multiple parameters related with the performance of the developed RBFNNs which can be studied. For each parameter, it is important to select a complete range of values in order to correctly test the designed models in different conditions. If the value range of a parameter is not well defined, the RBFNNs may only be tested in one specific condition.

6.2.1 Problem metrics

For the synthetic experiment, it has been chosen to employ the following value distributions for the parameters which influence the problem's difficulty:

Parameter	**Values**
b	[1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0]
σ^2	[0.00, 0.01, 0.02, 0.03, 0.04, 0.06, 0.07, 0.08, 0.09, 0.10]
Training Samples	[100, 500, 1000]

Table 1: Range of values for the difficulty parameters in the synthetic experiment

The selected values for b and σ^2 cover a wide range of scenarios. Including additional values would result in scenarios that closely resemble existing ones. Increasing the b parameter beyond 5.0 would only marginally increase the error rate, while higher values of σ^2 would also lead to significant errors, considering that $\sigma^2 = 0.1$ already produces substantial error.

In regard to the number of training samples, the three general cases are already being tested. These are:

- Having a low or insufficient number of training samples, which often results in an unexpected function shape because of the large variance.
- Having an enough number of training samples to predict correctly the expected function shape
- Having more training samples than needed, which often results in overfitting, when this case is not controlled.

6.2.2 Quality Metrics

Regarding the quality of a RBFNN model, two metrics have been obtained: the Normalized Mean Square Error (NMSE), and the t statistic. The NMSE is obtained from the Mean Square Error (MSE), which represents the mean square difference (error) between the real and the predicted data, and can be expressed with the following function:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2$$

Where $\hat{Y}_i$ are the observed values, and Y_i are the ones predicted by the model. Thereby, the NMSE can be computed by normalizing the MSE with the variance of the observed values ($NMSE = MSE \div VAR(Y)$). When a model has a good prediction accuracy, the NMSE takes a low value, where 0 is the best possible error. On the contrary, a large value for the NMSE means a poor prediction accuracy, being 1 the worst possible error.

The second quality metric obtained is the t statistic. It was devised by *Thomas G. Dietteritch* in 1998 [7], and it is used to compare the quality of different models. The t statistic is obtained by performing 5 replicated 2-fold cross-validations, and can be calculated as follows:

$$\bar{t} = \frac{p_1^{(1)}}{\sqrt{\frac{1}{5}\sum_{i=1}^{5} s_i^2}}$$

where $p_1^{(1)}$ is obtained by subtracting the testing data error to the training data error of the first group of the first cross-validation replication

$$p_1^{(1)} = p_A^{(1)} - p_B^{(1)}$$

and s_i^2 is the estimated variance, obtained by the following formula:

$$s_i^2 = (p_i^{(1)} - \bar{p_i})^2 + (p_i^{(2)} - \bar{p_i})^2$$

where $\hat{p_i} = (p_i^{(1)} + p_i^{(2)})/2$.

6.2.3 Model Metrics

Different metrics of the RBFNN models have been studied and discussed in this project to compare each model's performance with problems of different difficulty and nature. The selected metrics to study are the following:

- **Number of neurons of the model**: It is an important metric, as it defines the RBFNN's complexity and affects the computational time needed to create the network.
- **Number of reduced neurons**: With this metric it is possible to study the backward selection regularization process and determine if it works as intended.
- **Estimated smoothing parameter**: The smoothing parameter greatly affects a model's predictions. From the range of values it takes, as how it changes depending on the RBF used, it is an interesting factor to study.
- **L2 Regularization parameter**: The L2 regularization parameter selected for each problem is an important metric to study in order to see if the chosen range of values is wide enough, or if the models tend to always select the same candidate.
- **Computation time**: Another interesting metric to measure is the computation time, as it also provides information about the complexity of a model.

6.3 Result Analysis

In this section, the results of the presented experiments are shown and discussed. In the next section, the results of this study are summarized, and a conclusion is presented.

6.3.1 Synthetic Experiment

The first results which have been obtained correspond with the model's metrics. The following tables contain each mean model metrics obtained with the different values of b and σ^2 for each number of samples:

# Samples	**Neurons**	**Reduced Neurons**	σ^2	**L2**	**NMSE**	**Time (s)**
100	8,41	3,59	0,91	10^0	0,3105	22
500	18,50	4,50	0,88	10^0	0,1077	94
1000	20,79	6,21	0,84	10^0	0.1059	124

Table 2: Mean Gaussian RBF model metrics obtained with the different values of b and σ^2 for each number of samples

# Samples	**Neurons**	**Reduced Neurons**	σ^2	**L2**	**NMSE**	**Time (s)**
100	8,44	3,56	0,91	10^0	0,3076	21
500	18,71	4,29	0,89	10^0	0,1079	94
1000	20,58	6,42	0,85	10^0	0.1052	190

Table 3: Mean Ricker RBF model metrics obtained with the different values of b and σ^2 for each number of samples

From this tables, it can be observed that the obtained metrics for the two models are closely similar. The similarity in the number of neurons is normal given that in the designed RBFNNs, the initial neurons number is dependant from the number of training samples. As we train the two models with the same samples, the difference in the number of neurons of the two trained models has been produced by the regularization. Another noticeable metric is the *L2* regularization parameter, which has resulted in a value of 10^0 for all cases. Given this fact, a new selection of candidates of smaller magnitude near 10^0 could be explored in future studies to introduce more *L2* variation.

In order to prove the influence of the b and σ^2 parameters in the synthetic problem's difficulty, the following graphic (Figure 8) has been generated with the metrics obtained by training each model with 500 samples:

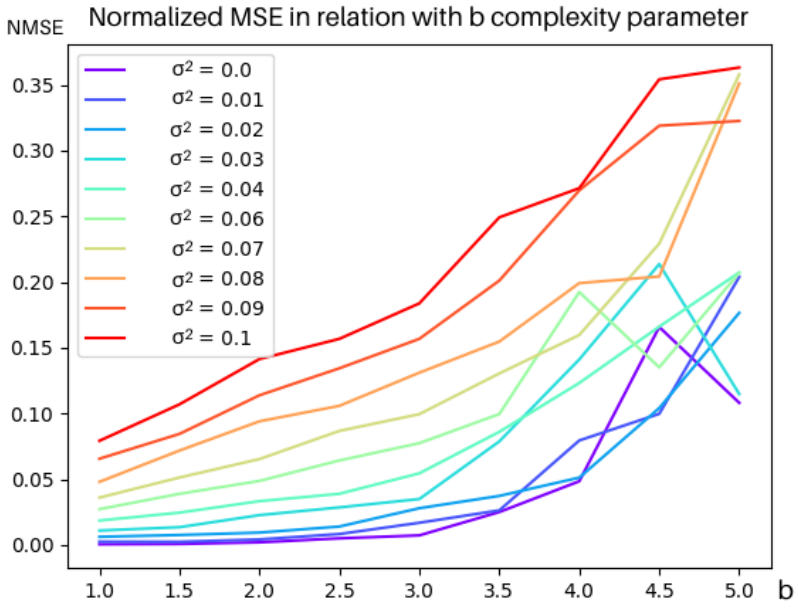

Figure 8: NMSE values obtained with the Ricker model for each b and σ^2 combination

By observing the provided table, it is clearly seen how NMSE takes larger values as the two mentioned parameters increase. The increase on the NMSE given by the b parameter follows an exponential tendency, which is displaced higher or lower depending on the noise in the samples.

Furthermore, a model comparison has been conducted with the intention of finding different patterns of functioning and performance quality depending on the problem's difficulty. To carry out this comparison, the b parameter has been fixed to its mean value ($b = 3$), and the t statistic (computed from the two models) has been calculated and stored for each noise and number of samples. To represent this matrix, a color has been given to each calculated test. If the result is **positive**, the Gaussian RBF model performs better and is represented by a green color. If the result is **negative**, the Ricker RBF model has a better performance and is represented by a red color. If the test is not meaningful, it is represented by a white color. The obtained representation can be seen in Figure 9:

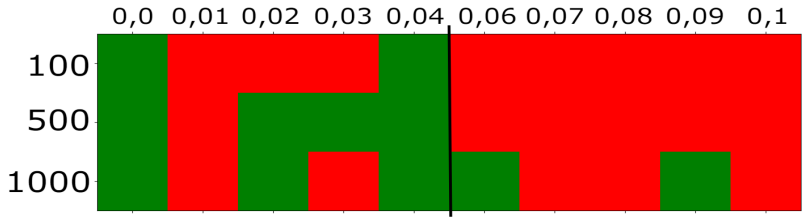

Figure 9: 5x2 CV test for each σ^2 and training sample size

Observing the patterns of this test matrix, it can be seen that the Ricker RBFN generally performs better than the Gaussian RBF when larger values of noise (from $\sigma = 0,05$) are present in the training data. However, when the problem has a small quantity of noise, no model seems to be clearly performing better than the other. These results support the idea of using any model indistinctly for low values of noise in the training data.

6.3.2 Realistic Experiment

For the realistic experiment, each RBFNN has been fitted with 1.000 training samples. Next, a prediction test has been conducted with 10.000 additional values. The model metrics obtained are summarized in the following table:

RBFNN Type	**Neurons**	**Reduced Neurons**	**L2**	**NMSE**	**Time (s)**
Gaussian	15	8	10^0	0,7678	108
Ricker	21	2	10^0	0,7661	115

Table 4: Model and quality metrics for each RBFNN with the realistic dataset

Even though the Ricker RBF model seems to perform better with the realistic data, the obtained NMSE values suggest the developed models are not performing well with the provided dataset. This idea can be confirmed by displaying each model's predictions compared with the real values. This can be seen in Figure 10:

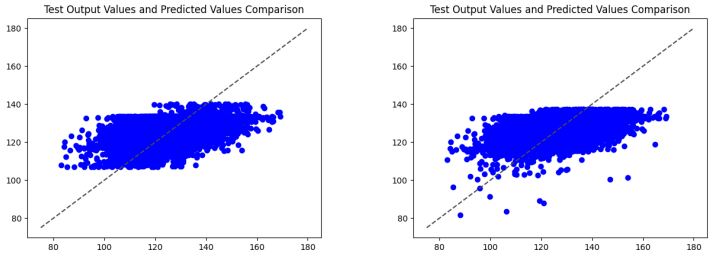

Figure 10: Real values (x) compared with the test predictions (y) of each model. Gaussian on the left, and Ricker on the right

These graphics show the difference between the predicted and real values. As stated before, the two models don't seem to be performing adequately. To better understand the cause of this problem, an additional graphic has been generated, showing the data distribution of the realistic dataset:

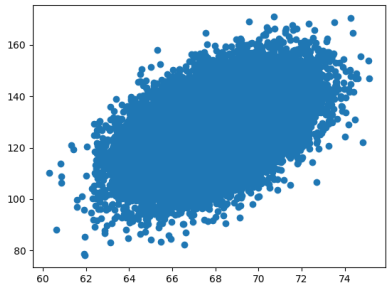

Figure 11: Data distribution of the dataset, where each height (x) is shown with its corresponding weight (y)

By observing the dataset distribution, it can be seen that it has a large inherent noise, which is correct by the nature of the data: a very tall person can weight very little, as a person of short stature can weight a lot. Therefore, it is possible that the large error obtained by the two RBFNN models is not due to a bad parameter estimation, but for the amount of noise present in the dataset.

By grouping all the weights obtained by the same height, it is possible to estimate the dataset's noise by obtaining the mean variance of the formed groups:

$$Dataset\ Noise = \frac{1}{n}\sum_{i=1}^{n} VAR(g_i)$$

Where g_i is the group containing all weights for a same height h_i in the dataset.

By applying this formula, a value of **104,58** is obtained with the realistic dataset. By multiplying the variance of the real values of the testing set with the obtained NMSE values, the MSE can be obtained and compared with the dataset's inherent noise:

$$MSE_{Gaussian} = NMSE \cdot VAR(Y) = 0,7678 \cdot 136,9397 = 105,14$$

$$MSE_{Ricker} = NMSE \cdot VAR(Y) = 0,7661 \cdot 136,9397 = 104,90$$

By performing these calculations, it can be seen that the prediction error of the RBFNN models is almost identic to the dataset's inherent noise. Therefore, the developed models achieve a performance near to the best possible with the evaluated dataset.

7 Conclusions

This project has been dedicated to the study and implementation of Radial Basis Function Neural Networks, presenting and discussing the Ricker Wavelet, which is a biologically plausible activation function. The two presented designs utilize a complete set of ML and statistic methods to improve their performance, which include *L2* regularization to deal with overfitting (Explained in Section 2.4.1) and the *sigest* algorithm (Detailed in Section 3) to efficiently find the shape parameter. Furthermore, additional techniques have been implemented in the models to improve their efficiency. For example, estimating an oversized k parameter for the k-means algorithm and then reducing the neuron number with AIC backward selection (explained in Section 2.4.3), thus avoiding k cross-validation.

Given the limited existing RBFNN libraries in the Python programming language, the designed models have been developed to be internally configurable, allowing the user to configure the model's activation function, the regularization process and the neuron estimation criteria, among other options. The developed implementation has not been explained in this project due to the consideration of its low relevance with the study. However, it will be published in the researcher's GitHub account.

When observing the results of the planned experiments, the correct functioning of the designed RBFNN models has been proven through the obtained estimates. Moreover, the validity of the synthetic experiment has also been proven by the gathered NMSE, which increased as expected in relation to the b and σ^2 difficulty parameters. A correct performance with realistic data has also been observed by both models.

Comparing the two models has shown interesting results, including an observed superior performance by the Ricker model when a large noise value was present in the data, while a similar performance has been observed with low noise values. These results may happen due to the influence of lateral inhibition, which could be explored further in future studies.

The developed RBFNN models have also shown a good performance quality with realistic data containing a large noise value. The obtained error for both models have been almost equal to the dataset's variance, which suggests that they can adapt to problems of different difficulty.

7.1 Future Work

While this project has mainly focused in the quality metrics obtained by each model as well as some internal metrics like the number of neurons or the estimated shape parameter, the internal configurations could be studied in more depth. The neuron distribution along the fitted data could be studied, as lateral inhibition probably displaces the Ricker model's neurons further from each other to avoid large signal inhibitions. For this study, various heatmaps and graphs could be generated to visually represent the neuron distributions of different models.

In this study, the smoothing parameter is estimated with the *sigest* algorithm. However, this may not be the most correct way to calculate it. A study could be performed to find better methods to estimate the smoothing parameter for the Ricker Wavelet. Furthermore, calculating of a different parameter for each neuron could also be studied in depth to compare its performance with models which only perform one estimation.

Even though more biologically plausible functions were originally planned for this project, which included the Ormsby and the Klauder wavelets, they were discarded due to having multiple smoothing parameters. Having a big number of smoothing parameters limits a model's efficiency, as it is very difficult to estimate or search the optimal values for each one. A study could be conducted with the objective of exploring different radial basis functions which share biological properties like lateral inhibition, or to find efficient estimation methods for the mentioned wavelets.

The generated Dietterich matrix could also be a topic of study. There exist different data mining algorithms like the Apriori algorithm which could help identify different *association rules*. These association rules express relationships between items in frequent itemsets such as knowing which model would perform better given different values for the difficulty parameters.

In order to greatly improve the efficiency of the developed implementations, different parallelism techniques could be applied with the intention of speeding up the creation of the RBFNN models. This extension could be focused in the k-means, and in the creation of the output of the second layer.

8 Project Management

8.1 Methodology

In order to have effective management for the tasks of this project, the Kanban methodology has been used. Kanban is a project management methodology centered in having a good visualization of the different pending tasks, based on the existing agile methodology. This working framework allows for good real-time organization, task prioritization and identification of possible bottlenecks. Each task is grouped in a category depending on its stage of completion, and it can change dynamically during its development. In this project, tasks have been grouped in four main categories:

- **To do**: The tasks in this category have been defined and are ready to develop, but no work has been dedicated to the tasks yet.
- **In progress**: The tasks in this category have been started and are being developed, but there is still work to be done.
- **Reviewing**: The tasks in this category have been completed, but further testing and review is still necessary to validate the correct functionality of the task.
- **Completed**: The tasks in this category have been developed and tested, validating its correct functionality.

A web application named ClickUp has been used to properly manage the defined tasks and correctly control its categories. This website makes use of the Kanban methodology, and has a clean user interface with a good visibility of all tasks grouped by categories. ClickUp also allows users to set and track their objectives using the OKR (Objectives and Key Results) framework, write descriptions for the different tasks and set a priority for each one of them. It can be accessed from any browser or downloaded as a mobile or desktop app.

8.2 Validation

During the development of the project, different validation processes have been executed to ensure the correctness of the designed Radial Basis Function Networks.

The project's development process has been supervised by the tutor. Periodically meetings have been scheduled, where the objectives and tasks of the project were be discussed. During the experimentation phase of this study, more meetings were scheduled in order to closely supervise its development.

During the training process of the different RBFNs, the optimal parameters for the Artificial Neural Networks have been obtained by the application of model validation techniques, such as cross validation. This processes ensure a good parameter selection for each model.

The training and evaluation of the developed RBFNs have been done multiple times, for each data science problem selected. This avoids false positives that could pass their testing phase by working correctly in one specific case.

8.3 Project Planning

This project has an approximate duration of 485 hours, which are spread across a span of 136 days. Its development started the 10th of February and ended June 27th. To ensure that the project had been completed on schedule, an average of 3.5 hours have been dedicated each day. However, some deviations have happened throughout the project due to exams and unforeseen circumstances.

8.3.1 Task Definition

The following is a comprehensive list of tasks which have been carried out during the project. Each task includes a description of the work to be performed, its estimated duration, and any dependencies it may have with other tasks. All tasks and dependencies are summarized in Table X and visually displayed in Figure X.

In order to have a better organization, all tasks are divided into four main groups: project management, research, implementation and testing, analysis and delivery.

Project Management

Project management is an important group of tasks which define the project's scope, planning, sustainability and organization. This group consists of five tasks:

- **Context and Scope**: Definition of the project's scope, contextualization and justification. Its general objective is presented, along with its relevance and methodology.
- **Project Planning**: Temporal planning for the complete development of this study, including a description of the different project phases and the resources and requirements associated with each of them.
- **Budget and Sustainability**: Total cost and impact produced during this project's development. An analysis of the sustainability of the project is included, along with a budget plan.
- **Integration in a Final Document**: Grouping of the different documents submitted previously with general improvements and adjustments based on the director and tutor's suggestions.
- **Weekly Meetings**: Periodic meetings which have been scheduled with the director of this project to supervise and discuss the progress of this study.

Research

In order to correctly develop an implementation for the different RBFN Network models it is fundamental to understand how RBFNNs work and be aware of all their important concepts. It is also necessary to have great knowledge about different data science problems in order to test the RBFNNs with a problem set that encompasses a wide range of areas and applications. This part is divided into three tasks:

- **RBFN Networks**: Research about the fundamentals of RBFN Networks and study the existing implementations and applications.
- **Activation Function Selection**: Research and select different biologically plausible activation functions to develop a RBFN Network design for each function.
- **Problem Selection**: Research and select a diverse set of problems in order to develop a performance comparison of the different RBFN Networks.

Implementation and Testing

Once different biologically plausible activation functions have been selected and there is sufficient knowledge about RBFN Networks, the implementation of different Artificial Neural Networks can be carried out. This group includes three tasks:

- **RBFN Networks Implementation**: For each biologically plausible activation function, develop a Radial Basis Function Neural Network. These designs may be based on already existing models, but several modifications are still required.
- **RBFN Network Training for Different Problems**: Once all models are implemented, it is still necessary to train the models for the selected problems.
- **Test and Measurement**: Test and measure the efficiency and prediction accuracy of the RBFNs for different problems.

Analysis

After implementing the different models, training them with a set of problems and obtaining the corresponding results, an analysis of the obtained data can be made in order to compare the RFBNN predictions. This group consists of two tasks:

- **Model Comparison**: Compare the individual results obtained for each activation function in order to determine the best performing models.
- **Conclusion Extraction**: Extract conclusions from the comparison and metrics obtained during the study to determine which problem each current activation function yields the best results for.

Delivery

When all the previous tasks are completed, it is still needed to complete the project's documentation. It is also important to prepare the study's oral defense. This final group includes two tasks:

- **Project Documentation**: This task will be done while the other tasks are being completed, with a final dedicated time to revise all contents.
- **Oral Defense Preparation**: It is of great importance having a well organized summary of all contents which can be presented to the corresponding tribunal members.

8.3.2 Resources

To be able to complete the specified tasks, it is necessary to count with certain resources. These resources have been classified in the following groups:

Human Resources

The main human resource of this study is the researcher, as it is responsible for the project's development. Both the director Luis Antonio Belanche Muñoz and the GEP tutor Joan Sardà Ferrer are also human resources involved in this study, as they are in charge of guiding and helping the researcher throughout the project.

Hardware Resources

A necessary resource for the realization of this study is a computer. Two different computers have been used during this project:

- Desktop computer with 32GB of RAM and Intel® Core™ i7-6700K CPU 4.0GHz.
- Laptop computer with 16GB of RAM and Intel® Core™ i7-9750H CPU 2.6GHz.

Software Resources

Google Meet has been used to effectuate the meetings with the research director. In order to control the different versions of the implemented code, Github has been used. ClickUp web-app has been used to properly organize the different tasks. Visual Studio Code has been used for the development of the different RBFNs. To write the documentation of this study, Overleaf has been used, which is a LaTeX editor. Finally, Ganttproject has been used to create the Gantt charts presented in this work.

Material Resources

Some material resources such as papers or books were consulted in order to accompany the online research.

8.3.3 Risk Management and Alternative Plans

During the development of this study, a risk analysis was conducted in order to dispose of an alternative planning strategy. Next, a solution for each planned problem is presented, suggesting an alternative task arrangement along with its level of risk.

- **Library Issues**: This problem has a medium risk in this project. An incompatibility or a bug in a used library could be resolved by searching for a more stable version, or finding a different library which offers the same functionality. This would require the PC, VSCode and the researcher as the main resources, and would mean having to reimplement the affected code with the new library. Although this problem can be easily solved in case of an alternative library, the additional time required to solve this issue is calculated considering a reimplementation of the library, which would take around 30 additional dedication hours.

- **Delays in the Delivery**: This problem would have a high risk in this project. A possible solution for this problem would be to develop an alternative planning in the future, when better time estimations can be made from experience. Ganttproject and the researcher would be the additional resources required. The development of an alternative planning can require an additional time between one or two hours, approximately. An increase in the dedicated time per day could also solve this problem in case the proposed solution is not enough to correct the delay in the delivery.

- **Limited Existing Research**: This problem has a medium risk in this project. Due to a low amount of existing research about this study's topic, a good solution could be to keep in contact with the director, who can provide detailed explanations or suggest physical or online documentation. The resources required for this solution would be the researcher and the project director. The dedicated time to solve this problem could be around 15 hours, considering five one-hour extra meetings with the director and ten additional hours to study the provided additional content and information.

- **Computational Complexity**: This problem has a medium risk in this project. Google Collab or Kaggle Kernel could be used to solve this problem, as they offer a high performance execution of trained models with various GPU options. This problem would require one of the mentioned software resources, and the dedicated implementation time would increase close to five hours in order to port the code to the new chosen platform.

ID	Name	Time(h)	Dependencies	Resources
PM	**Project Management**	**85**		
PM1	Context and Scope	25		PC, Researcher
PM2	Project Planning	10	PM1	PC, Researcher, Ganttproject
PM3	Budget and Sustainability	15	PM2	PC, Researcher, Google Spreadsheets
PM4	Integration in the Final Document	20	PM1, PM2, PM3	PC, Researcher, GEP Tutor
PM5	Weekly Meetings	15		PC, Researcher, Director
R	**Research**	**120**		
R1	RBFN Networks	45		PC, Researcher, Papers, Director
R2	Activation Function Selection	45		PC, Researcher, Papers, Director
R2	Problem Selection	30		PC, Researcher, Papers, Director
IT	**Implementation and Testing**	**130**		
IT1	RBFN Networks Implementation	50	R1, R2	PC, Researcher, VSCode, GitHub, Papers, Director
IT2	RBFN Network Training for Different Problems	50	R1, R2	PC, Researcher, VSCode, GitHub, Papers, Director
IT3	Test and Measurement	40	IT2	PC, Researcher, VSCode, GitHub, Papers, Director
A	**Analysis**	**60**		
A1	Model Comparison	30	IT3	PC, Researcher, Director, Obtained Results
A2	Conclusion Extraction	30	R, IT3	PC, Researcher, Director, Obtained Results
D	**Delivery**	**90**		
D1	Project Documentation	70		PC, Researcher, Overleaf, Obtained Results
D2	Oral Defense Preparation	20	D1	PC, Researcher, Project Documentation, Obtained Results
Total		**485**		

Table 5: Summary of the tasks to be performed

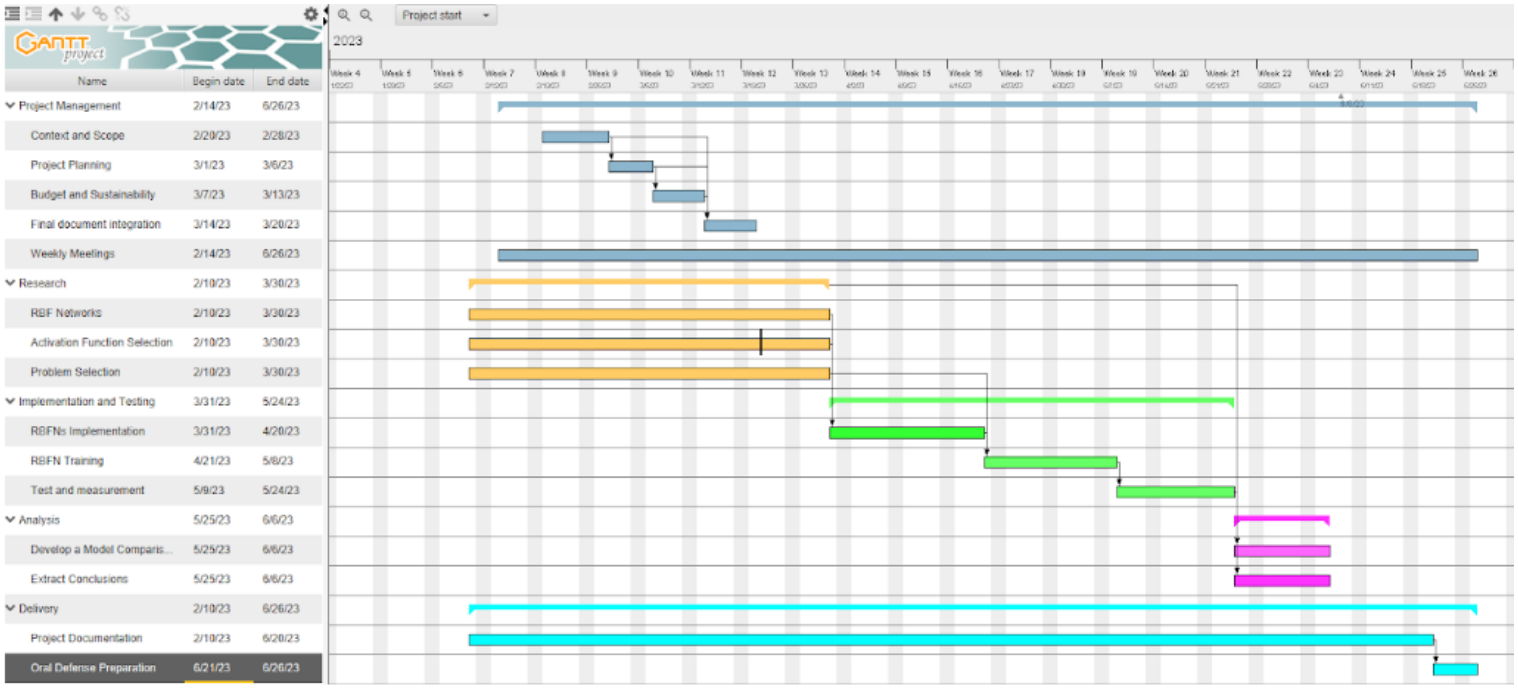

Figure 12: Gantt Diagram

8.4 Budget and Sustainability

In this section, a budget estimation has been done by identifying the different costs necessary for the project's correct realization, along with an estimation of the total needed budget.

8.4.1 Budget

Personnel Costs

An essential part inside the various costs in this study are the personnel costs. To give an approximate cost of the different tasks presented, each one has been be assigned to an individual or a personnel. Then the cost of a task has been calculated by multiplying the personnel's cost per hour by the needed amount of time. The following calculations have been done with three types of personnel:

- **Technical Writer**: This type of personnel is focused on the writing of the required documentation, which includes the Project Management specifications, the development of this project and its results.
- **Programmer**: This role is dedicated to the implementation and development phases of the project, where the usage of programming languages and measurement tools will be necessary.
- **Researcher**: This personnel's objective is to search and collect information about the fields of Radial Basis Function Networks and Activation Functions. From the compiled information and the developed study, it is also his responsibility to extract conclusions and plan possible extensions for the project.

The researcher of this study has played the different types of personnel presented, but the GEP tutor and the project's director can also play the technical writer and researcher roles when proposing corrections and suggestions.

The salary of each described role [12] is shown in the following table:

Role	**Annual Salary (€)**	**Total with SS (€)**	**Cost / h (€)**	**Role played by**
Technical Writer	26.263	34.191,90	19,5	R, GEP Tutor, Director
Programmer	26.198	34.057,40	19,46	R, Director
Researcher	35.259	45.836,70	26,19	R, Director

Table 6: Annual Salary of the presented personnel

The total calculated cost for each task (known as *CPA*) is summarized in table X:

Activity	**Import (€)**	**Activity hours**	**Tech. writer hours**	**Programmer hours**	**Researcher hours**
PM - Project Management	**1.707,375**	**85**	**70**	**7,5**	**7,5**
PM1 - Context and Scope	487,5	25	25	0	0
PM2 - Project Planning	195	10	10	0	0
PM3 - Budget and Sustainability	292,5	15	15	0	0
PM4 - Integration in the Final Document	390	20	20	0	0
PM5 - Weekly Meetings	342,375	15	0	7,5	7,5
R - Research	**3.142,8**	**120**	**0**	**0**	**120**
R1 - RBFN Networks	1.178,55	45	0	0	45
R2 - Activation Function Selection	1.178,55	45	0	0	45
R3 - Problem Selection	785,7	30	0	0	30
IT - Implementation and Testing	**2.529,8**	**130**	**0**	**130**	**0**
IT1 - RBFN Networks	973	50	0	50	0
IT2 - RBFN Network Training for Different Problems	778,4	40	0	40	0
IT3 - Test and Measurement	778,4	40	0	40	0
A - Analysis	**1.470,45**	**60**	0	**15**	**45**
A1 - Model Comparison	684,75	30	0	15	15
A2 - Conclusion Extraction	785,7	30	0	0	30
D - Delivery	**1.888,8**	**90**	**70**	**0**	**20**
D1 - Project Documentation	1.365	70	70	0	0
D2 - Oral Defense Preparation	523,8	20	0	0	20
Total	**10.215,43**	**485**	**140**	**152,5**	**192,5**

Table 7: Total calculated personnel cost for each task defined

General Costs

Amortization: Another important aspect inside the total calculated cost is the amortization. To calculate the amortization of the material resources used in this project, it has been considered an average daily usage of 3.5 hours during a total of 136 days in a four-year lifespan. It has also been considered that 80% of the project has been done with the laptop, and the remaining 20% with the desktop computer.

The following formula has been used for the amortization calculations:

$$Amortization = \frac{Resource\ cost \cdot Usage\ hours}{4\ years \cdot 136\ hours \cdot 3,5\ daily\ hours}$$

Hardware	**Price (€)**	**Time used (h)**	**Amortization (€)**
Personal Desktop Computer	1.600	97	81,51
Laptop Computer	1.000	388	203,78
Total	**2.600**	**485**	**285,29**

Table 8: Amortization costs of the employed resources

Electric Cost: Currently, the cost of electricity, represented by its €/kWh, is around 0,1638€/kWh [31]. After calculating the electric consumption cost of the different hardware used, the following table is obtained:

Hardware	**Power (W)**	**Time used (h)**	**Consumption (kWh)**	**Cost (€)**
Personal Desktop Computer	420	97	40,74	6,67
Laptop Computer	180	388	60,84	11,44
Total	**600**	**485**	**101,58**	**18,11**

Table 9: Electric cost of the used hardware

Internet Cost: By the personal experience of the researcher, it has been assumed a cost of 32€ per month in internet services. Considering a duration of 5 months, the estimated cost in internet connection is obtained by multiplying the monthly cost with the number of months, which results in a total of 160€.

Total General Cost:

Concept	**Cost (€)**
Amortization	285,29
Electric Cost	18,11
Internet Cost	160
GC Total	**463,4**

Table 10: Total General Cost (GC) of this study

Other Costs

Contingency Plan: After calculating the personnel cost and the total general cost, it is also important to add a contingency margin in case of unforeseen problems or incidences. For this project, an additional 15% of the initially calculated cost has been added in order to cover unexpected problems.

$$Contingency\ Cost = (10.215,43 + 463,40) \cdot 0,15 = 1.601,82$$

Incidental Costs: For the possible incidents planned in Section 8.3.3, an additional cost has been added depending on each incident's risk and the cost of its proposed solution.

Incident	**Cost (€)**	**Risk (%)**	**Final cost (€)**
Library Issues	120,28	10	12,03
Delivery Delays	485,51	30	145,65
Limited Existing Research	168,03	50	84,01
Computational Complexity	59,5	40	23,8
Total	**833,32**	**130**	**265,49**

Table 11: Incidental costs of the project

Total Cost

From the previous sections, the following total cost has been calculated:

Activity	**Cost (€)**
CPA Cost	10.215,42
GC Cost	463,4
Contingency	1.601,82
Incidental Costs	265,49
Total	**12.546,13**

Table 12: Total cost required for the project

Management Control

In order to control the budget estimations correctly during the development of the study, it is recommended to establish indicators and management control mechanisms to regulate the potential budget deviations. In this section, deviation formulas are presented, and will be used throughout the project to correct the budget differences.

- **Personnel Deviation**: It is caused when the planned personnel's dedicated hours don't correspond with the scheduled timeline.

  $$(\text{Planned cost per hour} - \text{Real cost per hour}) \cdot \text{Total hours consumed}$$

- **Amortization Deviation**: It is caused by using more or less time the mentioned hardware resources.

  $$(\text{Planned usage hours} - \text{Real usage hours}) \cdot \text{Cost per hour}$$

- **Incidental Costs Deviation**: It is caused by a bad estimation on the incidental costs initially planned.

  $$(\text{Planned incidental hours} - \text{Real incidental hours}) \cdot \text{Total incidental hours}$$

To calculate the general costs deviation from the described indicators, the following formula is used:

Cost Deviation = Personnel Deviation + Amortization Deviation + Incidental Costs Deviation

8.5 Sustainability

8.5.1 Self-assessment

In recent years, sustainability has risen as an important issue, given the need to balance economic development, social progress and environmental protection. To ensure a sustainable future, it is essential to consider all dimensions of sustainability, including the economic, environmental and social dimensions.

The completed poll has made me think that many people associate sustainability with products that minimize harm to the environment and society, promoting economical balance. However, it is also important to consider the economic dimension of sustainability, which requires additional thinking about economic growth.

In conclusion, I consider that to achieve sustainability, it is essential to measure the impact of projects in each of its dimensions before the required implementation. Doing so enables the identification of any sustainability issues that may happen, and the use of available tools and resources to address them.

8.5.2 Economic Dimension

Regarding PPP: Reflection on the cost you have estimated for the completion of the project:

The estimated cost of the project has been estimated reasonably, and each part has been carefully measured to be carried out in a real situation.

Regarding Useful Life: How are currently solved economic issues (costs) related to the problem that you want to address (state of the art)?

It is very possible to suffer economic issues in a project with this complexity, for each economic issue, a plausible solution has been presented. These solutions can vary the total economic cost, but without causing a high raise.

How will your solution improve economic issues (costs) with respect to other existing solutions?

Currently, the economic cost necessary to train a complex RBF Network is high. Improving its efficiency and temporal training cost will possibly reduce the needed economic cost.

8.5.3 Environmental dimension

Regarding PPP: Have you estimated the environmental impact of the project?

Having in mind the nature of this project, its impact has not been directly stated. This has been done based on the low environmental impact it has.

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources?

Some parts during the implementation and training of the different models will probably use libraries or pre-trained models to minimize the required electrical consumption.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?

To determine if some of the presented new models perform better than the existing solutions, the best way is to make a wide comparison using different real data science problems.

How will your solution improve the environment with respect to other existing solutions?

As stated before, the proposed new models may reduce the necessary electric and time consumption in case a better performance is achieved, therefore improving the environment.

8.5.4 Social dimension

Regarding PPP: What do you think you will achieve in terms of personal growth from doing this project?

This project will introduce me to research, and will make possible the learning of newly arisen topics, further investigating them. This study will make me learn about project management and tasks organization.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?

There currently exist very efficient designs of RBF Networks, most of them being specialized in a selected range of problems. This limits the usage of each model, but helps to keep a low complexity training.

How will your solution improve the quality of life (social dimension) with respect to other existing solutions?

This solution may present better performance models that may help programmers and ML users to spend less time in training.

Regarding Useful Life: Is there a real need for the project?

In order to maintain a dynamic evolution in the field of artificial intelligence and machine learning, it is always needed to keep investigating better performing models.

8.6 Changes and Difficulties

The project planning initially made in GEP has been followed throughout the project's development. At its current stage, some eventual deviations have arisen, affecting the project's objectives and time distribution.

The first deviation has happened during the research part of the project, specifically during the Activation Function Selection stage. This stage was especially important, given that the selected Activation Function would directly affect the model's predictions, performance and implementation complexity. Different functions have been considered, such as the Ormsby or the Klauder functions, and support programs have been developed in order to study their hyperparameters. The before unnoticed complexity of finding a good hyperparameter estimation method has resulted in a higher research time span. In order to control the mentioned research time increase, only the Ricker Wavelet function has been selected as an activation function, as it only has one hyperparameter. This solution has helped reduce the research time, but has affected the objective of selecting multiple possible activation functions.

A second difficulty has also affected the project in later stages. The development of the different RBFNN models has been more complex than expected, thus resulting in more needed time to finish the implementation. The reason for this time miscalculation is a lack of Python statistical libraries, with the existing ones being limited and unfinished. In R and other programming languages, these libraries are more abundant and complete, and include methods like sigest or step, which have been manually implemented for this project.

In terms of cost increase, it has been necessary to increase the number of dedicated hours for the research and implementation stages. The following table shows the additional cost with these hours:

Activity	**Import (€)**	**Extra Activity hours**	**Tech. writer hours**	**Programmer hours**	**Researcher hours**
R2 - Activation Function Selection	392,85	15	0	0	15
IT1 - RBFN Networks	233,52	12	0	12	0
Total	**626,37**	**27**	**0**	**12**	**15**

Table 13: Additional cost of the project with the new dedicated hours

8.7 Knowledge Integration

8.7.1 Programming Knowledge

The implementation phase of the project requires a good understanding of programming languages, as it is needed to interpret and understand code from different sources as well as knowing about various data structures and performance optimization algorithms. For this task, the knowledge gained in different subjects has been applied. These are Programming 1 and 2, EDA and Algorithmics.

8.7.2 Statistics and Artificial Intelligence

Statistics is the base of Artificial Intelligence models, and it is a universal tool for measuring and validating a study. In this project, Generalized Linear Models are used, along with the gathering of different statistical metrics. The PE subject has helped to better understand the Generalized Linear Models, and has eased the interpretation of the regularization process (throughout Aikaike's Information Criterion) and the model's results.

In the same way, the Artificial Intelligence subject has provided the necessary knowledge to correctly understand various concepts about RBFNNs, such as the activation functions, the design matrix or cross validation.

Some M1 and M2 concepts have also been used for matrix multiplications, various initial tests with a gradient descent algorithm and the calculation of euclidean distances.

8.7.3 Other Knowledge Fields

Different neuroscience articles have provided useful information about lateral inhibition and how it can affect a neuron's output. Moreover, Kukjin Kang et al. article provides evidence and investigation of Mexican-hat shaped electric signals present in the visual cortex.

The majority of the found activation functions, along with their parameters have been studied in the seismics field. These include functions such as the Ricker, Ormsby and Klauder Wavelets. Although this field is not related with this research, it has provided practical information about the studied basis functions.

8.8 Laws and Regulations

In this project there are no laws or regulations which directly affect it. However, third-party software is used, and it can have license and citation policies.

The usage of Python as the selected programming language for this project requires the agreement of the Python Software Foundation License (PSFL). This license grants all users the right to run, distribute, modify, study and copy its software. Nevertheless, it is not copyleft. As Python is used to run and distribute the developed RBFNN models, the PSFL terms and conditions are not breached.

For the development of a basic RBFNN model, a sigest method has been developed, based on its R implementation included in the Kernlab package. This package has a GPL-2 license, which grants users the ability to reproduce its code.

Once the implementation is finished, it is also possible that the datasets selected for the experimentation phase have a license under certain conditions and regulations. Therefore, it is important to be informed about their licenses in order to not violate any of their terms of usage.

References

[1] Hirotogu Akaike. *Information Theory and an Extension of the Maximum Likelihood Principle*, pages 199–213. Springer New York, New York, NY, 1998.

[2] Gardave S Bhumbra. Deep learning improved by biological activation functions. *arXiv preprint arXiv:1804.11237*, 2018.

[3] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[4] Dirk Brockmann. Keith haring's mexican hat. `https://www.complexity-explorables.org/explorables/keith-harings-mexican-hat/`. Last Accessed: March 2023.

[5] David S. Broomhead and David Lowe. Multivariable functional interpolation and adaptive networks. *Complex Syst.*, 2, 1988.

[6] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

[7] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10(7):1895–1923, oct 1998.

[8] K L Du and MNS Swamy. *Radial Basis Function Networks*, pages 251–294. Springer London, London, 2006.

[9] S. Gholizadeh, E. Salajegheh, and P. Torkzadeh. Structural optimization with frequency constraints by genetic algorithm using wavelet radial basis function neural network. *Journal of Sound and Vibration*, 312(1):316–331, 2008.

[10] J. Ghosh and A. Nag. *An Overview of Radial Basis Function Networks*, pages 1–36. Physica-Verlag HD, Heidelberg, 2001.

[11] Albert Gidon, Timothy Adam Zolnik, Pawel Fidzinski, Felix Bolduan, Athanasia Papoutsi, Panayiota Poirazi, Martin Holtkamp, Imre Vida, and Matthew Evan Larkum. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 367(6473):83–87, 2020.

[12] Glassdoor.es. Annual salaries in spain. `https://www.glassdoor.es/`. Last Accessed: March 2023.

[13] Patric Hagmann, Leila Cammoun, Xavier Gigandet, Reto Meuli, Christopher J Honey, Van J Wedeen, and Olaf Sporns. Mapping the structural core of human cerebral cortex. *PLOS Biology*, 6(7):1–15, 07 2008.

[14] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York, 2013.

[15] Kukjin Kang, Michael Shelley, and Haim Sompolinsky. Mexican hats and pinwheels in visual cortex. *Proceedings of the National Academy of Sciences*, 100(5):2848–2853, 2003.

[16] Alexandros Karatzoglou. Hyperparameter estimation for the gaussian radial basis kernel. `https://rdrr.io/cran/kernlab/man/sigest.html`. Last Accessed: June 2023.

[17] Richard H Kramer and Christopher M Davenport. Lateral inhibition in the vertebrate retina: The case of the missing neurotransmitter. *PLoS Biol.*, 13(12):e1002322, December 2015.

[18] Elisabeth Larsson, Krister Åhlander, and Andreas Hall. Multi-dimensional option pricing using radial basis functions and the generalized fourier transform. *Journal of Computational and Applied Mathematics*, 222(1):175–192, 2008. Special Issue: Numerical PDE Methods in Finance.

[19] Hamdy F. F. Mahmoud. Parametric versus semi and nonparametric regression models. *arXiv*, (1906.10221), 2019.

[20] Ian T. Nabney. Efficient training of rbf networks for classification. In *9th International Conference on Artificial Neural Networks*, pages 210–215, GBR, 1999. Volume 1 ISSN - 0537-9989.

[21] Matthew Mithra Noel, Shubham Bharadwaj, Venkataraman Muthiah-Nakarajan, Praneet Dutta, and Geraldine Bessie Amali. Biologically inspired oscillating activation functions can bridge the performance gap between biological and artificial neurons. *arXiv preprint arXiv:2111.04020*, 2021.

[22] Sung-Kwun Oh, Wook-Dong Kim, Witold Pedrycz, and Byoung-Jun Park. Polynomial-based radial basis function neural networks (p-rbf nns) realized with the aid of particle swarm optimization. *Fuzzy Sets and Systems*, 163(1):54–77, 2011. Theme: Classification and Modelling.

[23] Mark JL Orr. Recent advances in radial basis function networks. *Institute for Adaptative and Neural Computation*, pages 25–29, 1999.

[24] Mark JL Orr et al. Introduction to radial basis function networks. *Centre for Cognitive Science*, 1996.

[25] H Ryan. Ormsby, klauder, butterworth-a choice of wavelets. *CSEG Recorder*, 19(7):8–9, 1994.

[26] S.A. Sarra. Integrated multiquadric radial basis function approximation methods. *Computers and Mathematics with Applications*, 51(8):1283–1296, 2006. Radial Basis Functions and Related Multivariate Meshfree Approximation Methods: Theory and Applications.

[27] Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics*, 48(4):1875 – 1897, 2020.

[28] Bernhard Scholkopf, Kah-Kay Sung, Christopher JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765, 1997.

[29] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[30] SOCR Statistics. 25,000 records of human heights (in) and weights (lbs). `http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights`. Last Accessed: June 2023.

[31] Tarifasgasluz.com. Kwh price in spain. `https://tarifasgasluz.com/comercializadoras/endesa/precio-kwh`. Last Accessed: March 2023.

[32] Turgay Temel. Biologically-inspired learning: An overview and application to odor recognition. *System and Circuit Design for Biologically-Inspired Intelligent Learning*, pages 59–92, 2011.

[33] Yiwei Zhou, Huanwen Chen, and Yijun Wang. Role of lateral inhibition on visual number sense. *Frontiers in Computational Neuroscience*, 16, 2022.