Universidade de Pernambuco
Escola Politécnica de Pernambuco
Programa de Pós-Graduação em Engenharia da Computação

Carlos Henrique Maciel Sobral Timóteo

A Methodology for Quantitative Risk Analysis in Project Management Software

Master Thesis

Recife, June 2014

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Programa de Pós-Graduação em Engenharia da Computação

Carlos Henrique Maciel Sobral Timóteo

A Methodology for Quantitative Risk Analysis in Project Management Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia da Computação da Universidade de Pernambuco como requisito parcial para obtenção do título de Mestre em Engenharia da Computação.

Prof. Dr. Sérgio Murilo Maciel Fernandes
Orientador

Prof. Dr. Mêuser Jorge Valença
Coorientador

Recife, June de 2014

REVIEWER NOTES.
Recife, Mar 2014

Jorge sentou praça, na cavalaria
E eu estou feliz, porque eu também sou da sua companhia
Eu estou vestido com as roupas e as armas de Jorge
Para que meus inimigos tenham mãos e não me toquem
Para que meus inimigos tenham pés e não me alcancem
Para que meus inimigos tenham olhos e não me veja
E nem mesmo um pensamento, eles possam ter para me fazerem mal
Armas de fogo, meu corpo não alcançarão.
Facas e lanças se quebrem sem o meu corpo tocar
Cordas e correntes se arrebentem sem o meu corpo amarrar
Pois eu estou vestido com as roupas e as armas de Jorge
Jorge é da Capadócia, salve Jorge!
Perseverança, ganhou do sórdido fingimento
E disso tudo nasceu o amor.

Jorge da Capadócia
Jorge Ben Jor

What is, is, everything happen for a reason, when life kicks you, get up and rip its heart out, never stop fighting.

Roger W. Kivell
WWII Navy Frogman (Navy Seal)

Sumário

Capítulo 1

Introduction

How risky are software projects? Several studies about effectiveness of software cost, scope, schedule estimation techniques; surveys from software professionals in industry; and analysis of projects portfolio have been done to answer this question [1]. However, there is not a consensus.

Every project involves risk. There is always at least some level of uncertainty in a project's outcome, regardless of what the Gantt chart on the wall seems to imply. High-tech projects are particularly risky, for a number of reasons. First, technical projects are high varied. These projects have unique aspects and objectives that significantly differ from previous work, and the environment for technical projects evolve quickly. In addition, technical projects are frequently "lean", challenged to work with inadequate funding, staff and equipment. To make matters worse, there is a pervasive expectation that however fast the last project may have been, the next one should be even quicker [2].

Projects that succeed generally do so because their leaders do two things well. First, they recognize that a few of the work on any project, even a high-tech project, is not new. For this work, the notes, records, and lessons learned on earlier projects can be a road map for identifying, and in many cases avoiding, many potential problems. Second, they plan project work thoroughly, especially the portions that require innovation, to understand the challenges ahead and to anticipate many of the risks [2].

Some benefits of good risk management of software projects are:

1. reduction of costs associated with changes in software;
2. development of a response plan to unexpected events, i.e., a contingency risk plan;
3. prediction of likelihood of undesired events;
4. tracking the baselines of cost, schedule and quality.

Such factors may determine the success of projects [3] [4].

1.1 Motivation

In 2009, CHAOS Report [5] showed that 32% of projects achieved success - were delivered on time, on budget and with the promised requirements -; 44% of the projects were challenged - or schedule or budget or requirements were not fulfilled; not least, 24% of projects failed and were canceled. That is due to the risks involved in project activities and to a absent or defective software risk management [6].

Schmidt et al. [7] have noticed that many software development projects end in failure. They showed that around 25% of all software projects are canceled outright and as many as 80% of all software projects run over their budget, exceeding it by 50% in average. Paul Bannerman [8] states that industry surveys suggest that only a quarter of software projects succeed outright, and billions of dollars are lost annually through project failures or projects that do not deliver promised benefits. Moreover, the author shows evidences that it's a global issue, impacting private and public sector organizations [9].

Predict possible events in short, medium and long term is often failure. By analyzing risks and uncertainties, project managers commonly rely on intuition rather than logic and analysis. However, intuitive thinking is often subject to delusions, causing predictable mental errors and eventually poor decisions. A way to balance the effect of these psychological illusions is a systematic risk analysis and efforts to mitigate them through analytical methods.

It is difficult to manage something that can not be quantified. Project managers should quantify the probability of risk, the impact, and their cumulative effect on a project. Furthermore, it is important to evaluate the various mitigation options: the cost for each option and the required time to perform the mitigation [10].

Interpreting the concept of risk is a hard task, especially regarding the application of this knowledge in the development and use of efficient procedures for risk analysis in software project management techniques. Managing risk and uncertainty in software projects is critical to project management discipline. However, in times of economic crisis becomes much more difficult to perform risk management, due to the costs incurred.

Since it is an area of research that is growing, new and better methodologies to identify, measure and control risk items of software need to be developed. Keshlaf and Riddle [11] conclude that even if there are many approaches there is still a large gap regarding what is practiced by software industries.

1.2 Problem Description

Even though risk management in software project management is a healthy process, its adoption is still far from expectations. A few causes are the overloading of responsibilities on project managers, the low importance attributed to the area, the lack of knowledge of risk management, the costs incurred in risk management activities, the lack of technical skill and familiarity with specific tools. Consequently, the project is prone to the negative influence of risks without a contingency plan, which may lead to

project failure. Kwak and Ibbs [12] identified risk management as the least practiced discipline among different project management knowledge areas. The authors mention that, probably, a cause for it is that software developers and project managers perceive managing uncertainty processes and activities as extra work and expense. According to the benchmarking conducted in 2009 by the Project Management Institute, in 20% of the projects their managers do not perform all the planning processes and in only 35% of the projects, risk management is conducted according to a formal methodology, structured by policies, procedures and forms. Also, 46% of managers carry out management activities part a time.

Barry Boehm [13] defined risk as the possibility of loss or injury. That definition can be expressed by risk exposure formula. Even Boehm cites risk exposure as the most effective technique for risk priorization after risk analysis, Paul Bannerman [8] considers this definition limited and unsuitable. In classical decision theory, risk was viewed as reflecting variation in the probability distribution of possible outcomes, whether negative or positive, associated with a particular decision. This study takes into account Project Management Institute [4] definition whereupon project risk is a certain event or condition that, if it occurs, has a positive or negative effect on one or more project objectives. A complementary definition proposed by Haimes [14] is also considered, which express risk as a measure of the probability and severity of adverse effects.

A risk factor is a variable associated with the occurrence of an unexpected event. Risk factors are correlational, not necessarily causal and, if one of them occurs, it may have one or more impacts. According to Haimes [14], risks can often arise as the result of an underlying stochastic process occurring over time and space, but also, can occur based on deterministic risk factors. Risk estimation can be achieved based on historical information and knowledge from previous similar projects and from other information sources [4].

Risk is a concept that many find difficult to be understood because it involves two complex metrics: probability and severity of adverse effects [15]. A limitation in this definition is the practical difficulty of estimating the probability and impact of various risk factors, especially in software projects. Probabilities can only be defined with significance for activities that are repeated many times under controlled conditions. However, the unique nature of many software projects activities do not allow accurate estimation of their probabilities. Other limitation of that definition is that it only encompasses known or foreseeable threats, providing limited options to manage unnoticed threats, but also does not recognize unforeseeable threats. That is a consequence of the definition of risk in terms of probability and impact; since, to assess likelihood and impact is necessary to be able to predict an eventuality. In addition, there is another issue: the best decisions are based on numerical quantification - determined by patterns from the past - or on subjective evaluation of the uncertainties? It is not possible to quantify the future with certainty, but through likelihood, it is possible to predict it based on the past. Although it is difficult to find a standard software project, it is possible to classify activities and set patterns that enable the estimation. To Paul Bannerman [8], the usual solution to that problem in software projects is to observe the risk in a broad way, in terms of uncertainty, and evaluate it qualitatively.

Haimes [15] considers two premises in research of risk analysis, which will also

be considered during this study. First is that risk is usually quantified by mathematical formula of expectation. However, even though that formula enables a valuable measure of risk, it fails to recognize and or exacerbate the consequences of extreme events. Tom Kendrick presents in his book [16] a framework to identify and manage disasters. Second, states that one of the most difficult tasks in systems analysis is to know how to model it. Therefore, new proposals for quantitative analysis and modeling of systems taking into account its risks will contribute to the scientific advances in the field.

The need to manage risks (undesired events) increases exponentially with system complexity. Managing those events in such complex systems becomes difficult to identify and predict undesirable expected or unexpected events occurrence because of huge amount of risk factors involved and their relations. There is an increasing need for more systematic methods and tools to supplement individual knowledge, judgment and experience. These human traits are often sufficient to address less complex and isolated risks. For example, a portion of the most serious issues encountered in system acquisition are the result of risks that are ignored, due to its low likelihood, until they have already created serious consequences [17].

The Guide to the Project Management Body of Knowledgment [4] presents Monte Carlo Simulation as a good practice method to project risk analysis. However, there are some limitations in the adoption of this approach that makes it unfeasible [18]. Simulations can lead to misleading result if inappropriate inputs, derived from subjective parametrization, are entered into the model. Commonly, the user should be prepared to make the necessary adjustments if the results that are generated seem out of line. Moreover, Monte Carlo can not model risks correlations. That means the numbers coming out in each draw are random and in consequence, an outcome can vary from its lowest value, in one period, to the highest in the next. Therefore, alternative approaches must be considered to predict risk likelihood and impact, taking into account project risk characteristics and Monte Carlo Simulation limitations. Thus, risk analysis should be a more accurate and easier task, from users point of view. This work finds artificial neural networks as a valuable alternative to be considered in software project risk analysis.

1.3 Objectives

1.3.1 Research Questions

How to analyze risks in software project management considering disasters?
How quantitatively analyze risks in software project management?
Which data of risk registers of software project are available to perform the study?
How to develop a method to predict risks of software project management in order to efficiently support decision making?

1.3.2 General Objective

The main purpose of this dissertation is to define a methodology to determine which is a more efficient approach to software project risk analysis: Monte Carlo Simulation (MCS) technique, Linear Regression Models (LRM's) or Artificial Neural Networks (ANN's) alternatives - Multilayer Perceptron (MLP), Support Vector Machine (SVM), Radial Basis Function (RBF) and Neuro Fuzzy System (NFS) - to improve accuracy and decrease the error prone risk impact estimation.

1.3.3 Specific Objectives

- Develop a method to predict risk impacts through adoption of artificial neural networks to manage risks in software projects;
- Evaluate traditional approaches to risk impact estimation in terms of prediction error;
- Evaluate several artificial neural networks to obtain a better configuration to the chosen risk dataset;
- Determine a satisfactory limiar error to risk impact estimation.

The first objective consists on determining a methodology to estimate risk impact aiming to reach high efficiency, through minimizing estimation error. The second one involves studying Monte Carlo Simulation (MCS) and PERT Analysis to compare with Multiple Linear Regression Model (MLRM) and Regression Tree Model (RTM) aiming to comprove if it is a good practice to utilize MCS and PERT. THe third object is reached after experimenting several artificial neural networks as Multilayer Perceptron (MLP), Support Vector Machine (SVM) e Radial Basis Function (RBF). Moreover a Neuro-fuzzy System (NFS) was also considered on this study. Finally, a survey is performed with Project Management Institute (PMI) Risk Community of Practice (Risk CoP) to determine a satisfactory limiar error to estimate risk impact.

In summary, the methodology adopted in this study is to make statistical experiments to evaluate the prediction error of risk impact from PERIL dataset [2], a framework to identify risks in software project management. The selected techniques will estimate the outcome of risk impacts. RMSE (Root Mean Square Error) will be calculated thirty times for each approach, and then a hypothesis test may be necessary to assert which is a more accurate method that fits the dataset. Lastly, the method to estimate risk impact is determined. More details are presented in Chapter 4.

It is concluded that a MLP variation called MLPReg, is the successful approach to estimate risk impacts in this study. Besides that, all artificial neural networks alternatives are better than , both, linear regression models, monte carlo simulation either PERT analysis. Therefore, it could not be found any reason to assign monte carlo simulation and PERT recommended methods to risk analysis according to statistical experiments conducted here.

The rest of the dissertation is organized in the following chapters: Chapter 2 address project risk management, qualitative and quantitative risk analysis concepts, monte carlo

simulation, linear regression models and artificial neural networks concepts and characteristics. Chapter 3 describes PERIL database, data preprocessing methods to prepare the database to the study and describes algorithms configuration. Chapter 4 describes each experiment. Chapter 5 presents the obtained results for each experiment. Finally, Chapter 6 presents the conclusions and suggestions of future works.

The works described in this dissertation had results published in the following papers:

- C. H. M. S. Timoteo, M. J. S. Valença, S. M. M. Fernandes, "*Evaluating Artificial Neural Networks and Traditional Approaches for Risk Analysis in Software Project Management - A case study with PERIL dataset*", ICEIS 2014: *16th International Conference on Enterprise Information Systems*, Abril, 2014.

Capítulo 2

Literature Revision

2.1 Related Work

After a systematic literature revision, numerous approaches were explored in risk analysis, that includes Logistic Regression Model (LRM), Bayesian Belief Network (BBN), Artificial Neural Network (ANN), Discriminant Analysis (DA), Decision Tree (DT), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Fuzzy Set Theory (FST), Neuro-Fuzzy System (NFS), Extended Fuzzy Cognitive Maps (E-FCM) [19] [20] [21] [22] [23] [24] [25] [26].

Hu et al. [21] proposed a method to analyze software risks and predict outcomes of software projects. Genetic algorithm could be utilized as an optimization method to improve ANN in many ways, including weights, network structure and rule learning. The standard ANN was enhanced by introducing GA, in that study. Results of the experiments showed that after introducing GA to the ANN training process, the enhanced software risk evaluation model could be improved notably and achieved higher accuracy when compared to the SVM model. Zhang Dan [26] proposed an ANN prediction model that incorporates with Constructive Cost Model (COCOMO) which was improved by applying PSO, to provide a method which can estimate the software develop effort accurately. Attarzadeh e Ow [22] utilized ANN to improve accuracy of effort estimation compared to the traditional COCOMO model. Huang et al. [20] have presented a general framework for software estimation based on NFS, the authors improved cost estimation for COCOMO'81.

Yu [24] showed up a model based on the fuzzy theory. He overcame the difficulty of qualitative indicators and quantitative assessment in the traditional analysis methods. In addition, Saxena and Singh [25] explored neuro-fuzzy techniques to design a suitable model to utilize improved estimation of software effort for NASA software projects. Results showed that NFS has the lowest prediction error compared with existing models. Meanwhile, Lazzerini and Mkrtchyan [27] suggested a framework to analyze risks using E-FCMs and extended E-FCMs themselves by introducing a special graphical representation for risk analysis.

Mizuno et al. [19] have proposed a new prediction method for risky software projects. The authors have used the logistic regression model to predict whether a project

becomes risky or not. However, the proposed estimating approaches for the cost and the duration do not have absolutely high level of accuracy.

Dzega and Pietruszkiewicz [23] have presented results of risk analysis experiments performed using data mining classifiers such as C4.5, RandomTree and Classification and Regression Tree (CART) algorithms. Besides that, they described how boosting and bagging metaclassifiers were applied to improve the results and also analyzed influence of their parameters on generalization abilities in prediction accuracy. Due to a large number of unordered labeled attributes in datasets, MLP and SVM were rejected at early stages, producing low accuracy for each dataset.

In summary, some of those studies proposed methods to software project cost, schedule and effort estimation; another studies proposed approaches to risk classification and software project classification (success, challenged and failed). Moreover, the remainder presented techniques to risk impact estimation on software project management [24] [25] [27] [23].

2.2 Project Risk Management

According to the PMBOK [4], risk can be defined as an uncertain event or condition that, if it occurs has an effect on at least one of the project goals. It can be considered both a threat (negative impact) or opportunity (positive impact).

Risk management involves planning, identification, assessment and prioritization . In a better definition, risk management can be defined as the process of analyzing the risk exposure and determining how best deal with such exposure. One goal of this area is not only to identify, but also develop a robust approach to proactively manage the impact on the project [28].

According to PMBOK [4], Project Risk Management includes process as planning, identification, analysis, response planning, monitoring and controlling risks of a project. Its purpose is to increase likelihood and impact of positive events and reduce probability and severity of negative events. From management point of view, making informed decisions by consciously assessing what can go wrong, as well as its likelihood and severity of the impact, is at the heart of risk management. This activity involves the evaluation of the trade-offs associated with all policy options for risk mitigation in terms of their costs, benefits, risks and the evaluation of the impact of current decisions on future options.

A summary of project risk management processes are defined as follows. There are six topics for project risk management included in the planning and the monitoring and controlling groups.

- Planning risk management: the process of defining how conduct risk management activities in a project;
- Identifying risks: the process of determining risks that can affect project and documenting its characteristics;

- Performing qualitative risk analysis: the process of prioritizing risks to analyze or additional actions through assessment and combination of its occurrence probability and impact;
- Performing quantitative risk analysis: the process of analyzing numerically the effect of previous identified risks, in terms of general project objectives;
- Planning risk responses: the process of developing options and actions to increase opportunities and decrease threats to project objetives;
- Monitoring and controlling risks: the process of implementing risk responses planning, tracking identified risks, monitoring residual risks, identifying new risks and assessing the efficacy of risk treatment process during the whole project.

The Software Engineering Institute (SEI) has developed a risk management methodology named Software Risk Evaluation (SRE) that is specifically oriented to projects in the software industry. The SRE paradigm consists of six elements: five modules (identify, analyze, plan, track, control) and a central element (communication), which is the key to effective risk management [3] [29].

Boehm [13], Chapman [30], Fairley [31], Bandyopadhyay et al. [32] have presented different methods and models for risk management of software projects. However, there are common elements in all previous approaches: identification, assessment of the likelihood and impact, and responses planning to manage these risks if they occur.

Risk management in the large sense is useful to organizations, where many projects are undertaken. That management has main focus on aggregate risk for decision making such as interrupt, abort and continuing. Otherwise, in project leader perspective, there are only isolated projects, that should focus primarily on risk in the small sense [2].

Otherwise in the small sense, works to improve the chances for each individual project. In most other fields, risk management is primarily concerned with the mean values of large numbers of independent events. For project risk management, however, what generally matters most is predictability - managing the variation expected in the result for the specific project [2].

The goals of risk management for a single project are to establish a credible plan consistent with business objectives and then to minimize the range of possible outcomes. The larger range of possible durations for a project represents higher risk. Project risk increases with the level of uncertainty, both negative and positive [2].

Project risk management uses the two fundamental parameters of risk - likelihood and loss. Likelihood is generally the probability of a project event occur - though often by guessing, so it can be quite imprecise. Loss is generally referred to for projects as ”impact”, and it is based on the consequences to the project if the risk does occur. Impact is usually measured in time or cost, particularly for quantitative risk assessment [2].

The primary benefit of project risk management is either to develop a credible foundation for each project by showing that it is possible, or to demonstrate that the project is not feasible so it can be avoided, aborted, or transformed. Risk analysis can also reveal opportunities for improving projects that can result in increased project value.

Adequate risk analysis lowers both the overall cost and the frustration caused by avoidable problems. The amount of rework and unforeseen late project effort is reduced. Besides that, risk analysis uncovers weakness in a project plan and triggers changes, new activities, and resource shifts that improve the project [2].

2.2.1 Qualitative Risk Analysis

The process of qualitative analysis evaluates the characteristics of project risks identified individually and prioritizes them based on requirements established for the project. In other words, qualitative analysis assesses the probability of each event occurring and the individual effect of each of them to the project objectives. As that process does not directly address the overall risk to the project objectives, which result from the combined effect of individual events and their interactions with each other, then this can be achieved through the use of techniques of quantitative analysis [34].

2.2.2 Quantitative Risk Analysis

Analysis is the conversion of risk data into risk decision-making information. Analysis provides the basis for the project manager to work on the ”right”and most critical risks. Boehm [13] defines risk analysis objective as the assessment of the loss probability and loss magnitude for each identified risk item, and it assess compound risks in risk-item interactions. Typical techniques include performance and cost models, network analysis, statistical decision analysis and quality-factor (such as reliability, availability, and security) analysis.

Risk analysis depends on a good mechanism to identify risks. However, most of the methods assume that managers have the required experience to be aware of all pertinent risk factors, but it can not be the situation. Moreover, many of these methods can be time-consuming and thus too costly to use on a regular basis. Therefore, one popular method for identifying risk factors has been the use of checklists. Unfortunately, these checklists are based in small samples or, even worse, flawed in their risk historical data collection methods.

Most used techniques to risk analysis include [4]:

- Sensibility analysis: it helps to determine which risks have the higher potential impact on the project. A typical representation of the sensitivity analysis is the tornado diagram;

- Earned Monetary Value (EMV): it is a statistical concept that computes the average score when the future includes scenarios that may or may not occur (ie, under uncertainty analysis). A common use of this kind of technique is the decision tree analysis;

- Modeling and simulation: Simulation utilizes a model that converts the detailed specified uncertainties in their potential impact on project objectives. The general iterative simulations are performed using MCS;

- Specialized opinion: expert judgment (ideally by specialists with relevant and recent expertise) is required to identify the potential impacts on cost and schedule, to assess the likelihood, but also to define input variables and which tools to use.

The purpose of a quantitative risk analysis is to create a ”risk profile”of the project. For this purpose the following information are required: the chance of finishing the project within a certain period of time or budget; the success rate of projects; the worst, average and best estimates of duration and other project parameters [4].

Some studies propose new tools for quantitative risk assessment. Among them, Virine [10] presents the methodology Events Chain. In that paper, the activities of a project are not an uniform and continuous process, those tasks are affected by external events, that transform the activities from one event to another. The moment at which external events occur are probabilistic and can be set using a statistical distribution. Moreover, events can cause other events, thereby, creating the Event Chain. The analysis of those combinations is performed by Monte Carlo simulation.

Risk analysis demonstrates the uncertainty of project outcomes and is useful in justifying reserves for schedule and/or resources. It is more appropriate to define a window of time (or budget) instead of a single-point objective for risky projects. For example, the target schedule for a risky project might be twelve months, but the committed schedule, reflecting potential problems, may be set at fourteen months. Completion within (or before) this range defines a successful project; only if the project takes more than fourteen months will it be considered a failure [2].

2.3 Conventional Techniques for Risk Analysis

2.3.1 Monte Carlo Simulation

Monte Carlo simulation is a technique that computes or iterates the project cost or schedule many times using input values selected at random from probability distributions of possible costs or durations, to calculate a distribution of possible total project cost or completion dates [4].

A model is developed, and it contains certain input variables. These variables have different possible values, represented by a probability distribution function of the values for each variable. The Monte Carlo method is a detailed simulation approach through intensive computing to determine the likelihood of possible outcomes of a project goal; for example, the completion date or total cost. The inputs of the procedure are obtained randomly from specific intervals with probability distribution functions for the durations of schedule activities or items from cost baseline. Those different input values are used to construct a histogram of possible results to the project and its relative probability, but also the cumulative probability to calculate desired contingency reserves for time or cost. Additional results include the relative importance of each input in determining the overall project cost and schedule [35].

The invention of the method, especially the use of computers to do the calculations, was credited to Stanislaw Ulam, a mathematician working in the U.S. Manhattan Project during World War II. His work with Jon von Neumann and Nicholas Metropolis turned

statistical sampling from a mathematical curiosity to a formal methodology applicable to a wide range of problems [35].

Monte Carlo Simulation is a detailed simulation approach through intensive computation to determine the likelihood of possible outcomes of a project objective, for example, the completion date or the overall cost. The inputs to the procedure are obtained randomly from specific intervals with probability distribution functions for durations of schedule activities or items in costs baseline. Such distinct input values are used to construct a histogram of possible outcomes and relative probability in project, as well as the cumulative probability to calculate the desired contingency reserve of time or cost. Additional results include the relative importance of each input to determine the overall project cost or the project schedule. An example of prediction of schedule and cost risks are presented in Figure 2.1 [34].

In Figure 2.1, it is observed the forecast of a timeline for completion of a project. On the horizontal axis, there are the possible end dates of the timeline. How often those dates occur in sampling are represented by vertical bars. The cumulative frequency is represented by the dark curve and each related value is on the right edge of the chart. Based on the relative frequency, it is possible to predict the likelihood of the project be completed by a given date, int that example, the illustrated project has a 90% chance to finish until May 8, 2009.

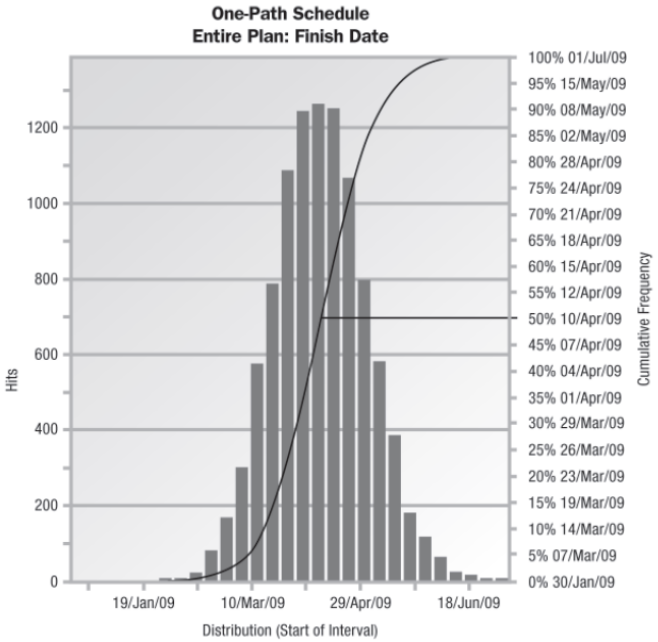

Figura 2.1: Exemplo de Resultado da Simulação Monte Carlo

2.3.2 PERT Analysis

PERT was originated by the U.S. Navy in 1958 as a tool for scheduling the development of a complete weapons system. The technique considers a project to be an acyclic network of events and activities. The duration of a project is determined by a system

flow plan in which the duration of each task has an expected value and a variance. The critical path includes a sequence of activities that cannot be delayed without jeopardy to the entire project. PERT can be used to estimate the probability of completing either a project or individual activities by any specified time. It is also possible to determine the time duration corresponding to a given probability [36].

The first step in applying PERT is to diagram the project network, in which each arc represents an activity and each node symbolizes an event (such as the beginning or completion of a task). Alternatively, each node can symbolize an activity. The second step is to designate three time estimates for each task: optimistic (a), pessimistic (b) and most likely (m). Small probabilities are associated with a and b. In the original PERT, a is the minimum duration of an activity; the probability of a shorter duration is zero. Similarly, b is the maximum duration; the probability that the duration will be less than or equal to b is 100%. No assumption is made about the position of m relative to a and b. In statistical terms, a and b are the extreme ends of a hypothetical distribution of duration times. The mode of the distribution is m. To accommodate flexibility in the positions of these parameters, the beta distribution is used. The beta distribution is useful for describing empirical data and can be either symmetric or skew [36].

The third step is to compute the expected value and variance of the duration of each activity in the project network. The mean of a beta distribution is a cubic equation. The PERT equation for the mean, Equation 2.1, is a linear approximation to this:

$$\tau_e = (\alpha + 4m + \beta)/6 \tag{2.1}$$

where $\tau_e =$ is the expected duration of an activity, m is equal to the mode, which occurs when a and b are symmetrical about m.

In unimodal probability distributions, the standard deviation of the distribution is equal to approximately one-sixth of the range. With 100% of the possible durations bound by a and b, the estimated variance of the duration is as follows:

$$\sigma_1^2 00(\tau_e) = [(b-a)/6]^2 \tag{2.2}$$

where σ^2 = variance of the activity duration. An alternative, presented in [36] is to define a and b as the 5% and 95% thresholds of the range, respectively. Then, the variance is as follows:

$$\sigma_9^2 0(\tau_e) = [(b-a)/3.2]^2 \tag{2.3}$$

The fourth step is to order the activities sequentially, from the beginning to the end of the project, in a tabular format, listing the optimistic, pessimistic, most likely, and expected durations and the variances. Fifth, forward and backward passes through the network are performed to identify the critical path, just as in the widely used critical path method. The central limit theorem is then applied as follows: The distribution of the sum of the expected durations of the activities along the critical path is approximately normal, particularly as the number of activities increases. The expected duration of each sum is equal to the sum of the expected durations. Similarly, the variance of each sum is the sum of the variances [36].

These applications of the central limit theorem enable the computation of project duration probabilities using the deviations from a zero mean of the standard normal variable (Z). These probabilities can be critical in making financial decisions about the viability of a project.

2.4 Statistical and Intelligent Computing Techniques for Risk Analysis

In this section we will initially explore two different predictive models that could be applied to risk analysis domain: multiple linear regression and regression tree models. Our choice was was not a consequence of some formal model selection step, regarding these models. Otherwise, the models still are two good alternatives for regression problems as they are quite different in terms of their assumptions regarding the "shape"of the regression function being approximated and they are easy to interpret and fast to run on any computer. The purpose of choosing these two common regression models is to validate the performance of others techniques. For example, if a technique has worse perfformance than multiple linear regression either regression tree models then it seems a bad alternative.

2.4.1 Multiple Linear Regression

Multiple Linear Regression is among the most used statistical data analysis techniques. These models obtain an additive function relating a target variable to a set of predictor variables. This additive function is a sum of elements of the formula $\beta_i \times X_i$, where X_i is a predictor variable matrix and β_i is the weight matrix in the multiple linear regression [37].

In linear regression, the data are modeled to fit a straight line. For example, a random variable Y, called a response variable, can be modeled as a linear function of another random variable X, called a predictor variable with the Equation 2.4:

$$Y = \alpha + \beta X, \tag{2.4}$$

where the variance of Y is assumed to be constant. The coefficients α and β (called regression coefficients) specify the Y intercept and slope of the line, respectively. These coefficients can be solved for by the method of least squares, which minimizes the error between the actual line separating the data and the estimate of the line. Given s samples or data points of the form $(x_1, y_1), (x_2, y_2), ..., (x_s, y_s)$, then the regression coefficients can be estimated using this method with Equations 2.5 and 2.6:

$$\beta = \frac{\sum_{i=1}^{s} (x_i - x)(y_i - y)}{\sum_{i=1}^{s} (x_i - x)^2}, \tag{2.5}$$

$$\alpha = y - \beta x, \tag{2.6}$$

where x is the average of $x_1, x_2, ..., x_s$, and y is the average of $y_1, y_2, ..., y_s$. The coefficients α and β often provide good approximations to otherwise complicated regression equations. Multiple regression is an extension of linear regression allowing a response variable Y to be modeled as a linear function of a multidimensional feature vector [38].

2.4.2 Regression Tree Model

A regression tree is a hierarchy of logical tests on some of the explanatory variables; thus, not all variables need to appear in the tree. A tree is read from the root node to leaf nodes. Each node of a tree has two branches. These are related to the outcome of a test on one of the predictor variables. The testing goes on until a leaf node is reached. At these leaves we have the predictions of the tree. This means that if it is necessary to use a tree to obtain a prediction for a particular sample, it is needed to follow a branch from the root node until a leaf, according to the outcome of the tests for the sample. The average target variable value found at the leaf reached is the prediction of the tree [37].

Figure 2.2 presents the regression tree model to PERIL dataset. From the root node it is possible to verify two branches, these branches shows logical tests under *Date0* variable. The following nodes represents possible outcomes taking into account only *Date0*. The second level shows logical tests under *Category3* and *Subcat2* variables. Next, it is found three leaf nodes. Each leaf node values represents the possible outcome and the number of samples in PERIL dataset, satisfying the conjugated logical tests in the previous tree levels. Lastly, logical tests under *Subcat0* variable is presented to generate two another leaf nodes.

Trees are usually obtained in two steps. Initially, a large tree is grown, and then this tree is pruned by deleting bottom nodes through a process of statistical estimation. This process has the goal of avoiding overfitting. This has to do with the fact that an overly large tree will fit the training data almost perfectly, but will be capturing spurious relationships of the given dataset (overfitting it), and thus will perform badly when faced with a new data sample for which predictions are required. The overfitting problem occurs in many modeling techniques, particularly when the assumptions regarding the function to approximate are more relaxed. These models, although having a wider application range (due to these relaxed criteria), suffer from this overfitting problem, thus requiring a posterior, statistically based estimation step to preclude this effect [37].

2.5 Artificial Neural Networks

An Artificial Neural Network (ANN) is a massively parallel distributed processor made up of simple processing units, which has a natural propensity to load experimental knowledge and become it available for use [40]. It adopts non-parametric regression estimates made up of a number of interconnected processing elements between input and output data. They have excellent learning and generalizing capabilities.

Another definition describes ANN as a system made up of interconnected processing elements, called neurons, which are presented in layers (one input layer, one or several

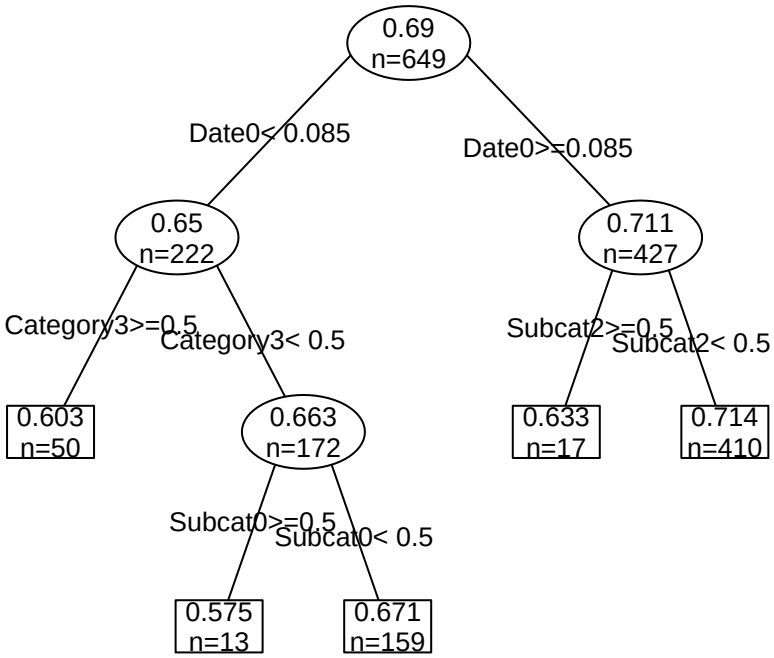

Figura 2.2: Regression Tree Model for PERIL

intermediate layers and one output layer) responsible for nonlinearity and memory in the network [41].

Artificial Neural Networks (ANN) are models that attempt to simulate the behavior and functioning of the human brain. Just as there are biological neurons, essential components to information processing in the brain, the RNA is composed by units which aim to perform the same functions of biological neuron. These components are called artificial neurons and were proposed in 1943 by Mc-Culloch and Pitts [42].

Following McCulloch and Pitts work arises the learning rule proposed by Donald Hebb [43] which is the basis of all learning rules. In his famous book, the author tried to find a neural mechanism to explain how the information can be stored and retrieved in neurons. The learning rule was stated as follows: ”When a neuron receives a stimulus from another neuron, and if both are highly active, the weight between them should be strengthened, otherwise weakened”. That neuron is called Perceptron. In 1960, Widrow

e Hoff [44] presented a learning rule for Perceptron extension, previously developed by Frank Rosenblatt [45], called ADALINE (*Adaptive Linear Neuron*). This rule based on least square method was known as delta rule. Paul Werbos [46], in 1974, developed backpropagation algorithm, this algorithm was subsequently popularized through Rumelhart and McClelland publication [47], in 1985.

McCulloch and Pitts neuron model tries to represent biologic neuron by adoption of a propagation rule and an activation function. Consider $x_1, x_2, x_3, ..., x_n$, as input variables x_j, in which $j = 1, ..., n$, of input neuron i. The net input net_i is given by propagation rule:

$$net_i = \sum_{j=1}^{n} w_{ij} x_j - \theta \tag{2.7}$$

where, w_{ij} are synaptic weights and θ is the activation threshold of the neuron.

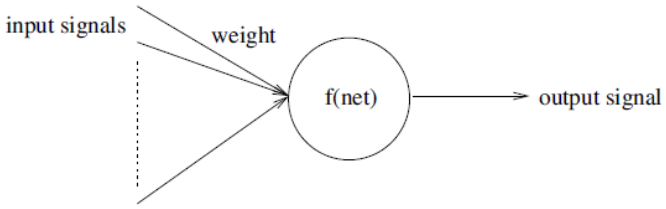

Figura 2.3: Representação gráfica da MLP com três camadas

The output y_i is given by $f(net_i)$, in which f is the activation function. There are many proposed activation functions, between them the most usual are: linear, step, gradient, logistic sigmoid, hiperbolic tangent and gaussian represented respectively by [41] [48]:

$$f(net_i) = net_i \tag{2.8}$$

$$f(net_i) = \begin{cases} \gamma_1 & \text{if } net_i \geq \theta \\ \gamma_2 & \text{if } net_i < \theta \end{cases} \tag{2.9}$$

$$f(net_i) = \begin{cases} \gamma & \text{if } net_i \geq \varepsilon \\ net_i & \text{if } -\varepsilon < net_i < \varepsilon \\ -\gamma & \text{if } net_i \leq -\varepsilon \end{cases} \tag{2.10}$$

$$f(net_i) = \frac{1}{1 + e^{-net_i}} \tag{2.11}$$

$$f(net_i) = \frac{e^{net_i} - e^{-net_i}}{e^{net_i} + e^{-net_i}} \tag{2.12}$$

$$f(net_i) = e^{-(net_i - \theta)^2 / \sigma^2} \tag{2.13}$$

2.5.1 MultiLayer Perceptron

The multilayer perceptron network is a generalization of the simple perceptron network by adding at least one intermediate layer, known as the hidden layer. In a layered network, neurons are arranged in each layer. In MLP, the first of them is the input layer, in which input variables are directly connected to a exclusive neuron. The next is the hidden layer that completely connects the neurons from previous layer to the neurons in output layer. Lastly, output layer represents ANN outcome. Each input in a neuron has an associated weight to be adjusted by training algorithm. A common MLP model contains a bias neuron. The MLP is a direct graph, in which inputs data are propagated from input layer to hidden layer*s* and from hidden layer to the output layer. The data flow in forward path in a MLP is known as ”forward phase”. The data flow in opposite way is the ”backward phase”.

One major concern of ANN is the stability-plasticity dilemma. Although continuous learning is desired in ANN, further learning will cause the ANN to lose its memory when the weights have reached a steady state [**?**]. The Backpropagation algorithm, defined further, is commonly used as the training method because it allow us to adjust weights of multilayer networks, towards Generalized Delta Rule [47].

The advantage of having intermediate layers is that the neural network starts to solve problems that are not linearly separable, thus enabling the approximation of any continuous function with only one layer, and any mathematical function, when exist more than one layer. Figure 2.4 illustrates MLP graphically presenting signals (inputs and outputs) and layers (input, hidden and output).

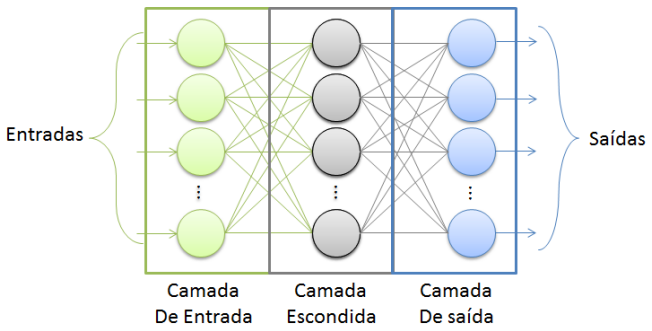

Figura 2.4: Representação gráfica da MLP com três camadas

The operation of MLP is divided in two stages: forward and backward. In forward stage, a neuron of a layer is connected to all neurons in next layer. The input signals are propagated from the input layer to the hidden layer and from the hidden layer to the output layer. Each neuron processes the inputs and provides an output. In that stage it is possible to calculate the error between the desired output to the network and the output presented by MLP. In the backward phase, the error is backpropagated and weights are adjusted using the algorithm of weight adjustment, initially random, called Backpropagation [41].

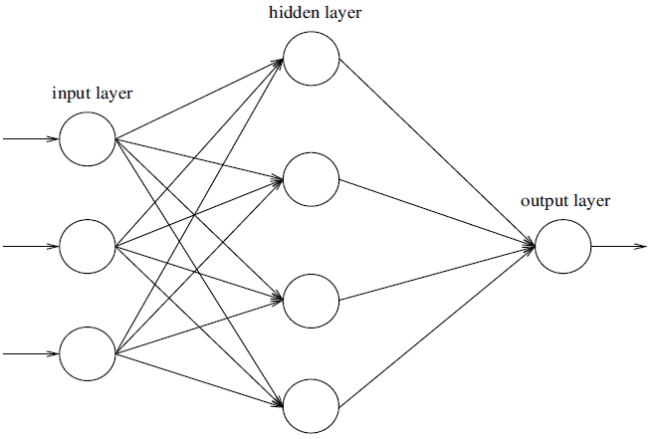

Figura 2.5: Representação gráfica da MLP com três camadas

Backpropagation Algorithm

The first phase of Backpropagation consists on inicializing weights with ramdom values. Thereafter, for each example from training subset the following steps are performed: signal propagation, forward phase and error backpropagation, backward phase. In forward phase it is possible to calculate the error between expected and calculated outcome, from MLP network. In backward phase error is backpropagated and weights are adjusted. It is necessary to calculate sensibility for each neuron. Sensibility for each layer is given by:

$$\delta_i^{m-1} = f^{m-1'}(net_j^{m-1}) \sum_{i=1}^{N_{neurons}} w_{ij}^m \delta_i^m \tag{2.14}$$

where w_{ij}^m are synaptic weights that will be adjusted, δ_i^m represents sensibility, index i represents neuron numbers of the layer that receives the signal and has $N_{neurons}$, ending $f^{m-1'}(net_j^{m-1})$ is the derivative of the activation function for neurons of layer "M-1"(the layer that emits signal) in relation to net input, net_j^{m-1}.

Weight adjustment is given by:

$$w_{ij}^m(t+1) = w_{ij}^m(t) + \alpha\delta_i^m y_j^{m-1} + \beta\Delta w_{ij}^m(t-1) \tag{2.15}$$

where α is the algorithm learning rate, β is the moment and $\Delta w_{ij}^m(t-1)$ is the correlation to weight w_{ij}^m in iteration $t-1$. The higher $alpha$ value the higher will be correlations performed in each iteration. β moment aim to making the algorithm less susceptible to getting trapped in local minima (in the original algorithm, $\beta = 0$), due to the surface of the error function be fairly complex.

During training, inputs are being presented to MLP randomly and the weights are being adjusted by backpropagation algorithm until reaches stop condition. The training stop is a critical decision because a premature stop in training can lead to underfitting (where weights of the MLP are not adjusted satisfactorily), and if the training becomes excessively long it can lead to overfitting (where the network memorizes the training

examples and loss occurs in generalizability) [50]. Aiming avoiding these two conditions, and consequently a better generalization capability, the training data set of the network can be divided in two parts: one subset for training in fact, and another for validation. Training subset is presented to the network and serves to make weight adjustments through training algorithm. validation subset serves to perform an training assessment. When REMQ of validation subset starts to increase, means that MLP is memorizing training subset and at this is appropriated interrupt the training.

MLP networks are a good approach for building classifiers, since besides coping with input-output nonlinear mapping, they have high power of generalization [51].

In [52], some clever techniques are utilized for risk analysis software projects. The paper concludes that a hybrid technique of neural networks and genetic algorithms achieved an accuracy of 85% to predict whether the project will succeed, fail completely or will be challenged.

2.5.2 Support Vector Machine

Support Vector Machine (SVM) is a machine learning technique applied to pattern recognition problems in which one seeks to achieve high generalization capability [49] [50]. The goal of SVM is to find a particular output hyperplane, called the optimal hyperplane that maximizes the margin of separation, as can be seen in Figure 2.6. In Figure 2.6, we observed two support vectors capable of linearly dividing output hyperplane into two classes. The variable r is the desired algebraic distance from support vectors to the optimal hyperplane separating each class. The larger the distance, the higher the generalization ability in support vector machine.

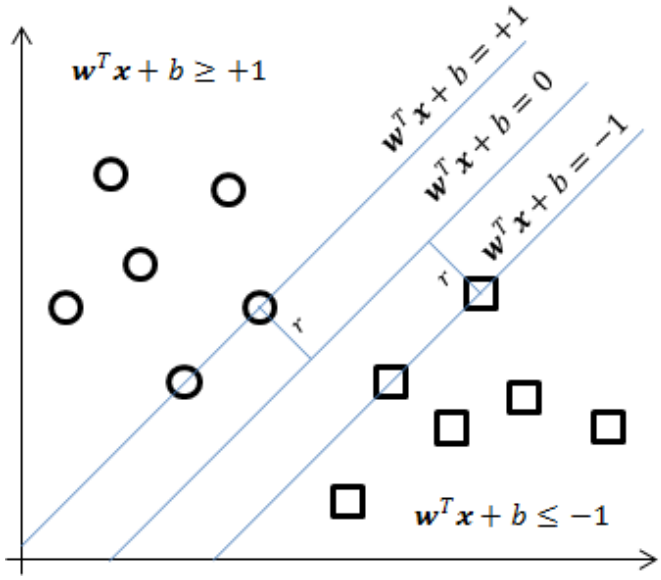

Figura 2.6: Vetores de Suporte e Hiperplano de Separação ótimo

SVMs were developed based on statistical learning theory, detailed by Vapnik in 1995 and 1998 [53] [54], and initially proposed by Vapnik and Chervonenkis in 1971

[55].

The development of SVMs aimed to obtain algorithms with high generalization capability. According to the theory of statistical learning this generalization error called Functional Risk can be calculated if one knows the probability distribution of the population [54]. However, for supervised learning problems generally the probability distribution of the population is not available, since in real problems it is used a population sample. Therefore it is only possible to calculate an error function, such as REMQ. This error function calculated from the sample data is called Empirical Risk. Depending on the sample size used for training, minimizing the training error does not necessarily mean minimizing the generalization error or Functional Risk.

An important concept in statistical learning theory is the VC dimension (Vapnik-Chervonenkis). The VC dimension is a scale index that measures the intrinsic complexity of a class of functions. A definition of the VC dimension is that this represents the maximum number of training examples that a machine learning is able to correctly classify, for all possible binary combinations of these data.

Thus, according to the statistical learning theory proposed by Vapnik [54], a neural network or any other machine learning during the supervised learning obtains its maximum capacity of generalization when during training it minimizes the functional risk. Minimising Functional Risk is equivalent to minimize empirical risk and structural risk associated with model complexity. The Functional Risk is limited, with probability $1-\delta$ by Equation 2.16.

$$R_{funcional} \leq R_{empirico} + R_{estrutural} \tag{2.16}$$

The Structural Risk can be calculated, as shown by Vapnik [54] as:

$$R_{estrutural} = \sqrt{\frac{h\left(\log\left(\frac{2N}{h}\right)+1\right)-\log\left(\frac{\delta}{4}\right)}{N}} \tag{2.17}$$

where h is the dimension VC.

The initial formulation of SMVMs for linearly separable problems was called rigid edges (maximum). The difference of these to a perceptron network is in the method of selecting the hyperplane that separates the classes. While a perceptron network seeks any hyperplane that satisfies the problem, a rigid margin SVM seeks the optimal hyperplane, ie, one whose margin of separation is maximum.

This way, the determination of SVMs weights is transformed into a constrained optimization problem, which requires a higher computational effort. The optimization of the SVMs weights with restricted maximum margin has been solved by the method of Lagrange multipliers.

Subsequently, flexible margins SVMs were developed with the aim of dealing with the training samples with errors and finally the SVM using nonlinear functions in the hidden layer of different base.

For a better understanding of SVMs consider a linear classification problem with linear input vector x_i $(i = 1,...,n)$, where n is the number of examples and outputs y_i (+1 ou -1). The selection of parameters should occur such that:

$$w^T.x_i + b \geq +1, \quad se \quad y_i = +1 \tag{2.18}$$
$$w^T.x_i + b \leq -1, \quad se \quad y_i = -1 \tag{2.19}$$

or in summary

$$y_i(w^T.x_i + b) \geq 1, \qquad i = 1,..,n \tag{2.20}$$

Margin d is the distance between two parallel planes and can be expressed as an euclidean distance in Equation 2.21.

$$d = \frac{|w^T.x_i + b|}{||w||} + \frac{|w^T.x_i + b|}{||w||} = \frac{2}{||w||} \tag{2.21}$$

where $||w|| = \sqrt{w_1^2 + w_2^2 + ... + w_n^2}$ is the euclidian norm of weights vector.

Therefore, maximizing the margin becomes a constrained optimization problem to minimize the following objective function $\min_w \frac{||w||^2}{2}$ subject to Equation 2.20.

This is a nonlinear optimization problem (the objective function is quadratic) with linear constraints which can be solved by means of Lagrange multipliers.

2.5.3 Radial Basis Function Network

Radial Basis Function (RBF) network emerged in 1988 as a possible alternative to MLP networks. In its traditional formulation is composed of three layers: an input layer, a hidden layer and an output layer. The main differences between the MLP and RBF networks are:

1. The neurons in the hidden layer have only radial basis functions as activation functions, which are located in such a way that only a few hidden units will be activated when receiving a given set of input samples;

2. In MLP networks activation functions have as a weighted average net inflow among input samples and the set of weights. On the other hand, in RBF networks the use of these activation functions of radial basis make it fired through a weighted norm of the difference between the input value and the radial basis function center;

3. The output layer is composed of linear processing units.

The most commonly used activation functions are [41] [48]:

1. Linear Function

$$\phi_i(x) = ||x_j - \mu_i|| \tag{2.22}$$

2. Cubic Function

$$\phi_i(x) = ||x_j - \mu_i||^3 \tag{2.23}$$

3. Gaussian Basis Function

$$\phi_i(x) = exp\left(-\frac{||x_j - \mu_i||^2}{2\sigma_i^2}\right) \tag{2.24}$$

4. Logistic Function

$$\phi_i(x) = \frac{1}{1 + exp\left(-\frac{||x_j - \mu_i||^2}{\sigma_i^2 - \theta}\right)} \tag{2.25}$$

5. Multi-quadratic Basis Function

$$\phi_i(x) = \sqrt{||x_j - \mu_i||^2 + \sigma_i^2} \tag{2.26}$$

6. Inverse Multi-quadratic Basis Function

$$\phi_i(x) = \frac{1}{\sqrt{||x_j - \mu_i||^2 + \sigma_i^2}} \tag{2.27}$$

7. Thin Blade Spline Basis Function

$$\phi_i(x) = \frac{||x_j - \mu_i||}{\sigma_i^2} \log\left(\frac{||x_j - \mu_i||}{\sigma_i}\right) \tag{2.28}$$

where, x_j are input samples, μ_i and σ_i represents respectively the center and the width (dispersion) of the i-th radial basis function and θ is an adjusted bias.

The calculation of the response of RBF corresponding to the neurons of the output layer is performed by Equation 2.29.

$$y_j = \sum_{j=1}^{s} w_{Kj}\phi_K(x) \tag{2.29}$$

The training of RBF networks can be accomplished in several ways. Among the proposed approaches, the most adopted training is called hybrid training, because it uses an unsupervised learning to determine the parameters of radial basis functions of the hidden layer and supervised learning to adjust the weights that links the hidden layer to the output layer. Therefore, since the radial basis functions are considered fixed after determining its parameters, adjusting the weights that links the hidden layer to the output layer for the RBF network is equivalent to an ADALINE network. In this case, when using linear neurons in the output layer and an objective function such as mean

square error, the training of these networks is very fast since we have a system of linear equations to be solved, which allows us to use linear techniques of matrix inversion.

During the first phase of training, unsupervised learning, the centers of radial basis functions can be determined by clustering algorithms such as k-means algorithm [56] and self-organized kohonen maps [50].

K-means method has the objective to find a set of K representative centers, μ_i (i=1,2,...,K), of radial basis function in terms of input samples x_j with $j = 1,2,...,N$. The algorithm divides the input data set in K subsets S_i each one containing N_i examples. The purpose of the method is to minimize Equation 2.30.

$$F = \sum_{i=1}^{K} \sum_{j \in S_i} ||x_j - \mu_i||^2 \tag{2.30}$$

where, μ_i is the mean of points belonging to S_i set calculated by

$$\mu_i = \frac{1}{N_i} \sum_{j \in S_i} x_j \tag{2.31}$$

Initially, the input samples are randomly located within each subset K and then the centers μ_i are calculated. Subsequently, the samples are redistributed according to its closer proximity with regard to these centers. This process is then repeated until there are no more changes between the groups of examples. This method also allows to determine the value of the width (dispersion) σ_i of radial basis function using the Euclidean distance.

Soon after completion of the non-supervised training where it was determined the parameters of radial basis functions, is held the second part of the training that consists of a supervised training where the weights that links the hidden layer to the output layer are determined. In this phase, the training is similar to a traditional network with use of an objective function such as mean square error. Several optimization techniques can be used for learning the weights of the output layer, responsible for the activation of the linear combination of the hidden layer, such as the delta rule, the method of least squares technique pseudo-inversion and Orthogonal Least Squares (OLS).

RBF networks are universal function approximators such as MLP. In general RBF networks have a training duration less than in MLP, since the hybrid RBF training techniques permits the use of linear fast convergence for determining the weights (between hidden and output layers) during supervised learning.

RBF networks such as local learning networks, require a larger number of samples for its training to obtain an accuracy similar to that MLP networks, which are networks of global adjustment. This implies that the MLP are usually better approximators of functions because the global adjustment tends to provide greater generalizability. However, in classification problems, the MLP tend to make greater misclassification ”false positive”than RBF networks

2.5.4 Learning Rules

A property of major importance for a neural network is its ability to learn from its environment and improve its performance. The improvement in performance occurs over time according to some predetermined measure. A neural network learns about its environment through an iterative process of adjusting its synaptic weights and bias levels. Ideally, the network becomes more educated about their environment after each iteration of the learning process.

Gradient Descent Learning

Gradient Descent (GD) requires the definition of an error (or objective) function to measure the neuron's error in approximating the target. The sum of square errors is usually used 2.32. The aim of GD is to find the weight values that minimizes the error function. This is achieved by calculating the gradient of the error function in weight space, and to move the weight vector along the negative gradient as illustrated for a single error in Figure 2.7 [48].

$$\varepsilon = \sum_{i=1}^{P_T} (e_i - c_i)^2 \tag{2.32}$$

where e_i and c_i are respectively the expected and calculated output for the i-th pattern, and P_T is the total number of input-target vector pairs (*patterns*) in the training set.

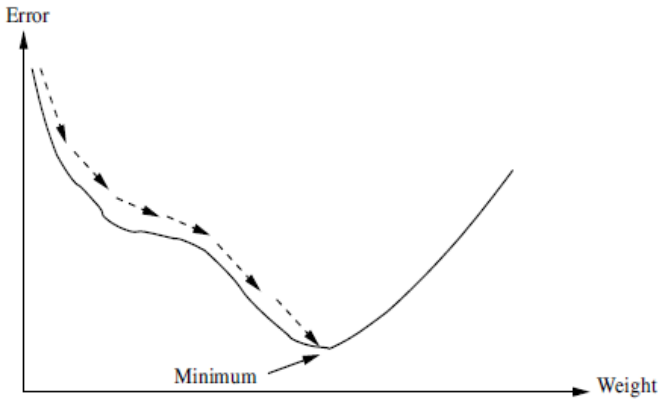

Figura 2.7: Vetores de Suporte e Hiperplano de Separação ótimo

Given a single training pattern, weights are updated using

$$v_i(t) = v_i(t-1) + \Delta v_i(t) \tag{2.33}$$

where

$$\Delta v_i(t) = \eta(-\frac{\partial \varepsilon}{\partial v_i}) \tag{2.34}$$

and η is the learning rate (i.e. the size of steps taken in the negative direction of the gradient). The Widrow-Hoff learning rule presents a solution for the step and ramp functions, while the generalized delta learning rule assumes continuous functions that are at least once differentiable [48].

Widrow-Hoff Learning

Widrow-Hoff learning rule assumes that objective function $f = net_i$. Then $\frac{\partial f}{\partial net_i}$. This learning rule is also referred to as the least-mean-square (LMS) algorithm, was one of the first algorithms used to train layered neural networks with multiple adaptive linear neurons (Madaline) [48].

Error Correction Learning

Delta rule is a generalization of the Widrow-Hoff rule [44] that uses activation functions that have real derivative throughout its domain and at least one minimum value. The learning rule is given by:

$$\Delta w_{ij} = \alpha(d_i - y_i)x_j f'(net_i) \tag{2.35}$$

in other words, that means the adjustment made to a synaptic weight of a neuron is proportional to the product of the error signal by the input signal from the synapse in question [51]. Thus, the adjustment to be applied to the weights is:

$$w_{ij}(new) = w_{ij}(old) + \alpha(d_i - y_i)x_j f'(net_i) \tag{2.36}$$

This algorithm ensures minimization of the error over time, by adjusting the weights, in other words, their convergence ensures that the adaptation of the weights is performed in a finite number of iterations. The proof of convergence can be found in Beale and Jackson [58].

Memory based Learning

In memory-based learning, all (or most of) previous experiences are explicitly stored in a large memory input-output examples classified correctly: $\{(x_i, d_i)\}_{i=1}^{N}$, where x_i represents an input vector and d_i is the corresponding desired response. Without loss of generality, one may restrict the desired response to a scalar. In a classification problem of binary patterns, for example, there are two classes/hypotheses to be considered, represented by ζ_1 e ζ_2. In this example, the desired response d_i is set to 0 (or -1) for the class ζ_1 and 1 for the class ζ_2. When you want to classify a test vector x_teste (not seen before), the algorithm responds searching and analyzing training data in a "local neighborhood"of x_teste.

Every learning algorithms based on memory involves two essential ingredients:

1. The criterion used to define the local neighborhood of the test vector x_teste;
2. The learning rule applied to the training samples in the local neighborhood x_teste.

On a more effective type of memory based learning known as the rule of the nearest neighbor, the local neighborhood is defined as the training example that is in the immediate vicinity of the test vector $x_t este$. In particular, it is said that the vector

$$x'_N \in \{x_1, x_2, ..., x_N\}, \tag{2.37}$$

is the nearest neighbor of $x_t este$ of

$$\min_i d(x_i, x_t este) = d(x'_N, x_t este), \tag{2.38}$$

where $d(x_i, x_t este)$ is the euclidean distance between vectors x_i and $x_t este$. The associated class with the minimum distance, that means, the vector x'_N is displayed as classification $x_t este$. This rule is independent of the fundamental distribution responsible for generating training samples.

Scaled Conjugate Gradient

Conjugate gradient optimization trades off the simplicity of GD and the fast quadratic convergence of Newton's method. Several conjugate gradient learning algorithms have been developed, most of which are based on the assumption that the error function of all weights in the region of the solution can be accurately approximated by Equação 2.39 [48]:

$$\varepsilon_T(D_T, w) = \frac{1}{2} w^T H w - \theta^T w \tag{2.39}$$

where H is the Hessian matrix. Since the dimension of the Hessian matrix is the total number of weights in the network, the calculation of conjugate directions on the error surface becomes computationally infeasible. Computationally feasible conjugate gradient algorithms compute conjugate gradient directions without explicitly computing the Hessian matrix, and perform weight updates along these directions.

An important aspect in conjugate gradient methods is that of direction vectors, $\{p(0), p(1), ..., p(t-1)\}$. These vectors are created to be conjugated with the weight vector w. That is, $p^T(t_1) w p(t_2) = 0$ for $t_1 \neq t_2$. A new conjugate direction vector is generated at each iteration by adding to the calculated current negative gradient vector of the error function a linear combination of the previous direction vectors. The standard conjugate gradient algorithm assumes a quadratic error function, in which case the algorithm converges in no more than n_w steps, where n_w is the total number of weights and biases. The Fletcher-Reeves conjugate gradient algorithm does not assume a quadratic error function. The algorithm restarts after n_w iterations if a solution has not yet been found [48].

The scale factors can also be calculated through Polak-Ribiere and Hestenes-Stiefer methods. Moller [59] proposed the scaled conjugate gradient (SCG) algorithm as a batch learning algorithm. Step sizes are automatically determined, and the algorithm is restarted after n_w iterations if a good solution was not found.

Quasi-Newton Learning

The derivatives in minimization algorithms are the gradient and the Hessian. The gradient must be known accurately as descent directions have to be calculated from them and approximations to gradient do not provide the required accuracy. On the other hand, the Hessian can be approximated by secant techniques. Since the Hessian is the Jacobian of the nonlinear system of equations $\nabla F(x) = 0$, it could be approximated. But, the Hessian has two important properties: it is always symmetric and often positive definite. The incorporation of these two properties into the secant approximation is an important aspect of the BFGS-BP and OSS-BP methods discussed subsequently. They are most often called positive definite secant updates [60].

Broyden-Fletcher-Goldfarb-Shanno back-propagation method

In Newton's method a quadratic approximation is used instead of a linear approximation of the function $F(x)$. The next approximate solution is obtained at a point that minimizes the quadratic function in the Equação 2.40.

$$F(x_{k+1}) = F(x_k + \Delta x_k) = F(x_k) + g_k^T \Delta x_k + \frac{1}{2}\Delta x_k^T A_k \Delta x_k \tag{2.40}$$

Hence, the obtained sequence is the Equation 2.41.

$$x_{k+1} = x_k - A_k^{-1} g_k \tag{2.41}$$

The main advantage of the Newton's method is that it has a quadratic convergence rate, while the steepest descent has a much slower, linear convergence rate. However, each step of Newton's method requires a large amount of computation. Assuming that the dimensionality of the problem is N, an O(N^3) floating-point operation is needed to compute the search direction d^k. A method that uses an approximate Hessian matrix in computing the search direction is the quasi-Newton method. Let H_k be an $N \times N$ symmetric matrix that approximates the Hessian matrix A_k; then the search direction for the quasi newton-Newton method is obtained by minimizing the quadratic function, described by Equation 2.42 [60]

$$F(x_{k+1}) = F(x_k) + g_k^T \Delta x_k + \frac{1}{2}\Delta x_k^T H_k \Delta x_k \tag{2.42}$$

As the matrix H_k is to approximate the Hessian of the function $F(x)$ at $x = x_k$, it needs to be updated from iteration to iteration by incorporating the most recent gradient information [60].

One-step Secant backpropagation method

One drawback of the BFGS update of Equation 2.42 is that it requires storage for a matrix of size $N \times N$ and calculations of order O(N^2). Although the available storage is less of a problem now than it was a decade ago, the computational problem still exists for large N. It is possible to use a secant approximation with O(N) computing. In this

method, the new search direction is obtained from vectors computed from gradients. If g_{k+1} is the current gradient, the new search direction p_{k+1} is obtained as the Equation 2.43 [60],

$$p_{k+1} = -p_{k+1} + B_k y_k + C_k s_k \tag{2.43}$$

where $g_{k+1} = \nabla F(x_{k+1})$ and the two scalars B_k and C_k are the following combination of scalar products of the previously defined vectors s_k, g_{k+1} and y_k (last step, current gradient and difference of gradients)

$$B_k = \frac{s_k^T g_{k+1}}{s_k^T y_k} \tag{2.44}$$

and

$$C_k = 1\left(1 + \frac{y_k^T y_k}{s_k^T y_k}\right)\frac{s_k^T g_{k+1}}{s_k^T y_k} + \frac{y_k^T g_{k+1}}{s_k^T y_k}. \tag{2.45}$$

The backtracking line search is used with the quasi-Newton's OSS optimization algorithm. At the beginning of learning, the search direction is $-g_0$ and it is restarted to $-g_{k+1}$ every N steps. The multiplier increases the last successful learning rate and first tentative step is executed. If the energy of the network is higher than the upper limiting value, then a new tentative is tried by using successive quadratic interpolation until the requirement is met. The learning rate is decreased to half after each unsuccessful trial [60].

Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm adaptively varies the parameter updates between the gradient descent update and the Gauss-Newton update [61],

$$\varepsilon_T(D_T, w) = \frac{1}{2} w^T H w - \theta^T w \tag{2.46}$$

where small values of the algorithm parameter λ result in a Gauss-Newton update and large values of λ result in a gradient descent update. The parameter λ is initialized to be large. If an iteration happens to result in a worse approximation, λ is increased. As the solution approaches the minimum, λ is decreased, the Levenberg-Marquardt method approaches the Gauss-Newton method, and the solution typically converges rapidly to the local minimum [61].

Marquardt's suggested update relationship in Equation 2.47.

$$\varepsilon_T(D_T, w) = \frac{1}{2} w^T H w - \theta^T w \tag{2.47}$$

2.6 Neuro-Fuzzy Systems

2.6.1 ANFIS: Adaptive Neuro-Fuzzy Inference Systems

In this section, we present a class of adaptive networks that are functionally equivalent to fuzzy inference systems, analyzed during this study. The architecture is referred to as ANFIS, which stands for adaptive network-based fuzzy inference system or semantically equivalently, adaptive neuro fuzzy inference system. We describe how to decompose the parameter set to facilitate the hybrid learning rule for ANFIS architectures representing both the Sugeno and Tsukamoto fuzzy models. We also demonstrate that under certain minor constraints, the radial basis functioin network (RBFN) is functionally equivalent to the ANFIS architecture for the Sugeno fuzzy model [62].

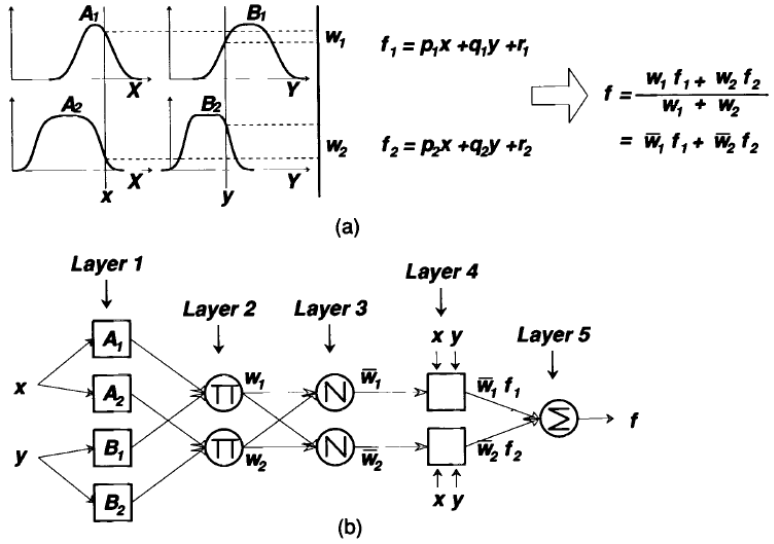

Figura 2.8: (a) A two-input first-order Sugeno fuzzy model with two rules; (b) equivalent ANFIS architecture.

ANFIS Architecture

For simplicity, we assume that the fuzzy inference system under consideration has two inputs x and y and one output z. For a first-order Sugeno fuzzy model, a common rule set with two fuzzy if-then rules is the following:

1. Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1x + q_1y + r_1$,
2. Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2x + q_2y + r_2$.

Figure 2.8(a) ilustrates the reasoning mechanism for this Sugeno model; the corresponding equivalent ANFIS architecture is as shown in Figure 2.8(b), where nodes of the same layer have similar functions, as described next (Here we denote the output of the i-ésimo node in layer l as $O_{1,i}$).

Layer 1: Every node i in this layer is an adaptive node with a node function

$$O_{1,i} = \mu_{A_i}(x), for i = 1,2 or O_{1,i} = \mu_{B_{i-2}}(y), for i = 3,4 \tag{2.48}$$

where x (or y) is the input to node i and A_i (or B_{i-2}) is a linguistic label (such as "small"or "large") associated with this node. In other words, $O_{1,i}$ is the membership grade of a fuzzy set $A(A_1, A_2, B_1 or B_2)$ and it specifies the degree to which the given input x (or y) satisfies the quantifier A. Here the membership function for A can be any appropriate parameterized membership function, such as the generalized bell function [62]:

$$\mu_A(x) = \frac{1}{1 + \left\| \frac{x - c_i}{a_i} \right\|^{2b}}, \tag{2.49}$$

Layer 2: Every node in this layer is a fixed node labeled Π, whose output is the product of all the incoming signals:

$$O_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_i}(y) for i = 1,2. \tag{2.50}$$

Each node output represents the firing strength of a rule. In general, any other T-norm operators that perform fuzzy AND can be used as the node function in this layer.

Layer 3: Every node in this layer is a fixed node labeled N. The i-th node calculates the ratio of the ith rule's firing strength to the sum of all rules' firing strengths:

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2} for i = 1,2. \tag{2.51}$$

For convenience, outputs of this layer are called **normalized firing strengths**.

Layer 4: Every node i in this layer is an adaptive node with a node function

$$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i), \tag{2.52}$$

where $\overline{w_i}$ is a normalized firing strength from layer 3 and p_i, q_i, r_i is the parameter set of this node. Parameters in this layer are referred to as "consequent parameters".

Layer 5: The single node in this layer is a fixed node labeled $\sum$, which computes the overall output as the summation of all incoming signals:

$$overall output = O_{5,i} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}, \tag{2.53}$$

Thus we have constructed an adaptive network that is functionally equivalent to a Sugeno fuzzy model. Note that the structure of this adaptive network is not unique; we can combine layers 3 and 4 to obtain an equivalent network with only four layers. By the same token, we can perform the weight normalization at the last layer. In the extreme case, we can even shrink the whole network into a single adaptive node with the same parameter set. Obviously, the assignment of node functions and the network configuration are arbitrary, as long as each node and each layer perform meaningful and modular functionalities [62].

The extension from Sugeno ANFIS to Tsukamoto ANFIS is straightforward, where the output of each rule $(f_i, i = 1,2)$ is induced jointly by a consequent membership

function and a firing strength. For the Mamdani fuzzy inference system with max-min composition, a corresponding ANFIS can be constructed if discrete approximations are used to replace the integrals in the centroid defuzzification scheme. However, the resulting ANFIS is much more complicated than either Sugeno ANFIS to Tsukamoto ANFIS. The extra complexity in structure and computation of Mamdani ANFIS with max-min composition does not necessarily imply better learning capability or approximation power [62].

Capítulo 3

Methodology

In this dissertation, it is proposed a methodology for risk analysis in software projects, from historical database of risk registers, through the use of artificial neural networks. To develop this methodology, first it is needed an extensive and real database of risks in software projects. However, there is a difficulty in finding publicly bases of representative and reliable data. A database called risk PERIL [2] showed to meet basic needs for this research, in addition to already be fully classified (more details in Section 3.1). This study requires an extensive database of risk registers from various projects around the world and in different time periods in order to show evidence of which are the main modes of failure in projects and a standard for explore risk responses.

Second, it is necessary to perform data preprocessing for input variables to be converted into numeric, standardized and selected for the study. Third, evaluate the performance of state of the art tools about forecast error, Monte Carlo simulation and PERT Analysis.

Fourth, implement each prediction model so we could select the best model in artificial neural networks: Multilayer Perceptron, Support Vector Machine, Radial Basis Function and the Adaptive Neuro-Fuzzy Inference System. Some of these selected models have some parameters and due to the diversity of possible values for each parameter, one needs to optimize the parameters of the models. This study implements a variation of meta-heuristic Particle Swarm Optimization (PSO) with a constriction coefficient of Clerk [48] to accomplish the task of optimizing the parameters of ANNs. The search space of the problem of optimization of these parameters is multi-dimensional, complex and has many local minima.

In Figure 3.1, it is observed a schematic showing that an optimization algorithm performs the four models in its various parametric settings so you can choose the most efficient model for the chosen database.

Fifth, a validation test of outcomes is carried out to check whether models based on artificial neural networks have lower error than state of the art models in risk analysis: Monte Carlo Simulation, PERT Analysis, Multiple Linear Regression, Regression Tree Model. The first two models have been implemented and are also evaluated for error in prediction. The last two, were considered baseline for being simpler models since they perform linear regression.

The selection of the best model is made in terms of accuracy in estimating the im-

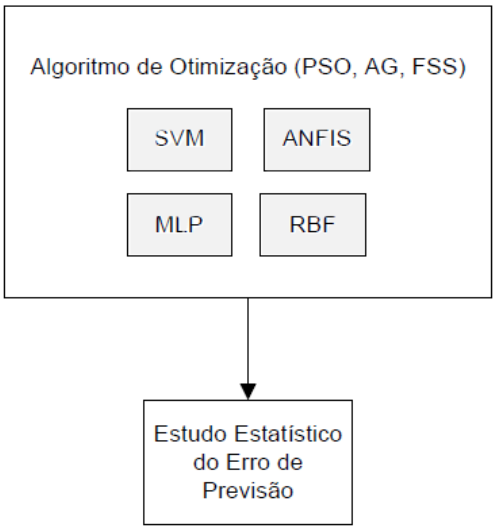

Figura 3.1: Schema for selection of best ANN

pacts of risks. It is difficult to obtain a representative measure of the accuracy of a model. However, Engelbrecht [48] and Saxena [25] suggest that the Root Mean Square Error (RMSE) is a convenient measure applicable to most problems. Accuracy in this case means the degree of closeness calculated for an expected output. RMSE is represented by Equation 3.1:

$$REMQ = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}(e_i)^2}, \tag{3.1}$$

where $e_i = f_i - y_i$, f_i is the calculated outcome, y_i is the expected outcome and n is the number of data pairs.

The eight selected techniques have predicted the outcome to risk impacts. Root Mean Square Error is calculated thirty times for each method. Nevertheless, a Non-paired Wilcoxon Test [63] may be necessary to assert which is a more efficient approach to fit dataset (e.g. PERIL). Non-paired Wilcoxon Test is used because there were no evidence that the samples came from a normally distributed population, either there were no relation between outcomes from different samples.

Furthermore, cross-validation [64] must be used to avoid the occurrence of overfitting or underfitting of data training. In this situation *early stopping* training is used to identify the beginning of overfitting because this method has been proved to be capable of improving the generalization performance of the ANN over exhaustive training [**?**] [48] [65]. Therefore, cross-validation method is used for each alternative, excluding Monte Carlo Simulation and PERT Analysis, to promote higher generalization performance. When cross-validation is adopted, it means that it is needed to partition the database into three disjoint groups: training, cross-validation and test subset. Training set is used in training phase the RNA's, at which learning takes place. Cross-validation subset is processed simultaneously with training subset and determine the stop in network training since test subset is used to obtain accuracy of the model, RMSE.

Finally, risk impact estimation and establishment of a confidence interval for a sample of the training set are obtained using the most accurate prediction model after validation tests. It is necessary to define a confidence interval of prediction for project managers and risk analysts establish the confidence level according to their needs. After all, this is the result they expect.

Figure 3.2 presents a flowchart with the established methodology for this study. The procedure begins with the selection of the database that serves as a set of input data, in this case the PERIL, and ends with the activity ”Definition of Confidence Interval”. All activities are executed sequentially, except activities ”Evaluation of Models of State of the Art”and ”Best Selection of Artificial Neural Network”that run in parallel.

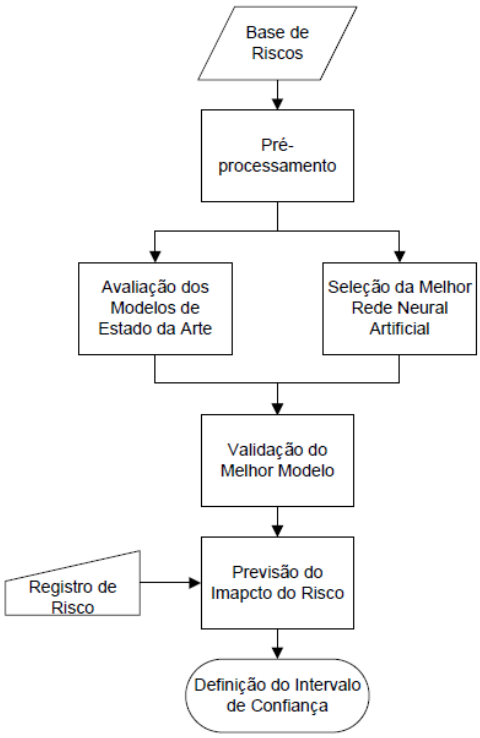

Figura 3.2: Flowchart of performed study

3.1 PERIL Database

A better risk management starts identifying potential problems, asserted here as risk factors. The adoption of available methods like: reviewing learned lessons, brainstorming, interviews and specialized judgment are suposed to be relative efficient alternatives, otherwise in most of situations it involves high costs. A low cost, extensive and accessible proposal is to utilize Project Experience Risk Information Library (PERIL) database [2]. PERIL database provides information and experience of other project risk management.

For more than a decade, in Risk Management Workshops, Kendrick have collected anonymous data from hundred of project leaders dealing with their past project pro-

blems. He has compiled this data in the PERIL database, which summarizes both a description of what went wrong and the amount of impact it had on each project. The dataset provides a sobering perspective on what future projects may face and is valuable in helping to identify at least some of what might otherwise be invisible risks or black swans [2].

According to Kendrick, in projects, the identified risks can be classified as ”known”, those anticipated during planning, or ”unknown”further identified during project execution. The purpose of this dataset is to provide a framework to identify risks, in such a way to increase the number of ”known”, and decrease the amount of ”unknown”risks [2].

Some characteristics of PERIL are:

- the data are not relational, they contain only a small fraction of the tens of thousands projects undertaken by the project leaders from whom they were collected;
- they present bias, the information was not collected randomly;
- they represent only the most significant risks;
- they are worldwide, with a majority from the Americas;
- they do not identify opportunities;
- they contain six hundred and forty nine registers, whose relative impact is based on the number of weeks delayed the project schedule;
- typical project had a planned duration between six months and one year;
- typical staffing was rarely larger than about twenty people.

Risk registers are categorized as scope, schedule and resource. Scope is decomposed in change and defect subcategories. Schedule is decomposed in dependency, estimative and delay subcategories. Resources is decomposed in money, outsourcing and people subcategories. One benefit of PERIL is that the author contemplates ”black swans”: risks with large impact, difficult to predict and with rare occurrence [66].

Kendrick chose to normalize all the quantitative data using only time impact, measured in weeks of project slippage. This tactic made sense in light of today's obsession with meeting deadlines, and it was an easy choice because by far the most prevalent serious impact reported in this data was deadline slip. Focusing on time is also appropriate because among the project triple constraints of scope, time ans cost, time is the only one that's completely out of our control - when it's gone, it's gone [16].

Table 3.1 shows the number of cases, the cumulative and average impact in weeks for each category and sub-category of root cause, beyond the meaning of each subcategory.

A disadvantage of this database is that it only accounts for risks that negatively impact on the project. The opportunities were not identified and maximized in that study. However, A major benefit is that the author presents some risks as black swans [16]: representing the idea of risks with broad impact, hard to predict and rare to occur. If the risk has negative impact, is known as a catastrophe, whereas, if you have positive impact, is known as a reward.

Tabela 3.1: Total project impact by root-cause categories and subcategories [16]

Root-Cause Subcategories	Definition	Cases	Cumulative Impact(weeks)	Average Impact(weeks)
Scope: Changes	Revision made to scope during the project	177	1,460	8.2
Resource: People	Issues arising from internal staffing	123	706	5.7
Scope: Defects	Failure to meet deliverable requirements	93	654	7.0
Schedule: Delays	Project slippage due to factors under the control of the project	102	509	5.0
Schedule: Estimates	Inadequate durations allocated to project activities	49	370	7.6
Resource: Outsourcing	Issues arising from external staffing	47	316	6.7
Schedule: Dependencies	Project slippage due to factors outside the project	41	262	6.4
Scope: Changes	Insufficient project funding	17	228	13.4

3.1.1 Black Swans

Calling some risks ”black swans”has been popularized of late by the writings of Nassim Nicholas Taleb [66]. The notion of a ”black swan”originated in Europe before there was much knowledge of the rest of the world. Because all the swans observed in Europe were white, a black swan was deemed impossible. It came as something of a shock when a species of black swans was later discovered in Australia. This realization gave rise to the metaphorical use of the term ”black swan”to describe something erroneously believed to be impossible.

Taleb’s concept of ”black swan”is a large-impact, hard-to-predict, rare event. It is nonetheless applicable to project risk management. In projects, it is common for project leaders to discount major project risks because they are estimated to have extremely low probabilities. But these risks do occur - The PERIL database is full of them - and the severity of problems they cause means that ignoring them can be unwise. When these risks do occur, the same project managers who initially dismissed them come to perceive them as much more predictable - sometimes even inevitable [16].

In PERIL database, there are 127 cases representing the most schedule slippage. As the database shows, these most damaging risks are not as rare as might be thought, and they need not be so difficult for project managers to predict if they get appropriate attention in the risk management process [16]. In many situations, the most difficult task is to identify and estimate ”black swans”due to its characteristic: emergent, unexpected, unpredictable and extreme impact events. Therefore, ”black swans”also will be included in this study.

3.2 Data Preprocessing

PERIL contains nominal and numeric values. So, nominal variables were expressed through binary variables. In that point, it is used fifteen binaries variables to represent nine nominal variables. Second, impact which represents the real output, are integer numbers. It has been noticed that impact probability distribution function fits with log-normal, gamma functions. Therefore, it was done a gamma data normalization [38]. The selection of the most significant variables for the study was performed after the results of the analysis promoted by Random Forest algorithm proposed in Luís Torgo book [37].

Figure 3.3 e 3.4 introduced input variables in histograms. All data are binary values represented by bar graphs, that means the number of occurrences for each value interval. Figure 3.5 presents gamma normalized real outcome from PERIL in a histogram. A shape of the distribution fitting function is also presented in a curve under the histogram.

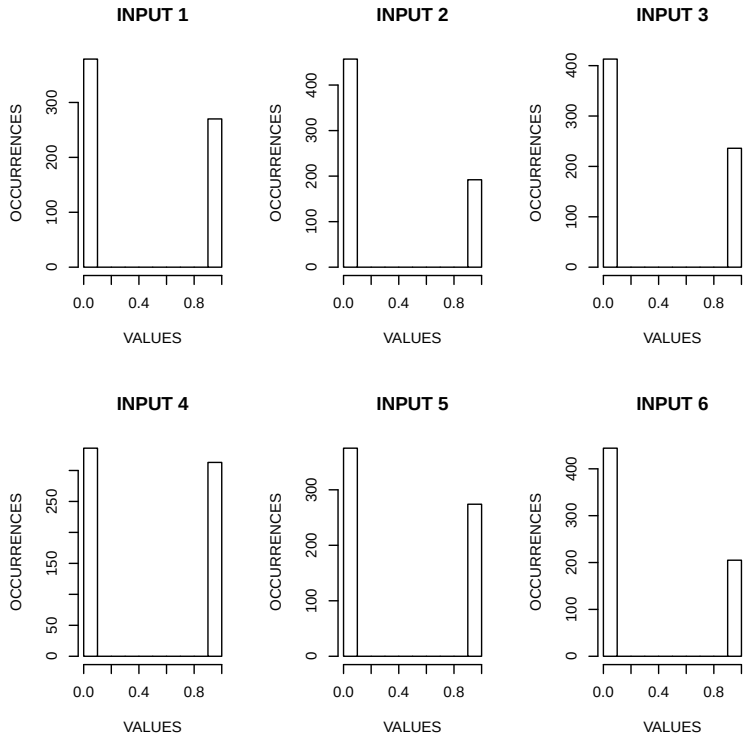

Figura 3.3: First six input variables

For our purpose, PERIL was split into three disjoint subsets - training, cross-validation and test subsets, corresponding to fifty, twenty-five and twenty five percent of the data-set, respectively. *Split-sample* cross-validation method was used for MLRM and RTM models. Whereas *early stopping* and *split-sample* cross-validation methods were combined and used for MLP, SVM, RBF and ANFIS training [67].

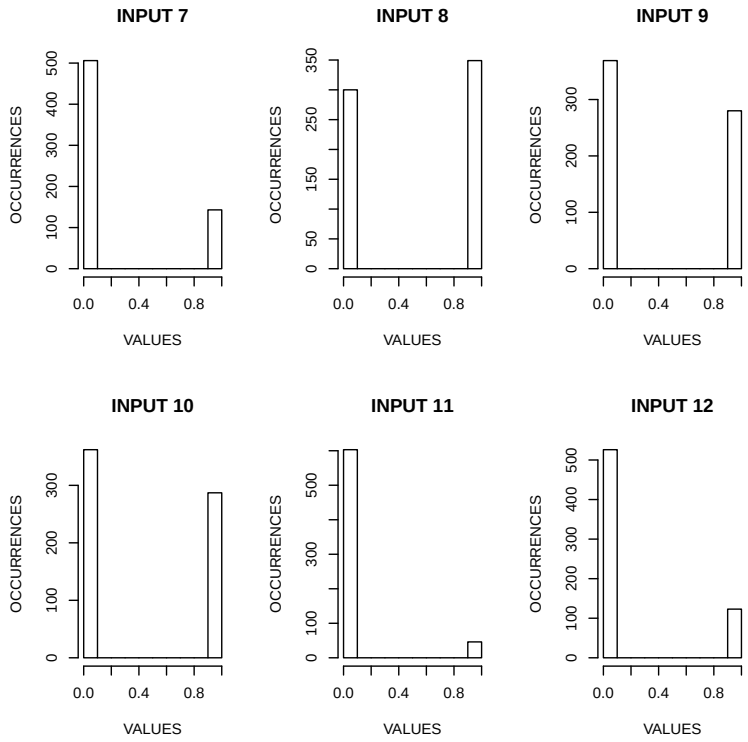

Figura 3.4: Last six input variables

3.3 State of art models

In this section, adopted configurations for experiments of state of art models are described.

3.3.1 Monte Carlo Simulation

MCS technique used the entire dataset in order to increase the performance prediction. It was filtered only the possible real outcomes to generate the calculated outcome. Towards this decision, we have reduced prediction issues and have improved its performance.

3.3.2 PERT Analysis

PERT analysis also used the entire database and only possible output for a given input was selected to calculate the new output, like MCS. From this setup the result of this model has been improved compared to a scenario where the database was not filtered.

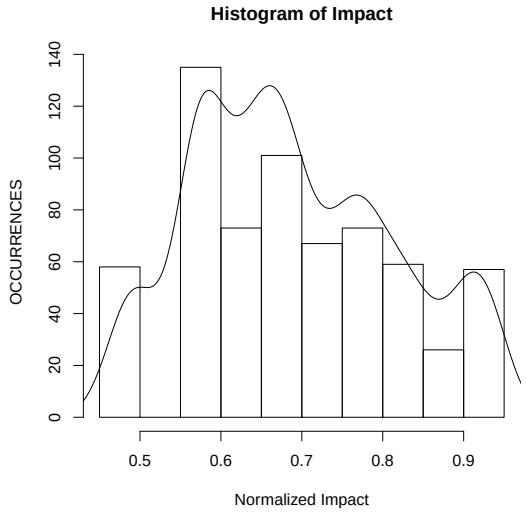

Figura 3.5: Histogram of impact and shape of the distribution fitting function

3.4 Linear Regression Model

In this section, the configurations adopted for experimentation with linear regression models are described.

The source code of MLR models were adapted from Torgo [37] in order to perform linear regression model training, cross-validation, outcome prediction and MAE evaluation. MLR and RTM models were analyzed statistically to define the baseline linear regression model for further analysis.

A study with these linear regression models are necessary because there is not a standard study to estimate risk impacts using PERIL. Therefore, during the experiments, the evaluation of the best linear regression model aims to define an upper limit value of RMSE. That means models who get errors above this upper limit are unsatisfactory.

3.4.1 Multiple Linear Regression Model

MLRM is a simplest possible model for impact estimation and risks in PERIL linear regression. After optimizing of Linear Regression Model by selecting the best input variables based on Akaike Information Criterion (AIC), seven of eleven variables were selected plus the independent term.

3.4.2 Regression Tree Model

MRA model builds a classification tree of input variables, in this case it may not be necessary to use all input variables for model construction. It tries to get errors smaller than MLRM through the selection of input variables that have linear correlation with predicted output. In Section 4, three regression tree models were analyzed with a multiple linear regression model.

3.5 Particle Swarm Optimization

For PSO, the parameters described in Table 3.2 were adopted. PSO variant utilized is the one that implements constriction Clerk coefficient, as explained previously.

Tabela 3.2: PSO Parameters.

Parameter	Value
Cognitive Coeficient	2.05
Social Coeficient	2.05
Inertia Factor	0.8
Particles Number	30
Cycles Number	600

Each particle in PSO is a candidate configuration. The fitness function of particles is the RMSE of thirty evaluations of artificial neural network in case whose parameters are each particle. The parameters of the MLP, SVM and RBF networks are determined after the execution of this algorithm.

3.6 Artificial Neural Networks

In this section, it is described the settings of artificial neural networks in this study.

3.6.1 MLPs

Some variations of the MLP model were used in this study. Several parameters can be modified such as the number of hidden layers, number of hidden neurons in each hidden layer, learning rate, momentum, maximum number of training cycles and learning rule. A MLP model used in this study is shown in Figure 3.6. This model has are ten hidden neurons in a single hidden layer.

Hidden layers were single and double layer. The number of neurons in hidden(s) layer(s), the learning rate and momentum have been determined by an optimization algorithm such as PSO to increase the accuracy of errors estimation. For analyzes, the maximum number of training cycles was set to six hundred.

Learning rate, momentum and neurons in hidden layer varied from values presented in Table 3.3. A better parameters configuration solution is shown in Table **??**. Figure **??** presents MLP model with the better configuration for PERIL. The model contains ten neurons in hidden layer.

Tabela 3.3: Parameters intervals to MLP model.

Parameter	Min. Value	Max. Value
Momentum	0.1	0.9
Learning rate	0.1	0.9
Hidden Neurons	1	100

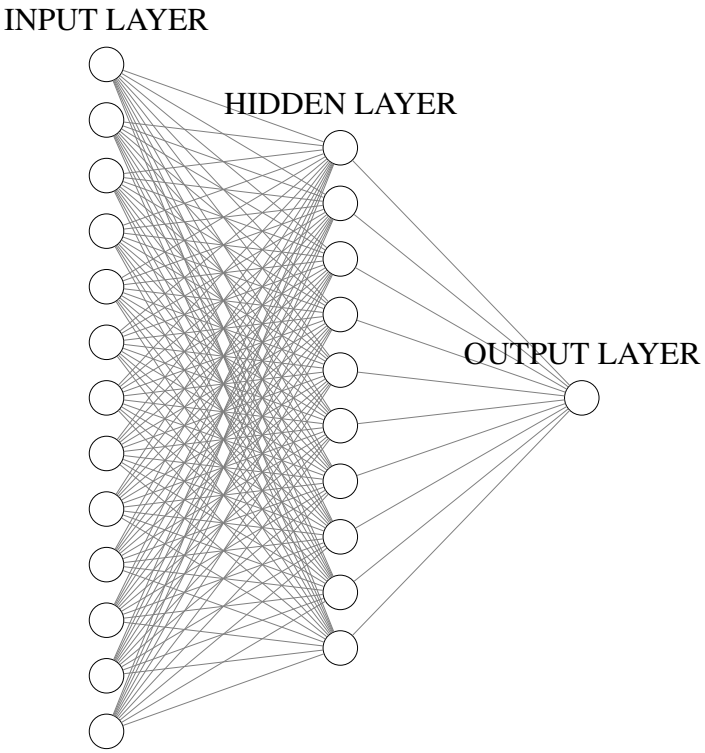

Figura 3.6: A MLP model utilized in this study.

Finally, the learning rules used in this study are Backpropagation, Levenberg-Marquardt, BFGS Quasi-Newton, Resilient Backpropagation, Polak-Ribiére Conjugate Gradient, Scaled Conjugate Gradient e One Step Secant.

In particular, a MLP called ”MLPRegressor”which has one hidden layer and whose learning rule aims to minimize the mean square error over a quadratic penalty through the BFGS Quasi-Newton method performed better than others variations.

3.6.2 SVM

SVM algorithm for regression used is SMOReg. In this algorithm RegSMOImproved is the optimization algorithm and PolyKernel is the kernel function as described in [68]. The pseudo-code for this algorithm is presented in Algorithm 3.6.2.

Algoritmo 1: Pseudocódigo do PSO

inicio
  inicializar as partículas no espaço de busca dispostas aleatoriamente;
  para cada *partícula* **hacer**
    extrair o *fitness* inicial;
    estabelecer seu $\vec{P}_{best}$ inicial;
  fin
  estabelecer o $\vec{G}_{best}$ inicial;
  enquanto *critério de parada não é atingido* **faça**
    para cada *partícula* **hacer**
      atualizar a velocidade da partícula;
      atualizar a posição da partícula;
      extrair o *fitness*;
      atualizar seu $\vec{P}_{best}$;
    fin
    atualizar o $\vec{G}_{best}$;
  fim
  retorna $\vec{G}_{best}$
fin

```
Begin
    peril <- read_file();
    peril_train <- partition(peril, 0, 50);
    peril_crossvalidation <- partition(peril, 50, 75);
    peril_test <- partition(peril, 75, 100);
    smo <- SMOReg();
    options <- [peril_train, peril_crossvalidation,
                RegSMOImproved, PolyKernel]
    SMOReg.runClassifier(smo, options);
    for instance in peril_test:
         calculated <- smo.classifyInstance(instance);
         wished <- instance.classValue();
         REMQ <- REMQ + (wished - calculated)^2
    end
    n <- peril_test.size();
    REMQ <- REMQ/n;
    REMQ <- sqrt(REMQ);
End
```

In Algorithm 3.6.2, data is read from a file, divided into training, cross validation and testing subsets. SMOReg regression model is instantiated, training is performed, calculated outcome is generated and the RMSE from the real and estimated output is calculated. This algorithm is used in fitness function for swarm optimization algorithm.

3.6.3 RBF

RBF neural network used is RBFRegressor, it minimizes the square error by BFGS method. The initial Gaussian centers are found using SimpleKMeans, which implements a K-Means algorithm. The initial sigma is set to the longest distance between any center and the nearest neighbor in the set of centers. The ridge parameter is used to penalize the size of the weights in the output layer, which implements a simple linear combination. The number of basis functions can also be specified. For this study only a global sigma is used for all basis functions. The pseudo-code for this algorithm is presented in Algorithm 3.6.3.

```
Begin
    peril <- read_file();
    peril_train <- partition(peril, 0, 50);
    peril_crossvalidation <- partition(peril, 50, 75);
    peril_test <- partition(peril, 75, 100);
    rbf <- RBFRegressor();
    options <- [peril_train, peril_crossvalidation, peril_test]
    RBFRegressor.runClassifier(rbf, options);
    for instance in peril_test:
         calculated <- rbf.classifyInstance(instance);
         wished <- instance.classValue();
         REMQ <- REMQ + (wished - calculated)^2
    end
    n <- peril_test.size();
    REMQ <- REMQ/n;
    REMQ <- sqrt(REMQ);
End
```

In Algorithm 3.6.3, data is read from a file, divided into training, cross validation and testing subsets. The instantiated regression model is SMOReg, training is overcame, calculated output is generated and RMSE is obtained from the real and estimated output. This algorithm is used in fitness function for swarm optimization algorithm.

3.6.4 ANFIS

ANFIS is a neuro-fuzzy system implemented in Matlab by Sugeno [62]. ANFIS uses a hybrid learning algorithm to identify parameters of Sugeno fuzzy inference system. It applies a combination of the method of least squares and gradient descent backpropagation for training the parameters of the membership function of the fuzzy inference system method. The fuzzy inference system used is genfis2, since there is a large number of input variables. The pseudo-code for this algorithm is presented in Algorithm 3.6.4.

```
Begin]
    inputs = csvread(peril,0,0,[0,0,648,10])
    targets = csvread(peril,0,11)
```

```
    tData = [inputs targets];
    in_fis = genfis2(inputs,targets, 0.7);
    trainOpts = [100,0.1,0.01,0.9,1.1]
    displayOpts = [1,1,1,1];
    chkData = []
    [fis,error,stepsize,chkFis,chkErr] =
         anfis(tData,in_fis,trainOpts,displayOpts,
         chkData,1);
    for err in error:
         REMQ <- REMQ + (err)^2
    end
    n <- peril.size();
    REMQ <- REMQ/n;
    REMQ <- sqrt(REMQ);
End
```

In Algorithm 3.6.2, data were read from file. The fuzzy inference system is instantiated, training is performed, error is generated and RMSE is obtained from the real and estimated outcome. This algorithm is used as fitness function for swarm optimization algorithm.

Capítulo 4

Experiments

The experiments performed utilized the models presented in Chapter 3, in which the database adopted is PERIL. They are established analysing regression models that are baseline for the study, accordingly to techniques usually utilized in academia and industry. After that, the state of art model is analyzed and, last but not least, this study concentrates in techniques proposed in work presented in Section 3.6. Results for each experiment are presented in Chapter 5.

General objective is to utilize the proposed method presented in Chapter 3 to determine a more precise approach to estimate risk impact. The adopted metric to verify the accomplishment of this objective is the prediction error, RMSE. A question to be answered is: Are ANNs more precise techniques to risk impact estimation in software project management?

Hypothesis for experiments are:

- H_0: There is no difference between using ANNs and state of art models to risk impact estimation;
- H_1: ANNs are more precise than state of art models for risk impact estimation;
- H_2: ANNs are less precise than state of art models for risk impact estimation.

Control objects are the source code developed for the experiment. Randomness, clustering and balancing criterias are used to facilitate statistical analysis. The experimental object is risk database, in our case PERIL. Finally, results are analyzed statistically, therefore a hypothesis test will be conducted as soon as results are obtained.

4.1 Multiple Linear Regression and Regression Tree Model

The first experiment set for this work is to establish a baseline method so that you can compare the performance of other approaches to the selected database. MRLM and MAR models are analyzed, the one which produces the smallest forecast errors (RMSE) will be selected.

In this analysis, the source code for the analysis of linear regression models were adapted from Torgo [37] in order to perform training, cross-validation, to obtain estimated outcomes and calculation of RMSE.

4.2 Monte Carlo Simulation and PERT Analysis

The second experiment is to analyze the performance of conventional techniques used in academia and industry, including best practices as determined by PMBOK [4]. As explained earlier, these approaches have been configured to obtain the best possible performance.

4.3 MultiLayer Perceptron and variations

The third experiment aims to analyze which of the variations of MLP obtains the lowest estimated RMSE. There are numerous possible combinations of MLP configurations, however only a subset of them was assessed. The best MLP configuration is used in the subsequent experiment.

This experiment and the next are the most significant experiments of this work. The importance of evaluating several alternatives to traditional MLP algorithm based on backpropagation is of fundamental importance since no previous work refined this study. Moreover, they did not investigate whether other approaches such as RBF and SVM could perform better.

4.4 MLP, SVM, RBF and ANFIS

The fourth experiment aims to elect the best technique based on Artificial Neural Networks to forecast the impact of hazards from the base PERIL database. The best settings for each approach is selected and RMSE is calculated for each technique.

4.5 Better Model Validation

Finally, a statistical test of the results of the best approach with those from the second experiment is performed to see if the model based on artificial neural network has higher precision than traditional ones. Validating the null hypothesis that artificial neural networks are more accurate and could meet the real needs of the industry, it is concluded that using the methodology proposed in this work it is possible to obtain a more precise RNA to estimate the impact of hazards.

Capítulo 5

Results

5.1 Multiple Linear Regression Model and Regression Tree Model

The first experiment set for this work is to establish a baseline so that you can compare the performance of alternative approaches to the selected database. Initially, this experiment consists of selecting among MLRM and RTM as a baseline. This can be done after discussing the information presented in Table 5.1 and Figure 5.1.

Table 5.1 shows descriptive statistics of normalized RMSE's to both algorithms. Average, standard deviation, minimum and maximum values are calculated for one MLRM and three RTM: $MLRM$ (cv.lm.v1), RTM_1 (cv.rpart.v1), RTM_2 (cv.rpart.v2) and RTM_3 (cv.rpart.v3).

Tabela 5.1: Descriptive statistics for normalized errors of linear regression models.

	$MLRM$	RTM_1	RTM_2	RTM_3
Average	**0.09912**	0.10238	0.10305	0.10361
Std Dev	**0.00391**	0.00423	0.00441	0.00426
Min.	**0.08956**	0.09214	0.09321	0.09372
Max.	**0.10746**	0.11231	0.11267	0.11359

In Figure 5.1, normalized RMSE's boxplots after predictions for RTM_3, RTM_2, RTM_1 and MLRM are presented. The minor RMSE is obtained in MLRM, this regression model also has minor standard deviation. From those information, we can affirm MLRM is a more efficient and precise model and will be introduced in the experiment as baseline method.

5.2 Monte Carlo Simulation and PERT Analysis

The second experiment is to analyze the performance of conventional techniques used in academia and industry, including best practices as determined by PMBOK [4]. As

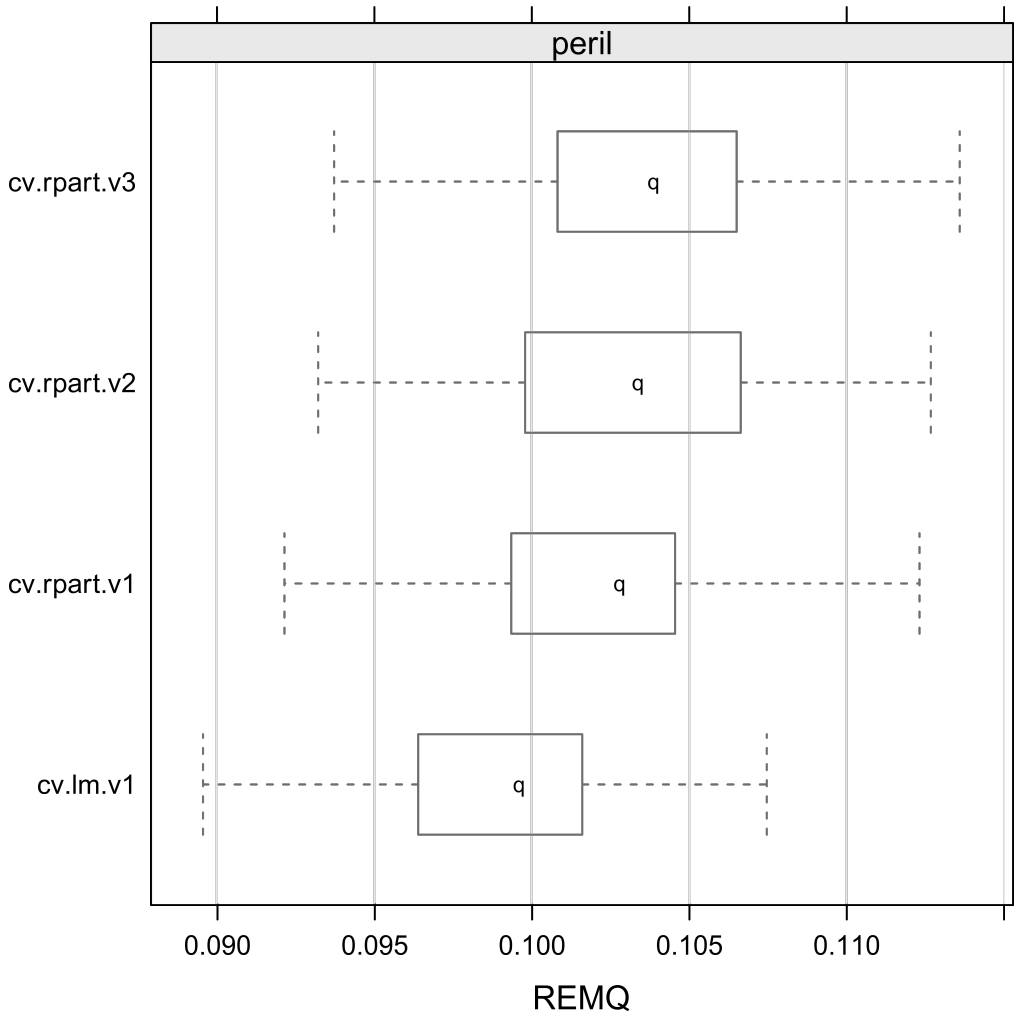

Figura 5.1: Boxplots for normalized errors of linear regression models.

explained earlier, these approaches have been configured to obtain the best possible performance.

Table 5.2 shows the descriptive statistics for normalized RMSE for both algorithms. The mean value, standard deviation, minimum and maximum, calculated for Monte Carlo Simulation and PERT Analysis, presents that the PERT Analysis has lower mean, minimum and maximum values. However, MCS has lower standard deviation, proving to be more precise between than.

In Figure 5.2, the boxplots of normalized RMSE after forecasts for MCS and PERT are presented. From this information, it can be stated that PERT Analysis is a more efficient state of art and slightly more precise, in this comparison.

PERT analysis proved to be a very interesting approach when is little risk information in learned lessons of previous projects, such as a risk register database. A detailed analysis using a database reveals that this technique has acceptable results to project

Tabela 5.2: Descriptive statistics for normalized errors of state of art models.

	MCS	PERT
Average	0.12640	**0.07466**
Std. Deviation	**0.01250**	0.01438
Minimum	0.10410	**0.04788**
Maximum	0.14950	**0.09122**

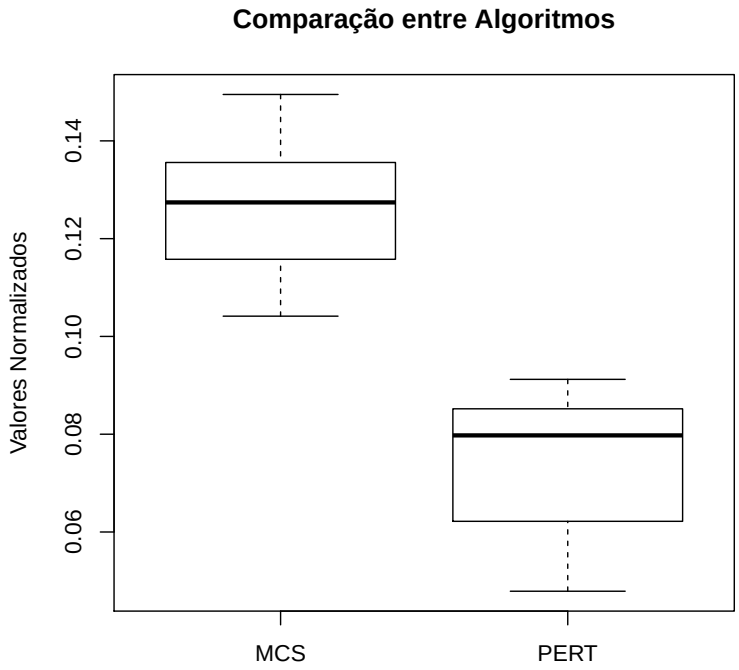

Figura 5.2: Boxplots for normalized errors of linear regression models.

managers.

5.3 MultiLayer Perceptron and variations

In this experiment, some MLPs were assessed. The main difference between than is the learning rule; Backpropagation, Levenberg-Marquardt, Broyden-Fletcher-Goldfarb-Shanno Backpropagation, Scaled Conjugate Gradient, Resilient-propagation, One-step Secant backpropagation and Quasi-Newton algorithms sre some of them selected.

Table 5.3 shows off descriptive statistics of normalized RMSE to approaches developed for this experiment. The mean, standard deviation, minimum and maximum values are calculated for each of the MLPs. ”BP”shows results for a MLP with backpropagation learning algorithm; ”LM”for an MLP with Levenberg-Marquardt algorithm; ”BFGS”for an MLP with Broyden-Fletcher-Goldfarb-Shanno Backpropagation algorithm; ”SCG”for an MLP with Scaled Conjugate Gradient algorithm; ”RP”for an MLP with Resilient-propagation algorithm; ”RPCG”for an MLP with Resilient-propagation

combined with Conjugate Gradient algorithm; ”OSS”for an MLP with One-step Secant backpropagation algorithm. Finally, ”Reg”for an MLP called ”MLPRegressor”with the BFGS Quasi-Newton algorithm. According to average, minimum and maximum values, it is observed that ”Reg”is a more efficient alternative, even with the largest standard deviation second this experiment.

Tabela 5.3: Descriptive statistics for normalized errors of MLPs models.

	BP	LM	BFGS	SCG	RP	RPCG	OSS	Reg
Avg	0.1000	0.0986	0.0980	0.0982	0.0979	0.0981	0.0995	**0.0516**
StD	0.0015	0.0018	**0.0011**	0.0018	0.0015	0.0021	0.0031	0.0042
Min	0.0973	0.0952	0.0945	0.0951	0.0946	0.0950	0.0943	**0.0427**
Max	0.1041	0.1035	0.1004	0.1037	0.1019	0.1041	0.1065	**0.0603**

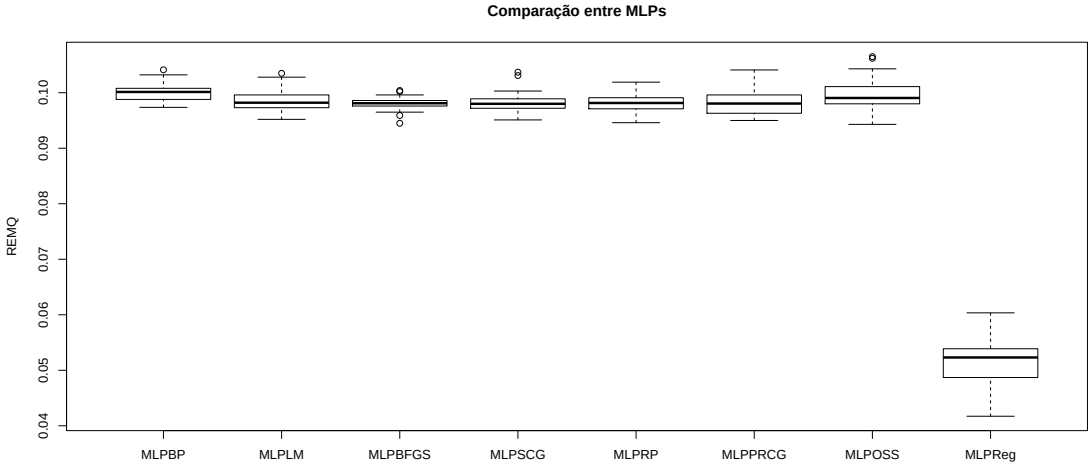

Figura 5.3: Boxplots for normalized errors of multiple linear perceptron networks.

In Figure 5.3, boxplots of normalized RMSE after forecasts for the eight MLPs are presented. From this information, it can be stated that a MLP called ”MLPRegressor”is a more efficient but still imprecise artificial neural network, according to experiment.

5.4 MLP, SVM, RBF and ANFIS

After evaluating an efficient neural network in the previous experiment, the fourth experiment aims to elect the best technique based on Artificial Neural Networks, among implemented for predicting risk impact from PERIL database.

Table 5.4 shows the descriptive statistics for the normalized RMSE to studied ANNs. The mean, standard deviation, minimum and maximum value calculated for a Neuro-Fuzzy model ANFIS, an SVM (called SMORegressor), an RBF network (called RBFRegressor) and for a ”MLPRegressor”show that the latter algorithm has much lower average, minimum and maximum values. However, the standard deviation of ANFIS

is lower. Therefore, ”MLPRegressor”proves to be a relatively efficient and accurate method compared with other techniques to risk impact estimation based on PERIL.

Tabela 5.4: Descriptive statistics for normalized errors of ANNs models.

	ANFIS	SMOReg	RBFReg	MLPReg
Avg	0.1079	0.09430	0.09004	**0.0516**
StD	**0.00003**	0.00488	0.00432	0.0042
Min	0.1078	0.08347	0.08024	**0.0427**
Max	0.1080	0.10284	0.09790	**0.0603**

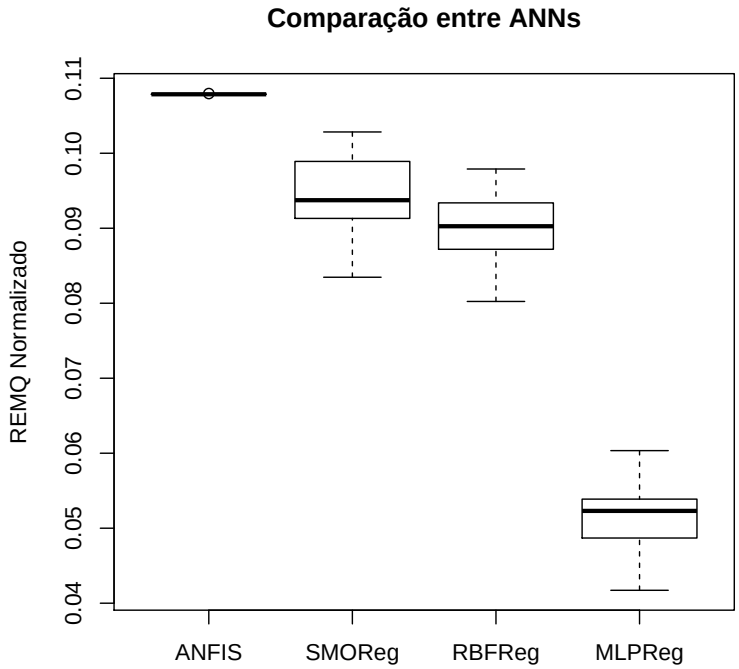

Figura 5.4: Boxplots for normalized errors of ANNs.

In Figure 5.4, boxplots of normalized RMSE after impact prediction of RNAs are shown. From this information, it can be stated that the ”MLPRegressor”is a more efficient Artificial Neural Network and slightly accurate, according to this comparison.

In Figure 5.5, the convergence of ANFIS is presented in terms of normalized resched for each season during training to be stopped, according to cross-validation. It is observed that the algorithm finds a local minimum and remains stuck until stopping the training. Therefore, it is concluded that the ANFIS has some limitations on convergence for this database.

In Figure 5.6, boxplots of expected and calculated impacts of fifth samples by ”SMORegressor”algorithm are presented. The ideal result is boxplots of both samples being as similar as possible, respecting median, interquartile range, minimum and maximum values. From the information presented, it can be concluded that boxplot

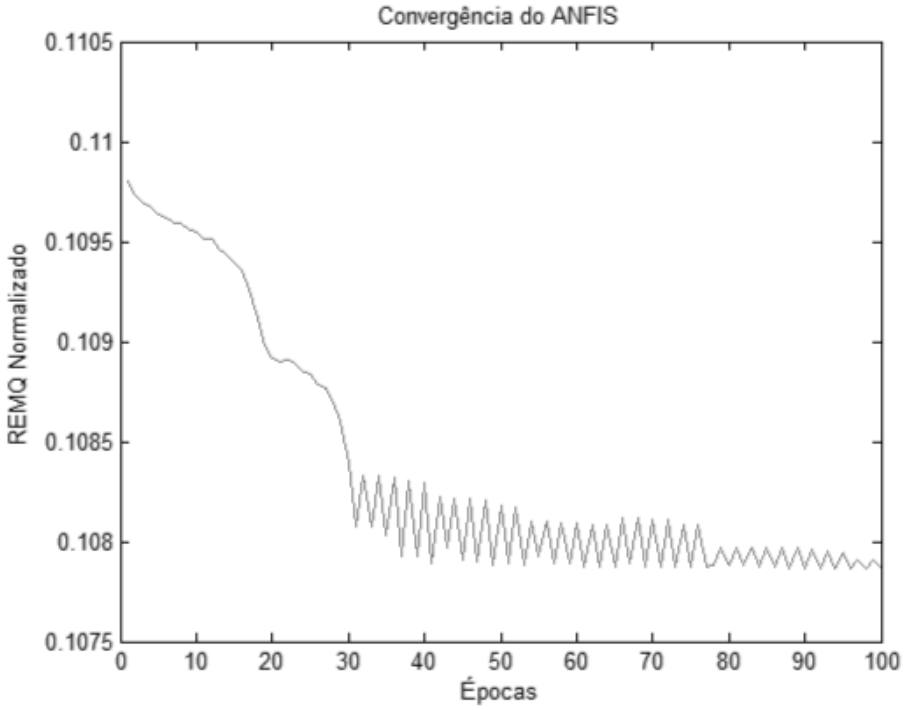

Figura 5.5: Convergence of ANFIS.

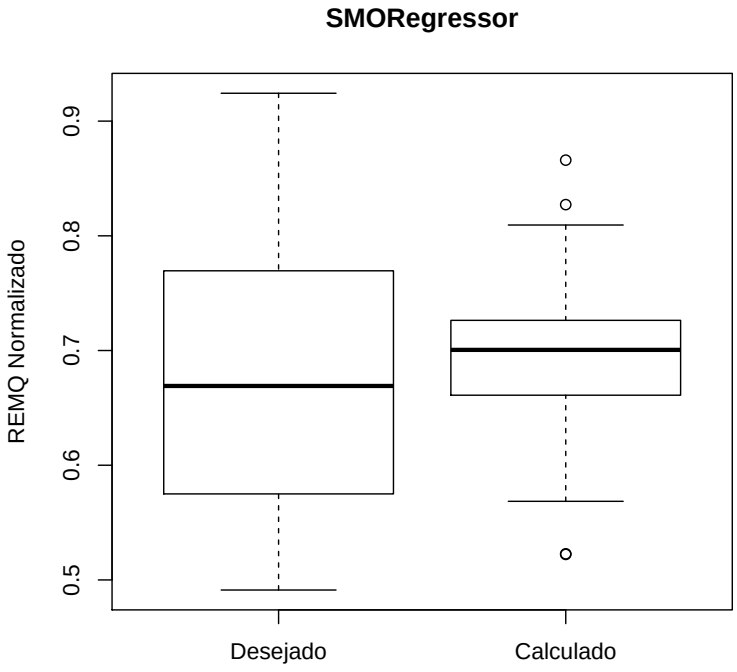

Figura 5.6: Boxplots for normalized errors of SVM.

coming from calculated impacts attempts to represent but distorts in all measures the expected impacts.

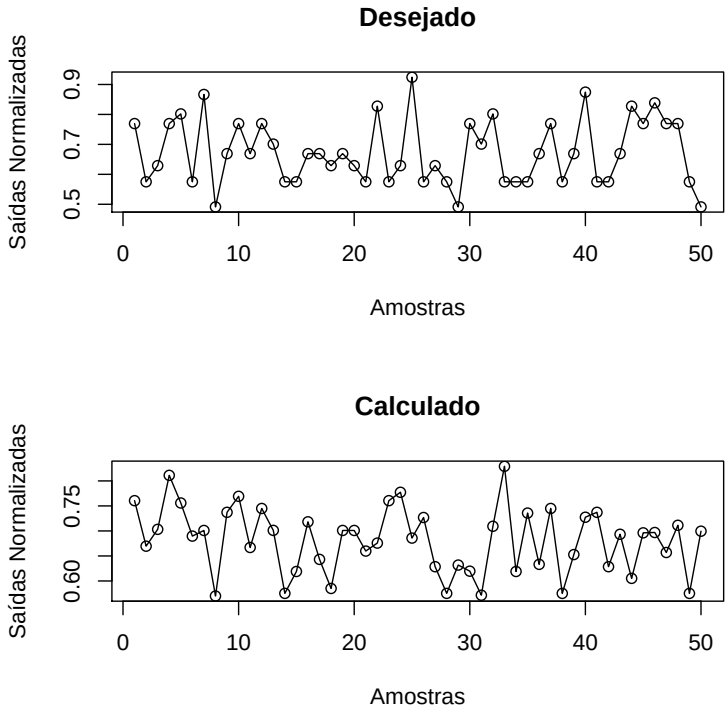

Figura 5.7: Normalized errors of expected and calculated impacts.

In Figure 5.7, the first fifty samples of expected and calculated outputs are presented graphically. From these graphs, it can be observed that the ”SMORegressor”algorithm has difficulty in estimating the expected results, showing almost no relationship with the last ones.

In Figure 5.8, the boxplots of expected and calculated impacts by ”RBFRegressor”of fifth samples are presented. The ideal result is that boxplots of both samples are as similar as possible, respecting the median, interquartile range, minimum and maximum values. From the information presented, it can be concluded that boxplots coming from the calculated impacts attempts but distorts to represent the lower quartile and maximum values of expected impacts.

In Figure 5.9, the first fifty samples of test subset and calculated outputs are presented graphically. From these graphs, it can be observed that the ”RBFRegressor”algorithm has difficulty in estimating the expected outcomes, mainly the maximum and minimum values.

In Figure 5.10, boxplots of expected impacts and calculated of fifty samples through ”MLPRegressor”algorithm are presented. The ideal result is that boxplots of the both samples to be as similar as possible, respecting the median, interquartile range, minimum and maximum values. From the information presented, it can be concluded that boxplot coming from the calculated impacts, but with distortions mainly in the lower quartile represents the expected impacts.

In Figure 5.11, the first fifty samples of tests subsets and from calculated outputs are presented graphically. From these graphs, it can be observed that ”MLPRegres-

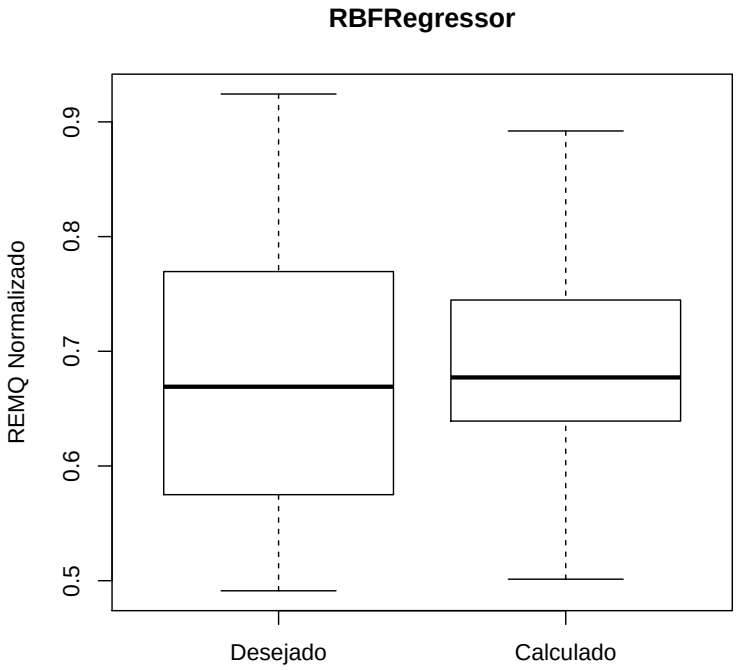

Figura 5.8: Boxplots for normalized errors of RBF networks.

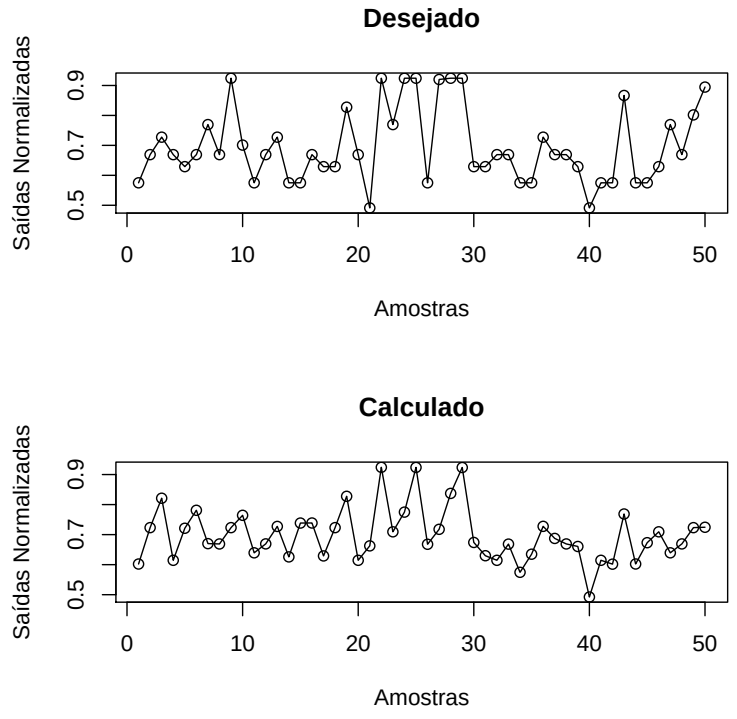

Figura 5.9: Normalized errors of expected and calculated impacts.

sor"algorithm has difficulty in estimating the expected results, but has approximate ten-

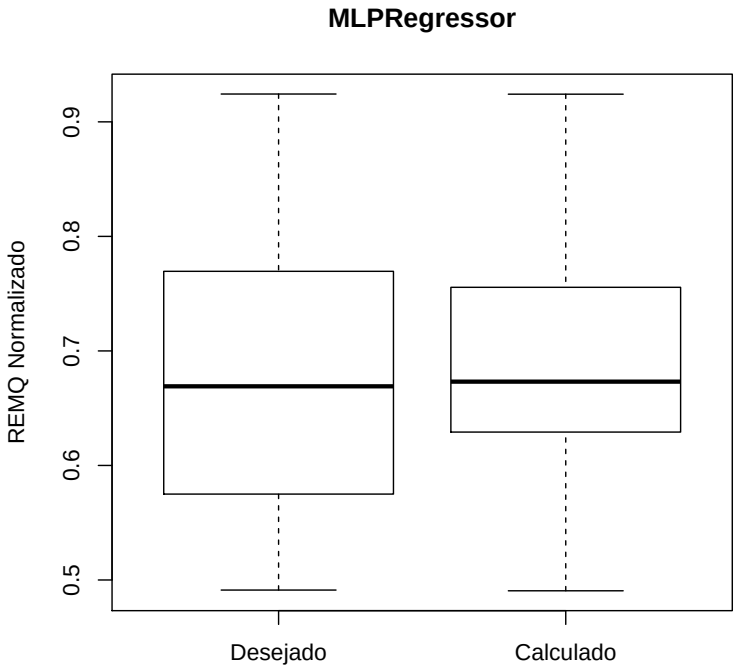

Figura 5.10: Boxplots for normalized errors of expected and calculated outcomes for MLPRegressor.

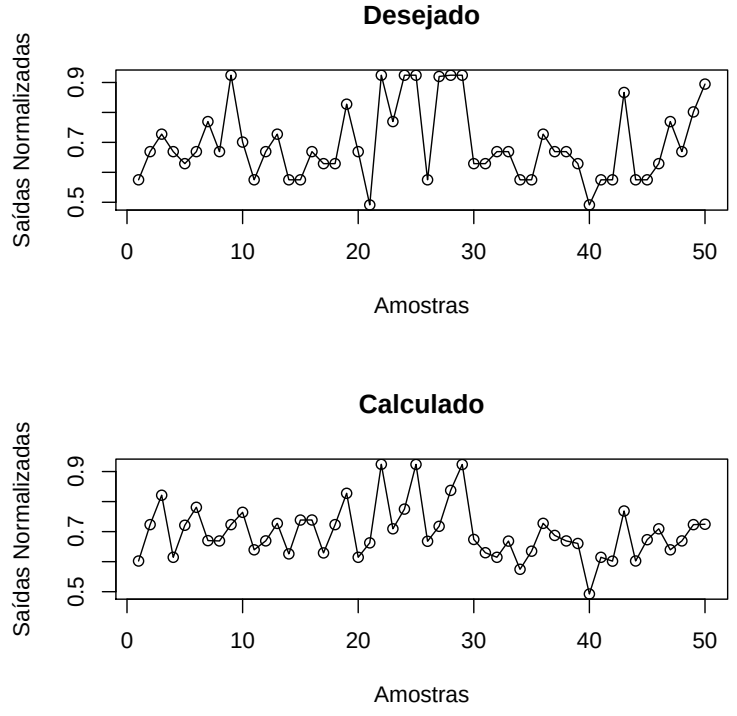

Figura 5.11: Normalized errors of expected and calculated impacts.

dency.

5.5 Better Model Validation

A Tabela 5.5 mostra a estatística descritiva do REMQ normalizado para as RNAs estudadas. Os valores de média, desvio padrão, mínimo e máximo, calculados para um MRLM, uma Simulação de Monte Carlo(SMC), uma Análise PERT(PERT) e para uma MLP "MLPRegressor"(MLPReg) mostram que o último algoritmo apresenta valores inferiores de média, desvio padrão, mínimo e máximo. Portanto, "MLPRegressor"prova ser um método mais eficiente e relativamente preciso, em comparação com as outras técnicas, para a estimativa do impacto de risco baseada na PERIL.

Tabela 5.5: Descriptive statistics for normalized errors of ANNs models.

	MLR	MCS	PERT	MLPReg
Avg	0.09912	0.12640	0.07466	**0.05168**
StD	0.00794	0.01250	0.01438	**0.00427**
Min	0.08956	0.10410	0.04788	**0.04172**
Max	0.10746	0.14950	0.09122	**0.06035**

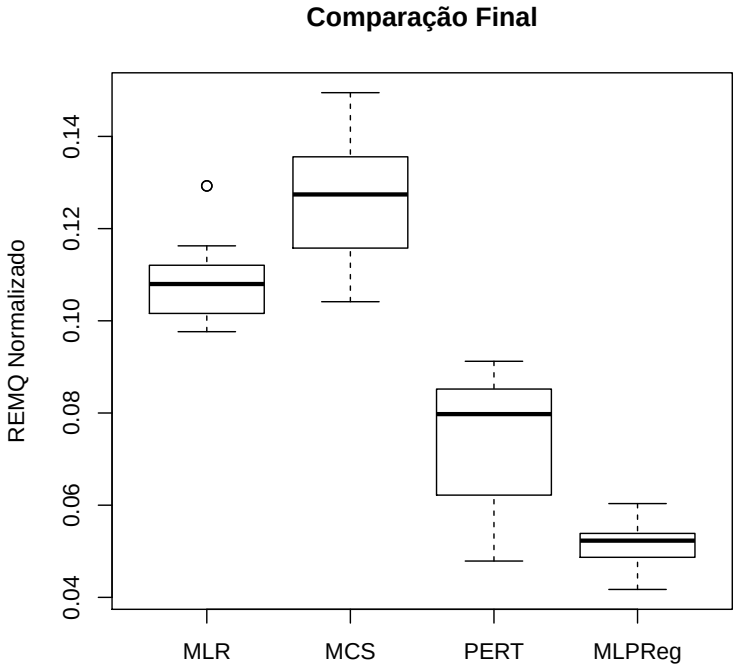

Figura 5.12: Boxplots for normalized errors of models in validation step.

In Figure 5.12, boxplots of normalized RMSE after the estimated impact of RNAs are shown. Monte Carlo Simulation (MCS) is immediately discarded from the analysis because it shows higher errors than Multiple Linear Regression Model (MLRM); PERT analysis (PERT) is presented as a simple and fast approach to estimate the risk impact, but as it is a statistical technique is subject to some situations that generate larger errors in the estimation; now, MLPRegressor (MLPReg) presents the lower mean, standard

deviation, minimum and maximum error values. From this information, it can be stated that the "MLPRegressor"is an Artificial Neural Network more efficient and accurate compared with the performance of other ANNs and state of art models.

Table **??** presents the results of non-paired Wilcoxon Tests for analyzed ANNs. The symbol Δ means that the technique on the left shows better results than the above; otherwise, the symbol ∇ simply means that the above technique presents better results than in the left. After analyzing the table, we can see that the "MLPRegressor"is better than the PERT Analysis that, in turn, is better than the Multiple Linear Regression Model.

Tabela 5.6: Hypothesis Tests for analyzed ANNs.

	MLR	MCS	PERT	MLPReg
MLR	-	Δ	∇	∇
MCS	∇	-	∇	∇
PERT	Δ	Δ	-	∇
MLPReg	Δ	Δ	Δ	-

5.6 Impact Estimation and Confidence Interval Definition

The last experiment consists in the practical used of the methodology proposed in this study, based on PERIL. As some steps described in the methodology were conducted in previous experiments, there remains only the generation of a confidence interval that is information understood by project managers and project and risk analysts.

The use of the confidence interval will determine the quality of the results generated. From the results shown by the model, will be generated a range that, for a given provision, will be 95 % chance of containing the true value. For this, maximum likelihood method will be used.

The maximum likelihood method considers that there are two sources of uncertainty in a forecast model. The first, σ_v is the noise variance, and the second, σ_w, is the variance of uncertainty. The σ_v is the variance of the errors generated by the set of cross-validation in the training phase. Those values follow a normal distribution and average to zero. But, σ_w refers to the variance of model uncertainty and is calculated from the use of the model to predict the errors generated by the network itself.

It is assumed that these two sources of error are independent. Then the calculation of the total variance for the model is given by Equation 5.1.

$$\sigma^2_{total} = \sigma^2_v + \sigma^2_w \tag{5.1}$$

No processo de validação cruzada do modelo, calcula-se o σ_v que é a variância dos erros. Para cada entrada do conjunto calcula-se o erro e ao final do processo a média desses erros e extrai-se sua variância usando a Equação 5.2.

$$\sigma_v = \frac{1}{n-1}\sum_{n}^{i=1}(Erro_i - \overline{Erro})^2 \tag{5.2}$$

where, n is the number of values; $Erro_i$ is the error correspondent to input i; $\overline{Erro}$ is the error average.

To calculate σ_w, errors are stored for all utilized inputs in training, including those in cross-validation subset. Since data are stored and normalized, a new model is obtained with new weights and links. This model will have as desired values the errors generated by the previous model. The σ_w will be calculated using the same formula used for the σ_v. The difference is the amount of data because, σ_w uses all errors of all subsets, not only the cross-validation.

Since σ_v and σ_w are calculated, one can calculate σ_{total}. This will be used to calculate the confidence interval from Equation 5.3.

$$x - t * \sigma_t otal < x < x + t * \sigma_t otal \tag{5.3}$$

where x is the computed outcome by the network and t is the value extracted from the Student's T table to the largest possible degree of freedom for a period of 95 % chance of containing the true value, as shown in table 5.1.

Tabela 5.7: A briefing of Student's T table.

$P(t_n \leq x)$				
n	0,750	0,900	0,950	0,975
30	0,683	1,310	1,697	2,042
40	0,681	1,303	1,684	2,021
60	0,679	1,296	1,671	2,000
120	0,677	1,289	1,658	1,980
∞	0,674	1,284	1,645	**1,960**

Capítulo 6

Conclusions and Future Works

This research has investigated the use of Artificial Neural Networks algorithms, like MLP, SVM and RBF, for risk impact estimation in software project risk analysis. We have carried out a statistical analysis using PERIL. The results were compared to MLRM, Monte Carlo Simulation, PERT analysis (proposed by [4]) and ANFIS System. We have considered improving risk impact estimation accuracy during software project management, in terms of RMSE mean and standard deviation. We have observed that a MLP alternative called "MLPRegressor"had minor standard deviation estimation error, and showed to be a promissory technique. Therefore, the selected ANN algorithms outperformed both linear regression and MCS.

Comentar sobre uma pesquisa rápida realizada com duzentos profissionais, os quais dez responderam às perguntas feitas e informaram que, no geral, entre 0% e 5% é o intervalo de erros ideal de previsão de impacto; 5% e 10% é um intervalo aceitável de erro na estimativa e 10% e 15% é um intervalo indesejado.

As perguntas realizadas foram:

1. What are the methods you utilize to risk analysis?
2. What are the advantages and disadvantages of those techniques in real-life problems?
3. Are they suitable for all your kind of projects?
4. Do you need historic risk management information (previous risk register) to risk analysis?
5. What is an acceptable estimation error percentage of risk impact? And what is considered an good estimation error percentage (10-15%, 5%-10%, 0%-5%)?

Portanto, resultados interessantes foram alcançados porém é preciso melhorar algumas técnicas utilizadas para que os resultados possam ser mais precisos e menores erros de previsão sejam gerados. Além disso, há uma carga computacional elevada no procedimento de utilização de um algoritmo de otimização para otimizar o desempenho das redes neurais, o que demanda tempo.

Como atividades futuras, foram identificadas.

- Realizar a validação prática dessa metodologia com informações reais para a estimativa e acompanhamento de riscos;
- Desenvolver uma abordagem eficiente e precisa para quando houver poucas informações sobre os riscos identificados num projeto e nenhum registro de riscos em projetos similares anteriores;
- Desenvolver uma abordagem inovadora e mais eficiente para a análise qualitativa dos riscos, baseada na classificação da natureza dos riscos;
- Desenvolver uma técnica para a avaliação de estratégias de mitigação de risco, baseadas no impacto se o risco ocorrer, no esforço para mitigação de risco, nas interações entre os fatores de risco e nos recursos disponíveis para os planos de mitigação;
- Desenvolver uma metodologia para a avaliação qualitativa, a avaliação quantitativa e planos de contingência de riscos em projetos, do ponto de vista do gerenciamento de portfólio de projetos para o alcance de objetivos estratégicos.
- Desenvolvimento de um *canvas* para o gerenciamento de riscos em projetos de forma ágil;

Referências Bibliográficas

[1] Alexander Budzier and Bent Flyvbjerg. Double whammy-how ict projects are fooled by randomness and screwed by political intent. *arXiv preprint arXiv:1304.4590*, 2013.

[2] Tom Kendrick. *Identifying and managing project risk: essential tools for failure-proofing your project*. Amacom: New York, 2003.

[3] Y. Y. Higuera, R. P. e Haimes. Software risk management, 1996.

[4] Project Management Institute. *A Guide to the Project Management Body of Knowledge*. 2008.

[5] The Standish Group. Chaos report. 2009.

[6] S. Islam. Software development risk management model - a goal driven approach. *Proc. Of ESEC FSE Doctoral Symposium 09 Amsterdam The Netherlands*, pages 5–8, 2009.

[7] Roy Schmidt, Kalle Lyytinen, Mark Keil, and Paul Cule. Identifying software project risks: an international delphi study. *Journal of management information systems*, 17(4):5–36, 2001.

[8] Paul L Bannerman. Risk and risk management in software projects: A reassessment. *Journal of Systems and Software*, 81(12):2118–2133, 2008.

[9] KPMG. Global it project management survey. 2005.

[10] L. Virine. Project risk analysis: How to make better choices in the uncertain times. *Proceedings of PMI Global Congress*, 2009.

[11] Riddle S. Keshlaf, A. Risk management for web and distributed software development projects. *The Fifth International Conference on Internet Monitoring and Protection*, 2010.

[12] Young Hoon Kwak and C William Ibbs. Calculating project management's return on investment. *Project Management Journal*, 31(2):38–47, 2000.

[13] B. W. Boehm. Software risk management: principle and practices. *IEEE Software*, 8:32–41, 1991.

[14] Yacov Y Haimes. *Risk modeling, assessment, and management*. John Wiley & Sons, 2011.

[15] Yacov Y(University of Virginia) Haimes. *Risk Modeling, Assesment and Management*. John Wiley & Sons, Inc., 3rd edition edition, 2009.

[16] T. Kendrick. *Identifying and Managing Project Risk: Essential Tools for Failure-Proofing your Project*. 2003.

[17] Ronald P Higuera and Yacov Y Haimes. Software risk management. Technical report, DTIC Document, 1996.

[18] Ibbotson Product Support. *Monte Carlo Simulation*, 2005. IbbotsonAssociates, 225 North Michigan Avenue Suite 700 Chicago, IL 60601-7676.

[19] Osamu Mizuno, Takuya Adachi, Tohru Kikuno, and Yasunari Takagi. On prediction of cost and duration for risky software projects based on risk questionnaire. In *Quality Software, 2001. Proceedings. Second Asia-Pacific Conference on*, pages 120–128. IEEE, 2001.

[20] Xishi Huang, Danny Ho, Jing Ren, and Luiz Fernando Capretz. A neuro-fuzzy tool for software estimation. In *Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on*, page 520. IEEE, 2004.

[21] Yong Hu, Jiaxing Huang, Juhua Chen, Mei Liu, and Kang Xie. Software project risk management modeling with neural network and support vector machine approaches. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 3, pages 358–362. IEEE, 2007.

[22] Iman Attarzadeh and Siew Hock Ow. A novel soft computing model to increase the accuracy of software development cost estimation. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 3, pages 603–607. IEEE, 2010.

[23] Dorota Dzega and Wieslaw Pietruszkiewicz. Classification and metaclassification in large scale data mining application for estimation of software projects. In *Cybernetic Intelligent Systems (CIS), 2010 IEEE 9th International Conference on*, pages 1–6. IEEE, 2010.

[24] PEI Yu. Software project risk assessment model based on fuzzy theory. *Computer Knowledge and Technology*, 16:049, 2011.

[25] Urvashi Rahul Saxena and SP Singh. Software effort estimation using neuro-fuzzy approach. In *Software Engineering (CONSEG), 2012 CSI Sixth International Conference on*, pages 1–6. IEEE, 2012.

[26] Zhang Dan. Improving the accuracy in software effort estimation: Using artificial neural network model based on particle swarm optimization. In *Service Operations and Logistics, and Informatics (SOLI), 2013 IEEE International Conference on*, pages 180–185. IEEE, 2013.

[27] Beatrice Lazzerini and Lusine Mkrtchyan. Analyzing risk impact factors using extended fuzzy cognitive maps. *Systems Journal, IEEE*, 5(2):288–297, 2011.

[28] A. Osundahunsi. Effective project risk management using the concept of risk velocity, agility and resiliency. *CAMERON International, Houston, TX, USA*, page 13, 2012.

[29] Ray C Williams, George J Pandelios, and Sandra G Behrens. *Software Risk Evaluation (SRE) Method Description: Version 2.0*. Carnegie Mellon University, Software Engineering Institute, 1999.

[30] Chris Chapman and Stephen Ward. *Project risk management: processes, techniques and insights*. John Wiley, 1996.

[31] Richard Fairley. Risk management for software projects. *Software, IEEE*, 11(3):57–67, 1994.

[32] Kakoli Bandyopadhyay, Peter P Mykytyn, and Kathleen Mykytyn. A framework for integrated risk management in information technology. *Management Decision*, 37(5):437–445, 1999.

[33] Vered Holzmann and Israel Spiegler. Developing risk breakdown structure for information technology organizations. *International Journal of Project Management*, 29(5):537–546, 2011.

[34] Project Management Institute. Practice standard for project risk management. 2009.

[35] Young Hoon Kwak and Lisa Ingall. Exploring monte carlo simulation applications for project management. *Risk Management*, 9(1):44–57, 2007.

[36] Wayne D Cottrell. Simplified program evaluation and review technique (pert). *Journal of construction Engineering and Management*, 125(1):16–22, 1999.

[37] Luis Torgo. Data mining with r. *Learning by case studies. University of Porto, LIACC-FEP. URL: http://www. liacc. up. pt/ltorgo/DataMiningWithR/. Accessed on*, 7(09), 2003.

[38] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[39] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[40] Lee C.S.G. Lin, C.T. Neural fuzzy systems: A neuro-fuzzy synergism to intelligent systems. 1996.

[41] M. J. S. Valença. Aplicando redes neurais: um guia completo. *Livro Rápido, Olinda–PE*, 2005.

[42] Pitts W. MCCulloch, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophisics*, pages 115–133, 1943.

[43] Donald O Hebb. *The organization of behavior: A neuropsychological theory*. New York: Wiley, 1949.

[44] Bernard WIDROW, Marcian E HOFF, et al. Adaptive switching circuits. 1960.

[45] Frank Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.

[46] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974.

[47] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[48] Andries P Engelbrecht. *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[49] S. Haykin. *Redes Neurais: Princípios e Práticas*. 2007.

[50] M. J. S. Valença. *Fundamentos das Redes Neurais - Exemplos em JAVA*. 2a edicao edition.

[51] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[52] Huang J. Chen J. Liu M. Xie K. Hu, Y. Software project risk management modeling with neural network and support vector machine approaches. *Third International Conference on Natural Computation (ICNC)*, 2007.

[53] V Vapnik. The nature of statistical learning theory. *Data Mining and Knowledge Discovery*, pages 1–47, 6.

[54] Vladimir N Vapnik. Statistical learning theory (adaptive and learning systems for signal processing, communications and control series), 1998.

[55] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

[56] Mohamad T Musavi, Wahid Ahmed, Khue Hiang Chan, Kathleen B Faris, and Donald M Hummels. On the training of radial basis function classifiers. *Neural networks*, 5(4):595–603, 1992.

[57] Sheng Chen, CFN Cowan, and PM Grant. Orthogonal least squares learning algorithm for radial basis function networks. *Neural Networks, IEEE Transactions on*, 2(2):302–309, 1991.

[58] Russell Beale and Tom Jackson. *Neural Computing-an introduction*. CRC Press, 2010.

[59] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[60] LM Saini and MK Soni. Artificial neural network based peak load forecasting using levenberg-marquardt and quasi-newton methods. In *Generation, Transmission and Distribution, IEE Proceedings-*, volume 149, pages 578–584. IET, 2002.

[61] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[62] J.S.R. Jang, C.T. Sun, and E. Mizutani. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. MATLAB curriculum series. Prentice Hall, 1997.

[63] Sidney Siegel. Nonparametric statistics for the behavioral sciences. 1956.

[64] S Amari, Noboru Murata, Klaus-Robert Müller, Michael Finke, and H Yang. Statistical theory of overtraining-is cross-validation asymptotically effective? *Advances in neural information processing systems*, pages 176–182, 1996.

[65] Shun-ichi Amari, Andrzej Cichocki, Howard Hua Yang, et al. A new learning algorithm for blind signal separation. *Advances in neural information processing systems*, pages 757–763, 1996.

[66] N. Taleb. *Fooled by randomness*. Nwe York: Random House, 2001.

[67] Kevin L Priddy and Paul E Keller. *Artificial Neural Networks: An introduction*, volume 68. SPIE Press, 2005.

[68] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, and K.R.K. Murthy. Improvements to the smo algorithm for svm regression. In *IEEE Transactions on Neural Networks*, 1999.