Analysis of learning algorithms for the similarity neural network

FACULTAT D'INFORMÀTICA DE BARCELONA

Pravallika Mallela
(SASTRA University)

Thesis Supervisor: Prof. Lluís Belanche

Abstract

A similarity neural network is a two-layer neural network based on similarity measures: the first layer computes the similarity between the input data and a set of prototypes, and the second layer gathers these results and predicts the output. The goal of the project is to analyse fast training algorithms and compare with more traditional learning algorithms, such as a standard two-layer MLP network. Along with the usage of similarity measures, this research also aims to develop a model that can handle complex, heterogeneous, imprecise, missing and complex data without changing their original form.

List of Figures

List of Tables

Context:

This Project is a Final Project Degree (TFG) at Barcelona School of Informatics supervised by Lluís Belanche. This project aims to design a two-layered similarity neural network and to understand to analyze fast training algorithms and compare with more traditional learning algorithms, such as standard two-layered MLP network.

1. Introduction

This thesis discusses the development of a neural network model to be used in Artificial Neural Networks(ANNs). These networks are the interconnected group of nodes, inspired by a simplification of neurons in a brain. Similar to the neurons in the brain, the neurons in ANN are connected to one another with a certain weight that has information about the input signal. This is the most useful information for neurons to solve a particular problem, as the weight usually excites or inhibits the communicated signal. Each neuron has an internal state, known as the activation signal. Output signals that are produced after the input signals and the activation rule are combined can be sent to other units.

Here, in this research, a general framework for dealing with data heterogeneity in ANNs under the similarity concept, in which the neuron model accepts the heterogeneous inputs and weights and computes the similarity function between input and the weights. The resulting model accepts mixtures of continuous and discrete quantities, additional to the missing information.

1.1. Similarity Neural Network

An SNN consists of a feed-forward multilayer perceptron (MLP) trained to learn a similarity measure between a couple of patterns. The SNN model consists of a single hidden layer with an even number of units and a unique output neuron encloses the range of [0,1] with a sigmoidal activation function. The hidden layers are fully connected with both the input and the output nodes. The similarity function is calculated using the similarity measure between the two vectors. This similarity measure is calculated using different similarity functions for different data types.

Similarity measure:

The state or fact of being similar or Similarity measures how much two objects are alike. A similarity measure in a data mining context is a distance with dimensions representing features of the objects. If the distance is small, two objects are very similar whereas if the distance is large we will observe a low degree of similarity. Similarities are usually non-negative and are often between 0 (no similarity) and 1(complete similarity).

For a similarity function $s: X \times X \rightarrow \mathbb{R}$,

$s_{ij} = s(x,y)$ for any $x, y \in X$

Similarity measure may be required to satisfy the certain axiom

Reflexivity: $s(x,x) = s_{max}$

Consistency: $s(x,y)=s_{max} \Rightarrow x = y$

Symmetry: $s(x,y) = s(y,x)$

Boundedness: A similarity s is lower bounded when inf exists

.

1.2. Heterogeneity in neural networks

Real-world data comes in many types and sources with a mixture of different data types and nature: numeric, non-numeric, qualitative, ordinal, nominal, images, text, signals, etc. The data also comes with incompleteness and uncertainty, additional to the original heterogeneity.

Many data analysis methods work on single- type data or only allow a few types. Limiting the number of features results in the partial study of the information available. Important relationships between the variables may not be processed together. The Extreme Learning Machine (ELM) is capable of processing heterogeneous data with neural networks.

1.3. Extreme Learning Machine

Extreme learning machines are feedforward neural networks for classification, regression, clustering, sparse approximation, compression, and feature learning with a single layer or multiple layers of hidden nodes, in which the parameters of hidden nodes (not just the weights connecting inputs to hidden nodes) need not be tuned. These hidden nodes can be assigned randomly and never updated (i.e. they are random projections but with nonlinear transforms), or they can be inherited from their ancestors without changing. The output weights of hidden nodes are typically learned in a single step in most situations, which basically amounts to a linear model being learned.

Unlike traditional feedforward network learning algorithms like backpropagation (BP) algorithm, the ELM does not use a gradient-based technique. With this method, all the parameters are tuned once. This algorithm does NOT need iterative training.

ELM Implementation:

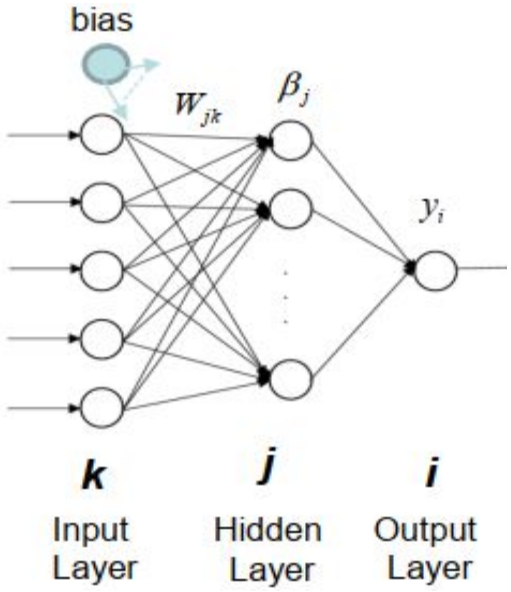

Figure 1.1.

1. Creating a matrix and bias for the input layer by random weights.

The weight matrix and bias size is (j□k) and (1□k) where the number of hidden nodes is j, and the number of input nodes is k.

$$W = \begin{bmatrix} rand & \dots & rand \\ \vdots & \ddots & \vdots \\ rand & \dots & rand \end{bmatrix},$$

2. Calculate the output matrix of the hidden layer.

The initial hidden-layer output matrix is determined by multiplying X which is training data with weight matrix transposition

$$(H_{init} = X.W^T)$$

3. Choosing the activation function

The sigmoid activation function is used here. Any other activation function can also be chosen(Fourier, hard limit, gaussian, multiquadrics, etc.)

Sigmoid activation function:

$$H = \frac{1}{1 + e^{-Hinit}}$$

4. Calculate the **Moore-Penrose pseudoinverse**

The Moore – Penrose generalized inverse of H can be determined using many methods. These methods include orthogonal projection, orthogonalization method, iterative method, and singular value decomposition (SVD), but are not limited to.

$$H^{+} = \left(H^{T}.H\right)^{-1}.H^{T}$$

5. Calculate the output weight matrix

$$\hat{\beta} = H^{+}.Y$$

1.4. Brief Description of the Project

This project aims to design a neural network that accepts heterogeneous information. For dealing with data heterogeneity in artificial neural networks, the concept of similarity was developed, in which the similarity between heterogeneous inputs and the weights is calculated using the similarity function.

This similarity neural network is a two-layer neural network based on similarity measures: the first layer computes the similarity between the inputs and the set of prototypes, and the second layer gathers these results and predicts the output. The resulting model accepts mixtures of continuous and discrete quantities, additional to the missing information.

1.5. State of the Art

In the literature, there exists a different kind of learning algorithm that is focused on developing a neural model and handling heterogeneity. For example, the work developed by Julio J. Valdes [1], which design a model using extreme learning machines with heterogeneous data types and the research done by Lluis A. Belanche-Munoz [2] and [6], introducing a class of neuron models accepting heterogeneous inputs and weights and also discussing the theory and applications of the heterogeneous data. Taking into the reference of the extreme learning machine for both regression and the multiclass classification [3] and [8] proposed by Guang-Bin Huang, ELM model was suggested as one of the faster learning algorithm where he suggested the usage of the single hidden layer feedforward network with N hidden nodes and with usage of any non-linear activation function. In this paper it first rigorously shows that SLFNs' input weights and hidden layer biases can be assigned randomly if the activation functions in the hidden layer are infinitely differentiable. After random selection of input weights and hidden layer biases, SLFNs can be considered simply as a linear system, and the output weights (linking the hidden layer to the output layer) of SLFNs can be analytically determined by simple generalized inverse operation of the hidden layer output matrices. Based on this concept, this paper proposes a simple learning algorithm for SLFNs called extreme learning machine (ELM) whose learning speed can be thousands of times faster than traditional feedforward network learning algorithms such as back-propagation (BP) algorithms while at the same time achieving better performance in generalisation. The proposed learning algorithm, unlike conventional learning algorithms, appears to meet not only the smallest training error, but also the smallest weight norm.This paper also claims that ELM tends to have better

scalability and achieve similar or much better generalization performance at uch faster learning speed.

The similarity metrics that are used are considered here by taking account of the definition given by Dekang Ling, in An Information-Theoretic Definition of Similarity [4] where all the metrics should be satisfying certain similarity conditions and axioms that define similarity. While taking account of similarity metric for fuzzy type data , the new similarity measure for generalized Trapezoidal Fuzzy numbers proposed by Xin Zuo, Lijun Wang, and Yuanlong Yue in "A New Similarity Measure of Generalized Trapezoidal Fuzzy Numbers and Its Application on Rotor Fault Diagnosis" [5].

Here, in this TFG we are interested in developing a neural model that aims to directly work on heterogeneous, imprecise and incomplete data and not by modifying the data into homogeneous lower-dimensional spaces, approximately equivalent in information content by considering the similarity metrics. And the similarity network built will also be analyzed with the other learning algorithms for the heterogeneous input.

1.6. Justification and existing solution

The requirement for this solution to be built is to handle heterogeneity in the data which leads to access to all the real-world examples that contain different types of data like real, ordinal, nominal, categorical, fuzzy, images and text directly without converting them into the homogeneous less dimensional spaces . This project is also designed to analyze and identify the algorithm among other traditional algorithms that can give better accuracy and efficiency. Accordingly, extending their scope by making Extreme learning machines (ELM) capable of processing heterogeneous information for addressing complex problems. Also, a number of experiments are carried out using several real-world benchmarking problems.

The existing solutions and machine learning methods do not handle heterogeneity well, requiring variables of the same type, information completeness. This research takes account of all the proposed methods and the existing solutions and analyses practically.

This thesis also desires to analyse the similarity neural network built with the other clustering methods and compare with extreme learning machines . Hence, building this model will cover new ground in research areas involving Neural network, handling heterogeneity and Extreme learning machines.

2. Scope and Objective

This thesis will first implement a similarity based-heterogeneous neural network using extreme learning machines. The neural network will be developed from scratch.

The project will be divided into the following stages:

1. Study and understand how heterogeneity data including missing values can be handled.
2. Creating a synthetic heterogeneous data set by sampling which are independent and identically distributed
3. Injecting missing values into the synthetic dataset
4. Calculating the similarity matrix by taking account of different similarity functions for different types of data types.
5. To learn and optimize the weights using the training data.
6. Adding the target variable to be predicted by the similarity neural network for both regression and classification.
7. Compare with several clustering methods(PAM, random, leader2)
8. Develop a user interface that facilitates the user in loading data for prediction.

2.1. Objective

The work is to develop a neuron model with the conceptual use of the similarity. Heterogeneity of the data is an additional account taken into account.

The general objective of the project is as follows:

- The construction of an efficient neuron model with high speed and accuracy.
- The integration of concepts of heterogeneity and similarity
- To extend the similarity measures to other data types (ordinal, fuzzy, interval, graph)

- To learn and optimize the weights using the training data.
- To add non-linearity to the final similarity.
- To add regularization to the learning process.
- To learn the parameter value that controls curvature of the sigmoid activation function.
- Considering ELM to train the neural network.
- Compare other traditional learning methods with ELM.

2.2. Scope of the Thesis

We limited the practical extent of the experiments in feedforward architectures to supervised learning. Limited to the inclusion of similarity measures over the most commonly encountered types of data (such as real, nominal, ordinal, sets, and fuzzy information forms). However, it may also include other less common forms

2.3. Requirements

A workstation with sufficient computation capability.
Prior knowledge of extreme learning machines.
A dataset with well-defined heterogeneous input and output values to be trained and tested to validate the model.

2.4. Possible Obstacles

As here we are going to analyze the learning methods for the heterogeneous data, the calculation of similarity measures for the different types of data to be done, which involves high coding.

The project to be developed has a limited time period. So, the project has to be implemented successfully and to complete the given task before meeting the deadlines.

When dealing with predictions for output, close supervision and being able to understand the outcome of output is not an easy task. To ensure the results are properly interpreted, I will use all the machine learning that I have learned before, ensuring there is no error.

3. Methodology

The methodology to be followed during the development process of the project is the iterative model. To change the design and to integrate other modules into the designed model at any given time, going back and accessing planning and requirement stages must be done repeatedly. Since this is an individual project, the individual must be extremely precise in evaluation and testing. All the stages of the development cycle must be gone through carefully to make sure the final output meets the requirements.

The iterative methodology for the development process is divided into the following stages:

1. *Planning Phase*: This is the first stage of the iterative model, where proper planning is done, which helps in mapping out the specifications documents, establish requirements and generally prepare for the upcoming stages of the cycle.
2. *Analysis and Design Phase*: Once the planning is complete for the cycle, an analysis is performed to point out the appropriate logic, models and to know any other requirements of this particular stage. Moreover, the design stage also occurs in this phase of the iterative model, where the technical requirements are established that will be utilized in order to meet the need of the analysis stage.
3. *Implementation Phase*: This is the third and most important phase of the iterative model. Here, the actual implementation and coding process is executed. All planning, specification, and design documents up to this point are coded and implemented into this initial iteration of the project.
4. *Testing Phase:* After the current build iteration is coded and implemented, testing is initiated in the cycle to identify and locate any potential bugs or issues that may have been in the software.

5. *Evaluation Phase:* The final phase of the Iterative life cycle is the evaluation phase, where we examine the status of the project and validate whether it is meeting the objectives and requirements.

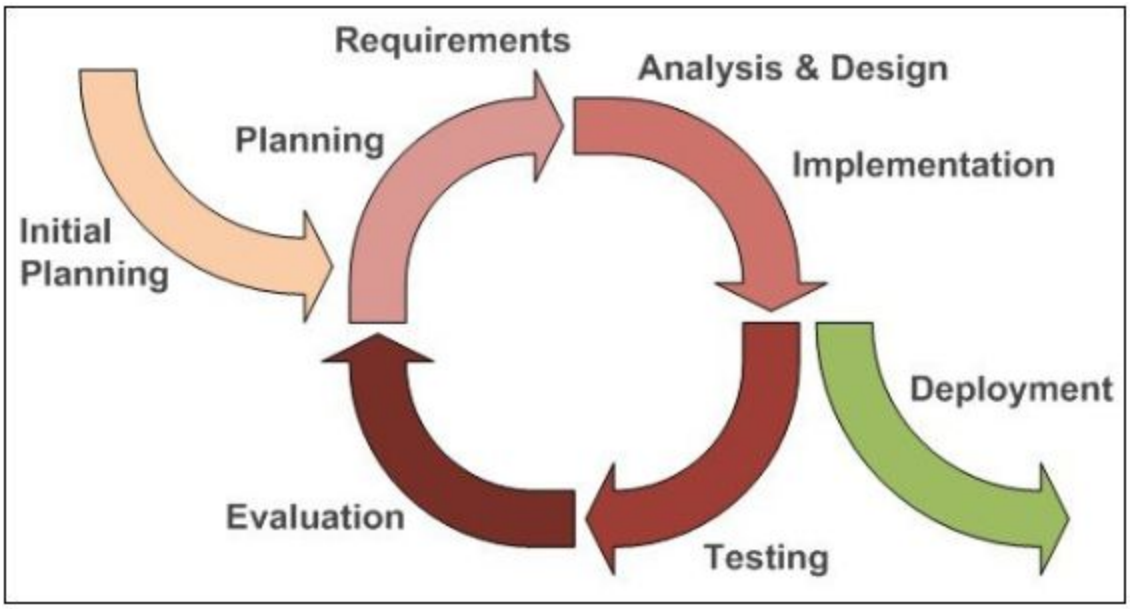

Figure 3.1.

4. Project Planning

4.1.Duration Of the Project

The estimated project duration is approximately 5 months, starting on 17th February 2020 and ending on 3rd July 2020 (the total duration of 138 days). The number of work hours is calculated as 420, with an average of 3 hours per day. The final defense for the project is scheduled for the 3rd of July 2020

4.2. Task Definition

The project is divided into the following phases, further task subgroups have been defined and listed out in Table 4.1.

Project Management

- P1- Context and Scope: Defining the context and Scope of the project takes about 15 hours and requires MS word.
- P2- Time Planning: For making the time plan for this project, it takes 10 hours and requires both MS word and Gantt project.
- P3- Budget, and Sustainability: Planning budget involved for each task and developing the sustainability report for the project requires about 10 hours and working with MS word.
- P4- Project Definition: Combining all the task documents mentioned above and generating Project Definition using MS word. Thus, this task requires the completion of P1, P2, and P3. It takes about 10 hours.
- P5- Meetings: Weekly meetings with the director of the project to analyze the progress of the project and to get suggestions about the changes to be made in the work done. Also discussing the further implementation. This also means that the ideas discussed in the meetings are implemented and updated weekly to the director. This takes about 30 hours.

Analysis

- A1- Study of research topic: The publications using extreme learning machines and similarity neural networks are studied to understand the concepts required to proceed with the project implementation. This takes about 25 hours.
- A2- Study of existing algorithms: Existing learning algorithms are analyzed and studied to be integrated into the model. This takes about 20 hours.
- A3- Study of similarity measures: Similarity measures function for different types of heterogeneous data to be studied and identified. This takes 10 hours.
- A4- Defining the functionalities: The functionality of the project along with its objective is defined. This takes about 10 hours.

Design, Implementation and Testing

- D1- Applying similarity measures: By using the heterogeneous dataset the similarity measures between the same types of data are measured and the neural network is designed. This takes about 70 hours.
- D2- Applying extreme learning machine algorithm: The ELM algorithm is applied to training data for the similarity neural network designed. This takes about 100 hours.
- D3- Analyzing with other traditional algorithms: This trained neural network using ELM is analyzed and tested with various other traditional learning algorithms to measure the accuracy. This takes about 50 hours.

Documentation and Verification:

- V1- Verification: To be verified to know whether the project meets all the requirements. It takes about 10 hours.
- V2- Documentation: The whole work done is to be documented. This takes about 40 hours.
- V3- Oral Presentation: Preparing for the final oral presentation of the project. It takes about 10 hours.

	Task	**Time Required**	**Dependencies**
	Project Management:		
P1	Context and Scope	15	
P2	Time Planning	10	
P3	Budget and Sustainability	10	
P4	Project Definition	10	P1, P2, P3
P5	Meetings	30	
	Analysis:		
A1	Study of the research topic	25	P4
A2	Study of existing algorithm	20	A1
A3	Study of similarity measures	10	A1
A4	Defining the functionalities	10	A3
	Design, Implementation, and Testing		
D1	Applying similarity measures	70	P4, A3
D2	Applying extreme learning machines	100	D1
D3	Analyzing with other traditional learning algorithms	50	P4, A2, D2
	Documentation and Verification:		
V1	Verification	10	D1, D2, D3
V2	Documentation	40	V1
V3	Oral Presentation	10	V2
	TOTAL	**420**	

Table 4.1.

4.3. Resources

The resources required for the development and the implementation of the project are listed below in Table 4.2.

Resources	**Mode**	**Task**
Laptop	A device used for coding, implementation, documentation, and studying	All
MS word	For Documentation	P1, P2, P3, P4
Gantt project	For generating Gantt Graphs	P2
R	Writing and executing code	D1, D2, D3
Jupyter Notebook	Writing code for similarity measures	D1
Google Chrome	For searching, studying and accessing datasets	All

Table 4.2.

4.4.Gantt Chart

The Gantt charts are provided for the given schedule in Fig 4.1(a) and 4.1(b).

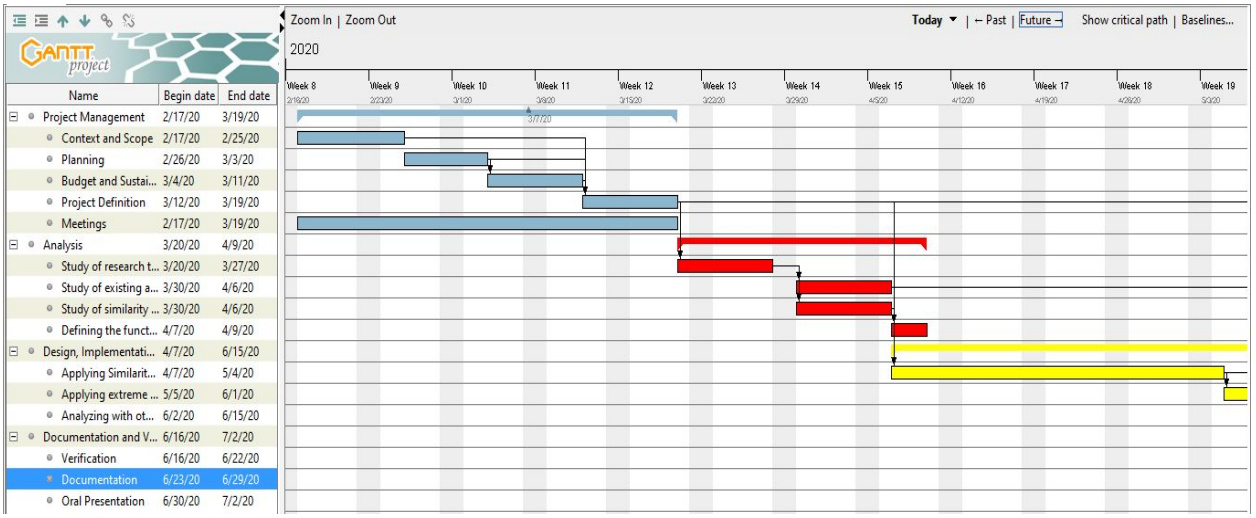

Figure 4.1(a).

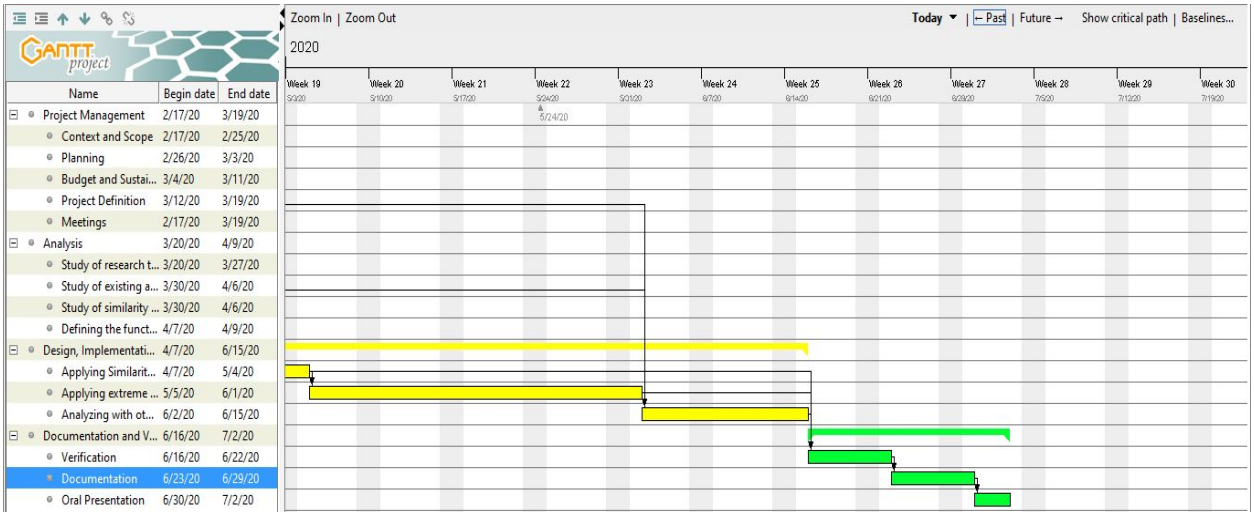

Figure 4.1(b).

4.5. Risk Management

The risks that could be encountered during the project implementation were already listed in deliverable 1. The same risks and the solutions to overcome the risks are considered as follows,

Complexity:

Keeping in mind the calculations of different similarity measures, the code is written is executed by tracking the running time and the space utilized.

Time Management:

As the project has to be implemented in a limited period of time, time allocation for each individual task is done based upon its complexity.

Efficiency:

When dealing with predictions for output, close supervision and being able to understand the outcome of output is not an easy task. To ensure the results are properly interpreted, all the concepts of algorithms studied theoretically will be used, ensuring that there is no error.

5. Estimation of Budget

5.1. Human Resource Budget

The Project will be carried out by a single student in collaboration with FIB with the guidance of the director. All the roles involved in the project are taken up by the student itself using the resources listed in the previous deliverable.

The roles required for this project are listed out in *Table 5.1* along with the cost per hour associated with the respective role. The cost associated with each role is calculated using the average salary wage quoted in [1]. Since the figures in the reference are meant for an individual with sufficient experience in the respective fields and for those who work 40 hours per week, the estimates listed for each role are decreased.

Role	**Estimated hours**	**Estimated cost per hour**	**Total estimated cost**
Project Manager	70	17€	1,190€
Data Scientist	150	13€	1,950€
Software Engineer	200	15€	3,000€
Total Estimated	**420 hours**		**6140€**

Table 5.1.

5.2. Non-Resource Human budget

The non-human resource budget involved in the project is the generic costs of hardware, software and living cost of the student.

Hardware:

In *Table 5.2,* an estimation of the cost of that hardware is provided taking into account their useful life and depreciation period.

Hardware	**Cost**	**Unit**	**Life**	**Estimated cost**
Lenovo Ideapad 510-15ISK	799€	1	4	799€
	Total			**799€**

Table 5.2.

Software:

All the software used in this project: Microsoft Word, Excel, GanttProject, R Studio, Jupyter Notebook are open source. Hence, no cost is incurred for the project.

Other costs:

Other costs involve the rent, internet, water, gas, food, and electricity for the student, this amounts to 500€/month. As I stay here for 5 months, the total living cost of the student is estimated to be 2500€.

5.3. Total budget

The total cost involved in the project is listed below in *Table 5.3*.

Concept	**Estimated budget**
Human Resources	6140€
Hardware	799€
Software	-

Other costs	2500€
Total	**9439€**

Table 5.3.

5.4. Budget Management

To ensure that the costs do not exceed the planned budget, management control must be done. The budget control is done by calculating costs after each phase of the project is completed, and the cost exceeded the estimated budget must be adjusted in the next phase.

The cost deviation is calculated as follows:

Cost deviation= |estimated cost - real cost|

If the estimated amount is higher than the real cost, then the left out amount can be passed on for the next phase.

6. Sustainability

Consideration of sustainability of the project and taking into consideration external factors like environment, economy and social impact must be done in order to make sure that the final product is sustainable in all the ways.

Environmental impact:

Since the project involves the usage of only a single laptop, the environmental impact is restricted to just the electricity usage of the laptop. The other environmental impact that the project may carry is internet services to perform research and to run the program, but these implications are minimal, and their impact on the environment is very low.

Economic impact:

The cost required to complete this project has been discussed and they involve human resources, hardware cost, software cost, and other costs. Hence, most of the costs are dedicated to human resources.

The model developed in this project is new, and the total costs are comparable to those that already exist, as hardware and software costs used here are minimal and only human resources are to be compared.

Social Impact:

Since this project trains the neural network model and analyses the different algorithms, the best and the efficient algorithm can be used for further prediction.

Nowadays, most of the data generated are heterogeneous (Example: patient record, containing real values, binary values, images like x-rays, heartbeat graphs) making it more difficult and time taking process for prediction of the outcome. This project is designed to take into account heterogeneous data and to find the best learning algorithm to train the model by considering all the features.

This is a necessary solution, as most of the solutions explained before do not deal with heterogeneous data or consider all the features.

7. Similarity Based Heterogeneous Neural models

Information composed of heterogeneous types of data may be described in a number of ways depending on which properties are mostly relevant to the kind of properties mostly relevant to the processing type needed. The two key approaches here are: (1) the creation of dedicated procedures that process the heterogeneous information directly, and (2) the transformation into a simplified representation that is suitable for more common machine learning methods.

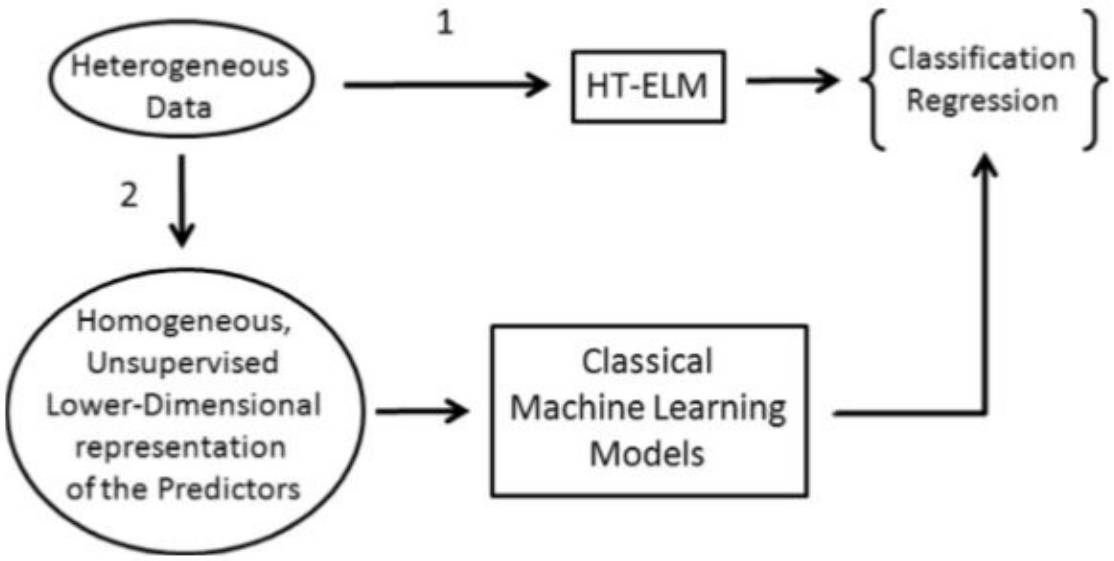

Fig 7.1. Two machine learning approaches for working with heterogeneous data: (1) Direct processing (example: HT-ELM models), (2) Transformation to a representation enabling standard methods to be used

The approach that we consider for working with the heterogeneous data here is the direct process , where we work directly with the heterogenous , imprecise and the incomplete data rather than converting them into homogenous lower dimensional spaces.

7.1. Heterogeneous Neural Model

The neural model allowing for heterogeneous and uncertain inputs, defined as a mapping h: $\widehat{H}^{n} \rightarrow R_{out} \subseteq \mathbb{R}$ is developed, Here $\mathbb{R}$ denotes the real and $\widehat{H}^{n}$ is a cartesian product of an arbitrary number n of source sets $\widehat{H}^{(k)}$,k=1…,n..These source sets may include $R_k = \mathbb{R}_k \cup \{\mathcal{X}\}$, extended fuzzy sets $\widehat{F}_k = F_k \cup \{\mathcal{X}\}$,finite sets $\widehat{O}_k = O_k \cup \{\mathcal{X}\}$, $\widehat{N}_k = N_k \cup \{\mathcal{X}\}$. The special symbol $\mathcal{X}$ extending the source set denotes the missing information. The neuron input are vectors composed of n elements containing real, fuzzy, ordinal, categorical and missing data.

Consider a function s: $\widehat{H}^{n} \times \widehat{H}^{n} \rightarrow [0,1]$ a similarity function in $\widehat{H}^{n}$. The function is formed by the combination of n partial similarities s_k. The s_k measures are normalized to a common real interval and computed according to different formulas for different variables. The neuron model can be now built supporting both similarity based and also handling the heterogeneity in the data.

A heterogeneous neural network is then built upon the existing neural model resulting in a hybrid structure integrating both similarity based network and the heterogeneous network.

7.2. Heterogeneous Similarity Measures

Here the value in the interval [0,1] is considered as the similarity measure. However, the other intervals can also be used.

In this thesis, the following data types are considered and the corresponding similarity measure is calculated.

Nominal : A variable with values which have no numerical value and with no definite order.

Ordinal: A numerical or non-numerical variable representing a linear order on a relation on a finite number of values.

Continuous: Continuous variables are numerical variables with an infinite uncountable number of values for two different values.

Fuzzy: A variable that shows some imprecise, linguistic, or vague properties.

Nominal variables.

For these types of variables the common similarity measure is overlap. Consider x,y ϵ N, where N is the nominal space

s(x,y) = 1 if x = y

s(x,y) = 0 if x $\neq$ y

Ordinal variables.

Assumed that the linear order exists between the values in a linear order space. Let O be the ordinal space and x,y ϵ O.

The similarity score for the ordinal variable is given by:

s(x,y) = 1 - $\frac{|x-y|}{card(O)-1}$, where card(O) denotes the cardinality of the set.

Continuous variables.

Let x,y ϵ $[r^-,r^+] \subset R$, $r^+ > r^-$, for any two values x,y ϵR the similarity metric is given by

s(x,y)=$\widehat{s}$ ($\frac{|x-y|}{sup|x-y|}$), where sup denotes the supremum: " least upper bound", it can often be considered as "max".

where $\widehat{s}$ =$(1- z)^{\alpha}$, $0 < \beta \leq 1$, $\alpha \geq 1$ is used with α =2 , β =1

Fuzzy variables.

Let $\overline{A}$ =(a_1,a_2,a_3,a_4,$w_{\overline{A}}$) and $\overline{B}$ =(b_1,b_2,b_3,b_4,$w_{\overline{B}}$) be two fuzzy numbers. The center of gravity is given by

$$y^*_{\overline{A}} = \frac{w_{\overline{A}} \times ((a_3 - a_2)/(a_4 - a_1) + 2)}{6} \quad \text{,if } a_4 \neq a_1,\ 0 \leq w_{\overline{A}} \leq 1$$

$$= \frac{w_{\overline{A}}}{2} \quad \text{,if } a_4 = a_1,\ 0 \leq w_{\overline{A}} \leq 1$$

$$x^*_{\overline{A}} = \frac{y^*_{\overline{A}}(a_3 + a_2) + (a_4 + a_1)(w_{\overline{A}} - y^*_{\overline{A}})}{2w_{\overline{A}}}$$

The normalization procedure is applied to $\overline{A}$ to avoid unexpected outcomes.

$$a'_1 = \frac{a_1 - min(a_1, b_1)}{max(a_4, b_4) - min(a_1, b_1)}$$
$$a'_2 = \frac{a_2 - min(a_1, b_1)}{max(a_4, b_4) - min(a_1, b_1)}$$
$$a'_3 = \frac{a_3 - min(a_1, b_1)}{max(a_4, b_4) - min(a_1, b_1)}$$
$$a'_4 = \frac{a_4 - min(a_1, b_1)}{max(a_4, b_4) - min(a_1, b_1)}$$

And a similar one is applied to $\overline{B}$.

Now the normalized fuzzy numbers will be $\overline{A}'$=(a'_1, a'_2, a'_3, a'_4, $w_{\overline{A}}$) and $\overline{B}'$=(b'_1, b'_2, b'_3, b'_4, $w_{\overline{B}}$), also satisfying $0 \le a'_1 \le a'_2 \le a'_3 \le a'_4 \le 1$ and $0 \le b'_1 \le b'_2 \le b'_3 \le b'_4 \le 1$ a'_i

The similarity is given by,

$$s(\overline{A}, \overline{B}) = s(\overline{A}', \overline{B}') =$$
$$= \left(1 - \frac{\sum_{i=1}^{4} |a'_i - b'_i|}{4}\right)^{1 - |x^*_{\overline{A}'} - x^*_{\overline{B}'}|}$$
$$\times \frac{min(P(\overline{A}'), P(\overline{B}')) + min(a(\overline{A}'), a(\overline{B}'))}{max(P(\overline{A}'), P(\overline{B}')) + max(a(\overline{A}'), a(\overline{B}'))}$$

where,

$$P(\overline{A}) = \sqrt{(a_1 - a_2)^2 + w^2_{\overline{A}}} + \sqrt{(a_3 - a_4)^2 + w^2_{\overline{A}}} + (a_3 - a_2) + (a_4 - a_1), \quad \text{if } a_4 - a_1 \neq 0$$
$$= w_{\overline{A}}, \quad \text{if } a_4 - a_1 = 0$$

Binary variables:

Jaccard's Coefficient:

Formula: $S_{ij} = \frac{p}{p+q+r}$

p: number of variables that are positive(i.e., 1) for both the objects i and j

q: number of variables that are positive (1) for the i^{th} object, and negative (0) for the j^{th} object

r: number of variables that are negative (0) for the i^{th} object, and positive (1) for the j^{th} object

s: number of variables that are negative (0) for both objects i and j

7.3. Heterogeneous Extreme Learning Machine

The function is constructed relevant for the heterogeneous extreme learning machine (ELM) . The function can be derived from the general mappings like

f: $\widehat{H}^n \rightarrow \mathbf{y}$

Where y is an abstract set.

Consider the mappings , h: $\widehat{H}^n \times \widehat{H}^n \rightarrow \mathbf{y}$. Likewise, if x,w $\in \widehat{H}^n$ and y $\in \mathbf{y}$, then y=h (x, w) . Now if w is a fixed parametrization element, h will comply to the function f. If $\mathscr{W} \subseteq \widehat{H}^n$ is a finite non empty subset (1,2,.. p i.e, cardinality of p), with the elements $\{w_1,w_2,...w_p\}$, it may be used to parameterize a corresponding array of h-mappings $\{h_1, ... h_n\}$. This kind of parameterized mappings are called heterogeneous neurons (h- neurons)

Heterogeneous neuron model:

A class of general neuron models that accept heterogeneous inputs (possible missing as well), from which different instances or specific models or heterogeneous neurons may be derived. Such derived or specific neuronal models can be used to conform a variety of architectures or networks, and in particular a hybrid feed-forward network of heterogeneous and classical neurons is studied

Consider, for example, a structural representation for h given by the composition of two mappings, that is, h = f o s, such that s represents the similarity function and f represents the activation function. The mapping h can be regarded as an n- array function parameterized by an n- array tuple w $\in \widehat{H}^n$ representing the weight of the neuron i.e, h(x, w) = f (s(x,w)). Here, we consider 'f' to be a sigmoidal activation function, $f(x)=1/(1 + e^{-x})$ where x $\in$ R. So h = f(s) is

the neuron response, where s is the x and w scalar product, and f is the activation function. A general two-mappings submodel is shown in Fig 7.1.

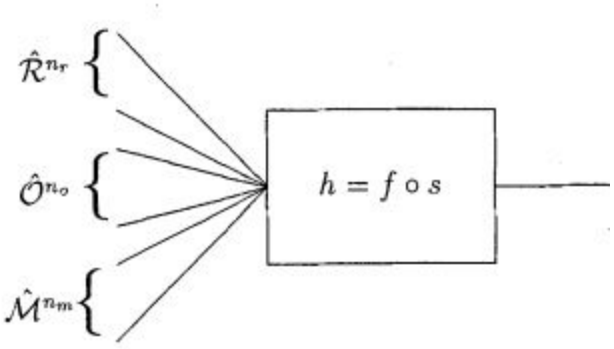

Fig 7.2.

The submodel of two mappings, with its components of similarity and activation.

Now, limiting the analysis to the case of two-mapping neuron models, a neuron that is very close to the classic, but more general than that, is one that accepts heterogeneous input. To obtain such a case, a similarity coefficient accepting as arguments tuples of the desired type will be sufficient as mapping s, and a possible choice is the general similarity coefficient of Gower, well known in the literature on multivariate data analysis.

For any two objects i , j described in terms of k variables, the similarity coefficient has in the values of [0,1] and is given by the expression,G_{ij}

$$G_{ij} = \frac{\sum_{k=1}^{n} g_{ijk}\delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}}$$

g_{ijk}: is a similarity score for objects i, j by variable value k. Their values are in intervals [0 , 1] and are calculated for numerical and qualitative variables according to various schemes.

δ_{ijk}: is a binary function which expresses whether or not both objects are comparable by their values w.r.t, variable k. It is 1 if and only if both objects for variable k have values that are different from X, and otherwise 0.

Activation function for Heterogeneous Extreme Learning Machine

As an activation function, a modified version of the classical sigmoid should be used, as its domain will now be that of the coefficient of similarity ([0, 1]), and not all reals as in the

classical case. Once again, there are many possible options for increasing nonlinear monotonous and symmetrical one-one mapping of [0 , 1] on [0 , 1].

A suitable family might be:

$$g(x,k) = \frac{-k}{(x-0.5) - a(k)} - a(k) \quad \text{if } x \leq 0.5$$

$$\frac{-k}{(x-0.5) + a(k)} + a(k) \quad \text{otherwise}$$

Where a(k) is an auxiliary function given by

$$a(k) = \frac{-0.5 + \sqrt{0.5^2 + 4 * k}}{2}$$

And k is a real-valued parameter that controls the curvature.

The behavior of this function family is described in the figure below,

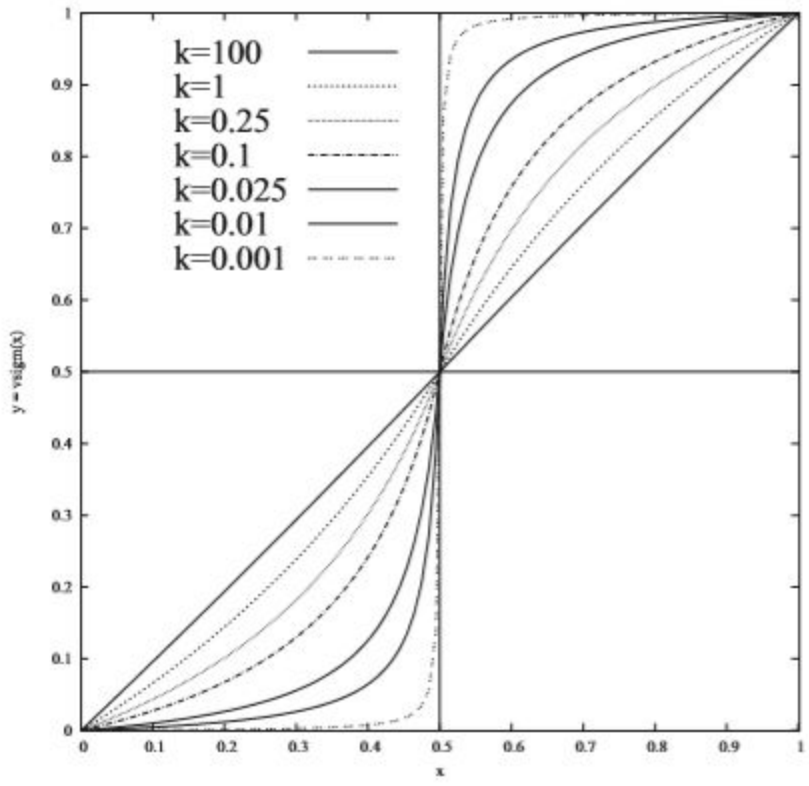

Fig 7.3. Sigmoids for different values of k

Remember that they are bounded non-constant continuous functions and the criteria of universal approximation theorems for ELMs are met when used as an activation function.

The activation functions that can also be used are Hard-limit function, Gaussian function and Multiquadric function which are also non-linear piecewise continuous functions.

The pseudocode of the HT-ELM algorithm taken from [1] is shown in Table 8.1.

HT-ELM based classification/regression (data with heterogeneous types).

Data: A training set $\mathbf{T}_{tr} = \{(\mathbf{x}_j, \mathbf{t}_j)\}_{j=1}^{N}$, $\mathbf{x}_j \in \hat{\mathcal{H}}^n$ (inputs),
$\mathbf{T} = \{\mathbf{t}_j\}, \mathbf{t}_j \in \mathbb{R}^m$ (targets),
a set of data types $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$,
a set of similarity functions $\mathcal{S} = \{\hat{s}_1, \ldots, \hat{s}_n\}$ for each type,
an aggregator operator Θ,
a bounded, continuous activation function $g(x), x \in \mathbb{R}$,
a set of continuous probability distributions $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$,
a number of hidden nodes (L),
a testing set $\mathbf{T}_{ts} = \{\mathbf{x}_j^{ts}\}_{j=1}^{N_{ts}}$.

Result: A trained ELM network. A processed testing set ($\mathbf{y}_j$: regression outputs, $\mathbf{c}_j$: classification outputs).

begin
  — *Training* —
  Randomly generate hidden node weights $(\mathbf{w}_i, b_i), i \in [1, L]$ using $\mathcal{P}$ so that for each $k \in [1, n]$, the elements $\mathbf{w}_{ik}$ will be of type τ_k (the same as for $\mathbf{x}_{jk}$).
  Calculate $\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_L, \mathbf{x}_1, \cdots, \mathbf{x}_N, b_1, \cdots, b_L)$ using $\mathbf{T}_{tr}, \mathcal{S}, \Theta$ and g(x).
  Determine the output weight matrix $\hat{\beta} = \mathbf{H}^{+}\mathbf{T}$
  — *Processing the testing set* —
  for $j \in [1, N_{ts}]$ **do**
    Calculate the actual output $\mathbf{y}_j$ of the ELM with $\mathbf{x}_j^{ts}$ as input
    if *classification* **then**
      Label $\mathbf{c}_j = \underset{d \in \{1,\ldots,m\}}{\arg\max} (\mathbf{y}_j)$.
    end
  end
end

Table 8.1.

8. Experimental Results and Comparisons

8.1. Synthetic Heterogeneous Dataset

The dataset used here is a synthetic heterogeneous dataset generated by, developing a program that is able to create a dataset of N observations described by:

- R: Real
- O: Ordinal
- B: Binary
- F: Fuzzy
- N: Nominal

and which can also be extended for the further data types.

Associated to every data type there is a way of generating values by sampling i.i.d. From a probability distribution.

i.i.d :

A good sample is characterized in statistics by being random, and therefore representative of the population at hand. The property of independence is at the basis of the principle of randomness. This means that whatever the i^{th} outcome of the sampling process is, there will be no dependence on the outcome of the next (i^{th} + 1) result. In other words, all results of a random sampling process are distinct from one another. The results are seen as realizations of a random variable, and thus the outcomes of every experiment have the same underlying distribution of probabilities. Such two random sample values are often called "i.i.d."(independent and identically distributed).

The sampling is done as follows:

- R $X \sim N(m,s^2)$

- Θ X ~ Mult(p_1,p_2,...p_n)
- b X ~ Bin(p)
- N X ~ Mult(p_1,p_2,...p_n)
- *f*1, *f*2

Here the fuzzy numbers can either be considered as triangular fuzzy numbers(*f*1) or the trapezoidal fuzzy numbers(*f*2)

Fuzzy numbers *f*1:

l = m - (10% of m)

r = m+ (10% of m)

Where m ϵ R , *l* ~ U(*l*, m) and *r* ~ U(m, *r*)

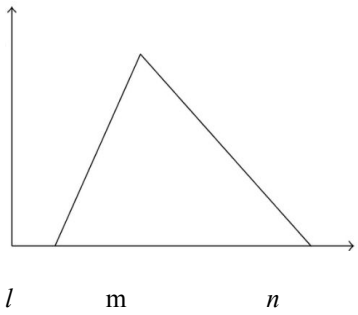

Fig 8.1. Trapezoidal Fuzzy numbers (TFN)

Fuzzy numbers *f*2:

l = m1 - (10% of m1)

r = m2+ (10% of m2)

Where m ϵ R , *l* ~ U(*l*, m1) and *r* ~ U(m2, *r*)

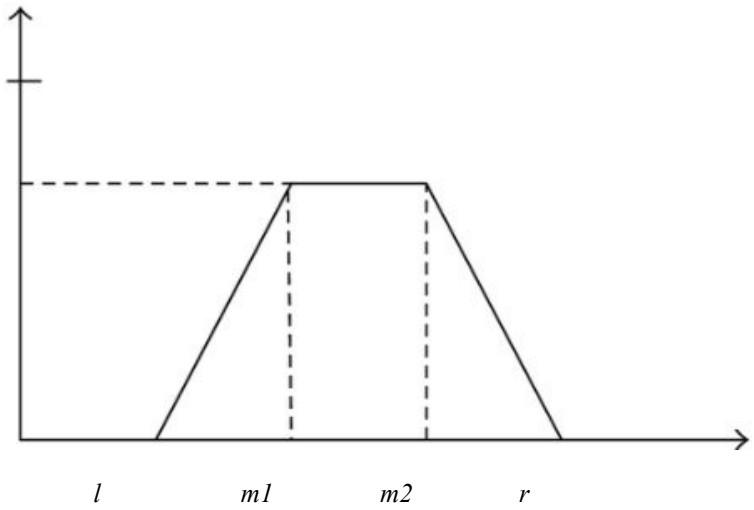

Fig 8.2. Generalized Trapezoidal Fuzzy Numbers

Once the sampling from probability distribution is done for N observations, to handle missing values injection of the missing values is to be done.

The injection of the missing values can be done by setting a parameter 'p' which represents the percentage of data to be replaced by empty values in the data generated.

Once the data generation along with the injecting the missing values is done, various datasets are generated in the same procedure for both regression and classification and then the similarity matrix is calculated.

The target value to be predicted by the similarity neural network is added at the last.

As the main idea is to handle heterogeneity along with the missing values and vagueness the data preprocessing step is not mandatory.

The heterogeneous data that is generated contains data types of real, ordinal, nominal, binary and fuzzy represented by R, O, N, B and F respectively.

The dataset consists of 8 columns with column names R1, R2, R3, O1, C1, B1, B2, B3 containing 1000 observations resulting in the dimension of the data to be 1000 rows and 8 columns. Now after injecting the missing values with the percentage of missing cells to be 20, 814 missing values are recorded.

	R1	R2	R3	O1	C1	B1	B2	B3
2	2.175208	6.304215	4.179417	3		0	0	
3	3.322638	5.277077	-0.79009	3	4	0	1	0
4	-0.5377	1.551439	4.131787	2	2	1	1	0
5	4.900162	7.117239	1.558613	3	2	0	1	1
6	3.006134	1.303563	9.375853	2	3	1	1	0
7	5.713347	7.289807	7.458088	1	4	0	0	1
8	3.622029	3.310419	4.599669	5	3	0	1	0
9	5.516561			3	3	1	0	1
10	-1.63798	-1.2213	-0.01137	3	2	0	1	1
11	4.431662	3.793674	5.887781	4	1	0	0	1

Fig 8.3. Synthetic Heterogeneous dataset generated.

8.2. Building similarity neural network

Once the dataset is generated the similarity matrix using gower's similarity metric is calculated. Pairwise gower's distance pwgower() a function is programmed that takes account of different kinds of similarity metrics considered theoretically for different kinds of data types. Using the similarity matrix generated the data is split into training and testing data to build the model and the model is then trained with the extreme learning machines method. HT-ELM networks were trained, with multiple hidden layer neurons, and also by changing the value of v-sigmoid's k in the range [0.1,5].

For classification different machine learning algorithms additional to ELM can be used like random forest, Leader and many other models,where a 10-fold cross validation is performed to predict the error rate.And also the performance comparison between ELM, SVM and other classification algorithms can also be analysed.

For Regression, in addition to ELM modeling, other models like SVM, M5- model tree can also be used. In all the comparison cases, 10- folds cross- validation model was used to predict the error rate.

9. Conclusions

Extracting knowledge from large, heterogeneous, and complex data is one of the main objectives. Not all the data- mining and machine learning methods can handle this kind of data without segregation and exclusions. ELM tends to provide positive computational results due to their simplicity, performance and speed. ELM extended to handle heterogeneous, imprecise and missing data also works well in the complex situations and provides a good level of accuracy along with the simplicity and the speed.

More research is necessary , exploring the other types of heterogeneities and also to handle large amounts of data.

References

[1] Julio J. Valdes, **"Extreme learning machines with heterogeneous data types"** National Research Council Canada, Information and Communications Technologies, Data Science for Complex Systems Group M50, 1200 Montreal Rd., Ottawa, Ontario K1A 0R6, Canada

[2] L. Belanche, **"Effective Learning with Heterogeneous Neural Networks"** Dept. de Llenguatges I Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain.

[3] Guang- Bin Huang, **"Extreme Learning Machine for Regression and Multiclass Classification"** Senior Member, IEEE, Hongming Zhou, Xiaojian Ding, and Rui Zhang

[4] Dekang Lin, **"An Information-Theoretic Definition of Similarity"** Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada R3T 2N2

[5] X. Zuo, L. Wang, Y. Yue, "**A new similarity measure of generalized trapezoidal fuzzy numbers and its application on rotor fault diagnosis",** Math. Probl. Eng. 2013 (2013) 10.

[6] L. Belanche, in **"Heterogeneous neural networks: theory and applications**", Polytechnic University of Catalonia, Barcelona, Spain, 2000. Ph.D. thesis.

[7] **Modeling Heterogeneous Data Sets with Neural Networks** ,Lluís A. Belanche Muñoz Dept. de Llenguatge i Sistemes Informàtics Universitat Politècnica de Catalunya Barcelona, Spain.

[8] **Extreme learning machine: Theory and applications**,Guang-Bin Huang, Qin-Yu Zhu, Chee-Kheong Siew School of Electrical and Electronic Engineering, NanyangTechnological University, Nanyang Avenue, Singapore 639798, Singapore

[9] "Salaries in Barcelona, Spain". Available: https://teleport.org/cities/barcelona/salaries/.[Accessed: 07-03-2020]

[10] **"A model of heterogeneous neurons and its use in configuring neural networks for classification",** Julio J. Vald~s] and Ricardo Garcfa Dept. of Languages and Informatic Systems. Polytechnical University of Catalonia, Jordi Girona Salgado, 1-3, 08034 Barcelona, Spain.

[11]https://medium.com/datadriveninvestor/extreme-learning-machine-for-simple-classification-e776ad797a3c