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

Carlos Henrique Maciel Sobral Timóteo

Uma Metodologia para a Análise Quantitativa de Riscos no Gerenciamento de Projetos de Software

Dissertação de Mestrado

Recife, Março 2014

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

Carlos Henrique Maciel Sobral Timóteo

Uma Metodologia para a Análise Quantitativa de Riscos no Gerenciamento de Projetos de 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, Março de 2014

REVIWER 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.

Michael Wittke
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 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. 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 defines the experiments to be undertaken. 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

Após uma revisão sistemática da literatura, foram identificadas inúmeras abordagens para a análise de risco: Modelo de Regressão Logística (MRL), Rede de Crenças Bayesiana (RCB), Redes Neurais Artificiais (RNA), Análise de Discriminantes (AD), Árvore de Decisão (ADE), Algoritmos Genéticos (AG), Otimização por Enxame de Partículas (PSO), Teoria dos Conjuntos Fuzzy (TCF), Sistemas Neuro-Fuzzy (SNF), Mapas Cognitivos Fuzzy Estendidos (E-FCM) [19] [20] [21] [22] [23] [24] [25] [26].

Hu et al. [21] propuseram um método para análise de riscos de software e previsão do resultado de projetos de software. Algoritmos Genéticos foram utilizados como um método de otimização do desempenho de Redes Neurais por meio da seleção dos pesos, da estrutura da rede e da regra de aprendizado. A RNA padrão, nesse estudo, teve uma melhoria no desempenho após a inclusão de AG. Os resultados dos experimentos mostraram que após a introdução de AG para o processo de treinamento da RNA, o modelo de avaliação de risco de software modificado pôde ser sensivelmente melhorado alcançando uma maior precisão quando comparado com o modelo SVM. Zhang Dan [26] propôs um modelo de previsão baseado em RNA que utiliza o Modelo de Custo Construtivo (COCOMO) que foi aprimorado após a aplicação do PSO, para prover um método de estimativa do esforço no desenvolvimento de software de forma precisa. Attarzadeh e Ow [22] utilizaram a RNA para melhorar a precisão da estimativa do esforço comparado com o modelo tradicional COCOMO. Huang et al. [20] apresentaram um *framework* genérico para a estimativa de software baseado em SNF e melhoraram a estimativa de custo para o COCOMO'81.

Yu [24] apresentou um modelo baseado na Teoria Fuzzy. Ele superou a dificuldade da avaliação de indicadores qualitativos e quantitativos nos métodos de análise tradicionais. Além disso, Saxena e Singh [25] exploraram técnicas Neuro-Fuzzy no projeto de um modelo adequado na utilização de uma melhor estimativa de esforço no desenvolvimento de software para projetos da NASA. Os resultados mostraram que o SNF tem o menor erro de previsão quando comparado com modelos existentes. Por outro lado, Lazzerini e Mkrtchyan [27] sugeriram um *framework* para análise de riscos usando E-FCM e E-FCM Estendidos, pela introdução de uma representação gráfica especial para

análise de risco.

Mizuno et al. [19] propuseram um novo método de previsão para projetos de software arriscados. Os autores utilizaram o modelo de regressão logística para estimar se um projeto pode tornar-se arriscado ou não. No entanto, a abordagem de previsão proposta para o custo e a duração não tem um nível alto de precisão.

Dzega e Pietruszkiewicz [23] apresentaram resultados de experimentos de análise de riscos realizados usando classificadores de mineração de dados tais como C4.5, RandomTree e Árvore de Regressão e Classificação (CART). Além disso, eles descreveram como os metaclassificadores *boosting* e *bagging* foram aplicados para melhoria dos resultados e também analisaram a influência de seus parâmetros em habilidades de generalização para a precisão da estimativa. Devido a um grande número de atributos desordenados na base de dados, MLP e SVM foram rejeitados prematuramente, gerando baixa precisão para cada conjunto de dados.

Em resumo, alguns desses estudos propuseram métodos para a estimativa de custo, cronograma e esforço de projetos de software; outros estudos propuseram abordagens para classificação de risco e de projetos de software (sucesso, desafiado ou falho); os demais apresentaram técnicas para estimativa do impacto do risco na gestão de projetos de software [24] [25] [27] [23].

2.2 Project Risk Management

Risco pode ser definido como um evento incerto ou condição que, se ocorrer, afeta pelo menos um dos objetivos do projeto. Ele pode ser considerado tanto como uma ameaça (impacto negativo) quanto uma oportunidade (impacto positivo) [4].

O Gerenciamento de Riscos envolve processos de planejamento, identificação, avaliação e priorização dos mesmos. Numa definição mais apropriada, gerenciamento de riscos pode ser definido como o processo de análise do grau de exposição ao risco e na determinação de como melhor lidar com essa exposição. Essa área objetiva não somente a identificação, como também o desenvolvimento de uma abordagem robusta para gerenciar proativamente o impacto dos riscos no projeto [28].

De acordo com o PMBOK [4], o gerenciamento de riscos em projetos envolvem processos relativos ao planejamento, identificação, análise, planejamento de respostas, monitoramento e controle de riscos de um projeto. Seu objetivo é aumentar a probabilidade e o impacto dos eventos positivos e reduzir a probabilidade e severidade dos eventos negativos. Do ponto de vista de gerenciamento, a tomada de decisões conscientes pela avaliação do que pode dar errado, assim como a probabilidade e a gravidade do impacto, é o cerne do gerenciamento de riscos. Essa atividade envolve a avaliação das vantagens e desvantagens associadas a todas as alternativas propostas para mitigação de riscos em termos de seus custos, benefícios, riscos e da avaliação do impacto das decisões atuais sobre as alternativas futuras.

Um resumo dos processos no gerenciamento de riscos é definido a seguir. Seis são os tópicos incluídos nos grupos de atividades de planejamento, monitoramento e controle.

- Planejar o gerenciamento dos riscos: o processo de definição da condução das atividades de gerenciamento de risco num projeto;

- Identificar riscos: o processo de determinação dos riscos que possam afetar o projeto e a documentação de suas características;
- Realizar a análise qualitativa dos riscos: o processo de priorização dos riscos para análise ou ações adicionais através da avaliação e combinação de suas probabilidades de ocorrência e impacto;
- Realizar a análise quantitativa dos riscos: o processo de análise numérica do efeito de riscos identificados previamente, em termos dos objetivos gerais do projeto;
- Planejar respostas aos riscos: o processo de desenvolvimento de opções e ações para aumento das oportunidades e diminuição das ameaças aos objetivos do projeto;
- Monitorar e controlar os riscos: o processo de implementação do planejamento de respostas aos riscos, rastreamento de riscos identificados, monitoramento dos riscos residuais, identificação de novos riscos e avaliação da eficácia do processo de tratamento de risco durante todo o projeto.

O Software Engineering Institute(SEI) desenvolveu uma metodologia de gerenciamento de risco chamada *Software Risk Evaluation*(SRE) que é especificamente voltada para projetos na indústria de software. O paradigma SRE é composto por seis elementos: cinco módulos (identificação, análise, planejamento, acompanhamento e controle) e um elemento central (comunicação), que é a chave para a efetiva gestão de riscos [3] [29].

Boehm [13], Chapman [30], Fairley [31], Bandyopadhyay et al. [32] apresentaram métodos e modelos diferentes de gerenciamento de riscos em projetos de software. No entanto, há elementos em comum em todas as abordagens anteriores: a identificação, a avaliação da probabilidade e impacto, e o planejamento a respostas para manuseio desses riscos se eles ocorrerem [33].

O gerenciamento de risco, no sentido amplo, é útil para organizações que têm um portfólio de projetos. Esse gerenciamento foca principalmente no risco agregado para a tomada de decisões como interromper, abortar e continuar com a realização do projeto. No entanto, na perspectiva de um líder de projetos, há apenas projetos isolados, em que o enfoque está centrado em cada uma das atividades para cada projeto. Segundo Kendrick [2], o gerenciamento de risco de projetos deve focar em riscos no sentido restrito.

O gerenciamento de risco de projetos, no sentido restrito, serve para melhorar as chances de cada projeto individual alcançar o sucesso. Na maioria dos outros campos, o gerenciamento de risco está preocupado principalmente com os valores médios de um grande número de eventos independentes. Para o gerenciamento do risco de projetos, no entanto, o que geralmente mais importa é a previsibilidade - gerenciar a variação esperada no resultado para o projeto específico [2].

Os objetivos da gestão de risco para um único projeto são estabelecer um plano crível, consistente com os objetivos de negócio, e na minimização do intervalo de resultados possíveis. Quanto maior o intervalo de duração possível para um projeto mais elevado o seu risco. O risco do projeto aumenta com o nível de incerteza, tanto negativa quanto positiva [2].

O gerenciamento do risco de projetos utiliza dois parâmetros fundamentais de risco: probabilidade e perda. Probabilidade é geralmente a possibilidade de um evento ocorrer - como é frequentemente obtida através de suposição, logo pode ser bastante imprecisa. Perda é geralmente designado em projetos como ”impacto”, e baseia-se nas consequências para o projeto no caso de ocorrência do risco. Impacto é geralmente medido em tempo ou custo, particularmente para avaliação quantitativa de riscos [2].

O principal benefício do gerenciamento de riscos em projetos é tanto o desenvolvimento de uma base de credibilidade para cada projeto, mostrando que o mesmo é possível, quanto da demonstração da inviabilidade do projeto, mostrando que o mesmo deva ser evitado, abortado ou transformado. A análise de risco pode também revelar oportunidades para melhoria dos projetos que podem resultar em altos retornos no investimento. A análise de riscos adequada reduz tanto o custo total quanto a frustração causada por problemas evitáveis. A quantidade de retrabalho e atraso imprevisto de esforço no projeto é reduzida. Além disso, a análise de risco revela fraquezas num plano de projeto e desencadeia mudanças, novas atividades e realocação de recursos que melhoram o resultado do projeto [2].

2.2.1 Qualitative Risk Analysis

O processo de análise qualitativa avalia as características dos riscos de projetos identificados individualmente e prioriza-os baseado nas requisições estabelecidas para o projeto. Em outras palavras, a análise qualitativa avalia a probabilidade de cada evento ocorrer e o efeito individual de cada um deles nos objetivos do projeto. Como tal processo não aborda diretamente o risco global para os objetivos do projeto, que resultam do efeito combinado dos eventos individuais e as interações entre si, então isso pode ser obtido através do uso de técnicas de análise quantitativa [34].

2.2.2 Quantitative Risk Analysis

Análise é a conversão de dados de risco em informação para tomada de decisão. A análise fornece a base para o gerente de projetos trabalhar nos riscos certos e mais críticos. Boehm [13] define o objetivo da análise de risco como a avaliação da probabilidade e magnitude de perda para cada item de risco identificado, e ele avalia os riscos compostos das interações dos itens de risco. Técnicas típicas incluem modelos de desempenho e custo, análise de rede, análise de decisão estatística e análise de fatores de qualidade (tais como confiabilidade, disponibilidade e segurança).

A análise de risco depende de um bom mecanismo de identificação de riscos. No entanto, a maioria dos métodos assume que os gerentes devam ter a experiência necessária a respeito de todos os fatores de risco pertinentes, o que nem sempre corresponde a realidade. Além disso, muitos desses métodos podem ser demorados e, portanto, muito dispendiosos para se utilizar de forma regular. Portanto, um método popular para a identificação de fatores de risco tem sido a utilização de listas de verificação. Infelizmente, essas listas de verificação são baseadas em pequenas amostras ou, ainda pior, viciadas pelos seus métodos de coleta de dados históricos de risco.

As técnicas mais utilizadas para análise de risco incluem [4]:

- Análise de Sensibilidade: ajuda a determinar quais riscos têm os maiores impactos potenciais no projeto. Uma representação típica disso é o diagrama de tornado (explicar isso);

- Valor Monetário Agregado(EMV): é um conceito estatístico que calcula o valor médio do impacto do risco agregado quando o futuro inclui cenários que podem ou não ocorrer (ou seja, sob a análise da incerteza). Um uso comum desse tipo de técnica é a análise da árvore de decisão;

- Modelagem e simulação: simulação utiliza um modelo que converte as incertezas especificadas e detalhadas em seu potencial impacto sobre os objetivos do projeto. As simulações iterativas gerais são realizadas usando Simulação de Monte Carlo;

- Opinião especializada: opinião especializada (de especialistas com experiência relevante e recente) é necessária não apenas para identificação dos impactos potenciais sobre o custo e o cronograma e para avaliação da probabilidade, como também para definir as variáveis de entrada e quais ferramentas utilizar.

O objetivo da análise quantitativa de riscos é criar um ”perfil do risco”do projeto. Para tanto, são necessárias as seguintes informações: a chance de o projeto ser finalizado dentro de um certo período de tempo ou orçamento; a taxa de sucesso de projetos; as estimativas de pior caso, médio e melhor caso e outros parâmetros do projeto [4].

Alguns trabalhos propõem novas ferramentas de análise quantitativa de riscos. Entre eles, Virine [10] apresenta a metodologia da Cadeia de Eventos. Nesse trabalho, as atividades de um projeto não são um procedimento uniforme e contínuo. Essas tarefas são afetadas por eventos externos, que transformam as atividades de um evento para outro. O momento em que os eventos externos ocorrem são probabilísticos e podem ser definidos utilizando uma distribuição estatística. Além disso, eventos podem causar outros eventos, criando, portanto, a Cadeia de Eventos. A análise dessas combinações é realizada através da Simulação de Monte Carlo.

A análise de risco demonstra a incerteza dos resultados do projeto e é útil para justificar reservas de cronograma e/ou recursos. É mais apropriado para definir uma janela de tempo (ou orçamento) em vez de um objetivo único para projetos arriscados. Por exemplo, o cronograma alvo para um projeto arriscado pode ser doze meses, mas o cronograma empenhado, refletindo problemas potenciais, pode ser fixado em quatorze meses. A conclusão no prazo (ou antes) desse intervalo define um projeto de sucesso; somente se o projeto demorar mais de quatorze meses será considerado um fracasso [2].

2.3 Conventional Techniques for Risk Analysis

2.3.1 Monte Carlo Simulation

A Simulação de Monte Carlo é uma técnica que computa ou itera o custo ou cronograma do projeto muitas vezes usando valores de entrada selecionados aleatoriamente a partir de distribuições de probabilidades de custos ou durações possíveis, para calcular uma distribuição dos custos totais ou datas de finalização do projeto [4].

Um modelo é desenvolvido, e ele contém algumas variáveis de entrada. Essas variáveis de entrada tem resultados possíveis diferentes, representados por uma função de distribuição de probabilidade de valores para cada variável. O método de Monte Carlo é uma abordagem de simulação através de computação intensiva para determinar a probabilidade de resultados possíveis de um objetivo do projeto, tais como a data de finalização ou o custo total. As entradas do procedimento são obtidas aleatoriamente a partir de intervalos específicos com funções de distribuição de probabilidade para as durações das atividades no cronograma ou itens da linha de base de custo. Esses diferentes valores de entrada são utilizados não apenas para construir um histograma de resultados possíveis para o projeto e sua probabilidade relativa, mas também a probabilidade cumulativa para calcular as reservas de contingências desejadas para o cronograma ou custo. Resultados adicionais incluem a importância relativa de cada entrada para determinar o custo e o cronograma geral para o projeto [35].

O desenvolvimento desse método, especialmente o uso de computadores para realizar os cálculos, foi creditado a Stanislaw Ulam, um matemático que trabalhou no Projeto Americano de Manhattan durante a Segunda Guerra Mundial. O seu trabalho com Jon von Neuman e Nicholas Metropolis transformaram a amostragem estatística de uma curiosidade matemática para uma metodologia formal aplicável a uma grande variedade de problemas [35].

A Simulação de Monte Carlo é uma abordagem detalhada de simulação através de computação intensiva para determinar a probabilidade de resultados possíveis de um objetivo do projeto; por exemplo, a data de conclusão ou o custo total. As entradas para o procedimento são obtidas aleatoriamente a partir de intervalos específicos com funções de distribuição de probabilidade para as durações das atividades do cronograma ou itens da linha de custo. Esses valores de entrada diferentes são usados para construir um histograma de possíveis resultados do projeto e sua probabilidade relativa, como também, a probabilidade cumulativa para calcular as reservas de contingência desejadas de tempo ou custo. Resultados adicionais incluem a importância relativa de cada entrada na determinação do custo geral do projeto e cronograma. Um exemplo de resultados de estimativa de riscos de cronograma e custo são apresentados na Figura 2.1 [34].

Na Figura 2.1, observa-se a previsão de finalização para um cronograma de um projeto. No eixo horizontal, encontram-se as possíveis datas de finalização do cronograma, a frequência de ocorrência dessas datas na amostragem são representadas pelas barras verticais, a frequência cumulativa é representada pela curva escura e pelos respectivos valores na borda direita do gráfico. A partir da frequência relativa, é possível prever a probabilidade do projeto finalizar até determinada data, nesse exemplo, esse projeto teve 90% de chance de finalizar até 08 de Maio de 2009.

2.3.2 PERT Analysis

PERT foi criada pela Marinha dos EUA em 1958 como uma ferramenta para programar o desenvolvimento de um sistema de armamento por completo. A técnica considera o projeto sendo uma rede acíclica de eventos e atividades. A duração de um projeto é determinada por um plano de fluxo do sistema no qual a duração de cada atividade tem um valor esperado e uma variância. O caminho crítico inclui uma sequência de

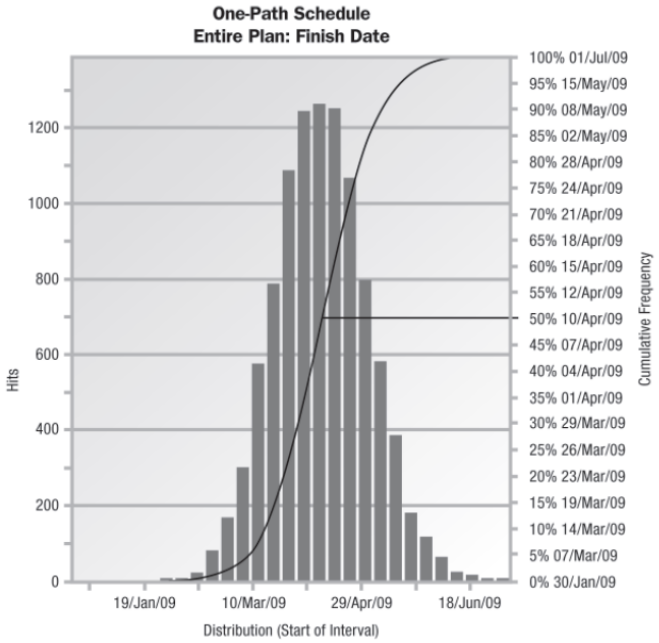

Figura 2.1: Exemplo de Resultado da Simulação Monte Carlo

atividades que não podem ser adiadas, sem ameaçar o projeto por inteiro. PERT pode ser usado para estimar a probabilidade de completar um projeto ou atividades individuais por qualquer período de tempo especificado. É também possível determinar a duração do intervalo de tempo correspondente a uma dada probabilidade [36].

O primeiro passo na aplicação do PERT é desenvolver diagramas da rede do projeto, em que cada arco representa uma atividade e cada nó simboliza um evento (como o início ou conclusão de uma tarefa). Alternativamente, cada nó pode simbolizar uma atividade. O segundo passo é designar três estimativas de tempo para cada tarefa: otimista (a), pessimista (b) e mais provável (m). Pequenas probabilidades estão associadas com a e b.No diagrama original, a é a duração mínima de uma atividade; a probabilidade de uma duração mais curta é zero. Da mesma forma, b é a duração máxima; a probabilidade de uma duração menor ou igual a b é 100%. Nenhuma suposição é feita sobre a posição de m relativo a a e b. Em termos estatísticos, a e b são os extremos de uma distribuição hipotética de tempos de duração. A moda da distribuição é m. Para acomodar a flexibilidade nas posições destes parâmetros, a distribuição beta é usada. A distribuição beta é útil para a descrição de dados empíricos e pode ser simétrica ou enviesada [36].

O terceiro passo é calcular o valor esperado e variância da duração de cada atividade no diagrama de rede do projeto. A média de uma distribuição beta é uma equação cúbica. A equação para essa média, Equação 2.1, é uma aproximação linear dessa forma:

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

onde τ_e é a duração estimada de uma atividade, m é igual à moda, que ocorre quando a e b são simétricos com relação a m.

Em distribuições de probabilidade unimodais, o desvio-padrão da distribuição é igual a cerca de um sexto do intervalo. Com 100% das durações possíveis vinculadas a a e b, a variação estimada da duração é como se segue:

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

onde σ^2 é a variância da duração da atividade. Uma alternativa, apresentada em [36], é definir a e b como os limiares de 5% e 95% do intervalo, respectivamente. Então, a variância é como se segue:

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

O quarto passo é ordenar as atividades em seqüência, do início ao fim do projeto, em um formato tabular, listando as durações otimista, pessimista, mais prováveis, esperadas e as variâncias. Em quinto lugar, os passos *forward* e *backward* através da rede são realizadas para identificar o caminho crítico, tal como no método do caminho crítico amplamente utilizado. O teorema limite central é então aplicado como se segue: a distribuição da soma das durações previstas das atividades ao longo do caminho crítico é aproximadamente a distribuição normal, particularmente a medida que o número de atividades aumenta. A duração esperada de cada soma é igual à soma das durações esperadas. Do mesmo modo, a variação de cada soma é a soma das variâncias [36].

Essas aplicações do teorema do limite central permitir o cálculo de probabilidades de duração do projeto usando os desvios a partir de uma média zero da variável normal (Z). Essas probabilidades podem ser cruciais na tomada de decisões financeiras quanto à viabilidade de um projeto [36].

2.4 Statistical and Intelligent Computing Techniques for Risk Analysis

Nesta seção, serão explorados inicialmente dois modelos de previsão que poderiam ser aplicados para o domínio de análise de risco: regressão linear múltipla e modelos de árvore de regressão. A escolha não foi conseqüência de alguma etapa de seleção formal de modelos, contudo, ainda assim, os modelos apresentam-se como duas boas alternativas para problemas de regressão como são bastante diferentes em termos de suas suposições sobre a ”forma”da função de regressão sendo aproximada e pela facilidade de interpretar e velocidade para rodar em qualquer computador. O objetivo da escolha desses dois modelos de regressão comuns está relacionado com a validação do desempenho de outras técnicas. Por exemplo, se uma técnica de previsão tem um desempenho pior do que modelos de regressão múltipla linear, como de árvores de regressão, então parece ser uma alternativa inadequada.

2.4.1 Multiple Linear Regression

A Regressão Linear Múltipla está entre as técnicas de análise de dados estatísticos mais utilizadas. Esse modelo obtém uma função aditiva relacionando uma variável de saída

a um conjunto de variáveis de entrada. Essa função aditiva é a soma de elementos da fórmula $\beta_i \times X_i$, onde X_i é uma matriz estimadora de variáveis e β_i é a matriz de pesos na regressão linear múltipla [37]

Na regressão linear, os dados são modelados para caber numa linha reta. Por exemplo, uma variável aleatória Y, chamada variável de resposta, pode ser modelada como uma função linear de uma variável aleatória X, chamada de variável de previsão com a Equação 2.4:

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

onde a variância de Y é assumida ser constante. Os coeficientes α e β (chamados coeficientes de regressão) especificam o interceptação de Y e a inclinação da linha, respectivamente. Esses coeficientes podem ser resolvidos pelo método dos mínimos quadrados, que minimiza o erro entre a linha real separando os dados e a estimativa da linha. Dado s amostras ou pontos de dados da forma $(x_1, y_1), (x_2, y_2), ..., (x_s, y_s)$, então os coeficientes de regressão podem ser estimados usando esse método por meio das Equações 2.5 and 2.6:

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

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

onde x é a média de $x_1, x_2, ..., x_s$, e y a média de $y_1, y_2, ..., y_s$. Os coeficientes α e β frequentemente provêm boas aproximações mesmo para equações de regressão complexas. A regressão múltipla é uma extensão da regressão linear, permitindo uma resposta variável Y para ser modelada como uma função linear de um vetor de características multidimensional [38].

2.4.2 Regression Tree Model

Uma árvore de regressão é uma hierarquia de testes lógicos em algumas das variáveis explanatórias do modelo. Logo, nem todas as variáveis aparecem na árvore. Uma árvore é lida do nó-raiz para os nós-folha. Cada nó da árvore tem dois galhos. Eles estão relacionados ao valor de um teste em cada uma das variáveis de previsão. O teste continua até um nó-folha ser alcançado. Nesses nós-folha tem-se os valores previstos pela árvore. Isso significa que se é necessário usar uma árvore para obter-se uma estimativa para uma amostra particular, é necessário seguir um galho do nó-raiz até um nó-folha, de acordo com os resultados do teste para a amostra. O valor médio da variável alvo encontrado no nó-folha alcançado é a previsão da árvore [37] [39].

A Figura 2.2 apresenta o modelo de árvore de regressão para a base de dados do PERIL. Partindo do nó-raiz é possível verificar dois galhos que mostram testes lógicos para a variável *Date0*. Os próximos nós representam resultados possíveis levando-se em conta somente *Date0*. O segundo nível da árvore mostra testes lógicos associados

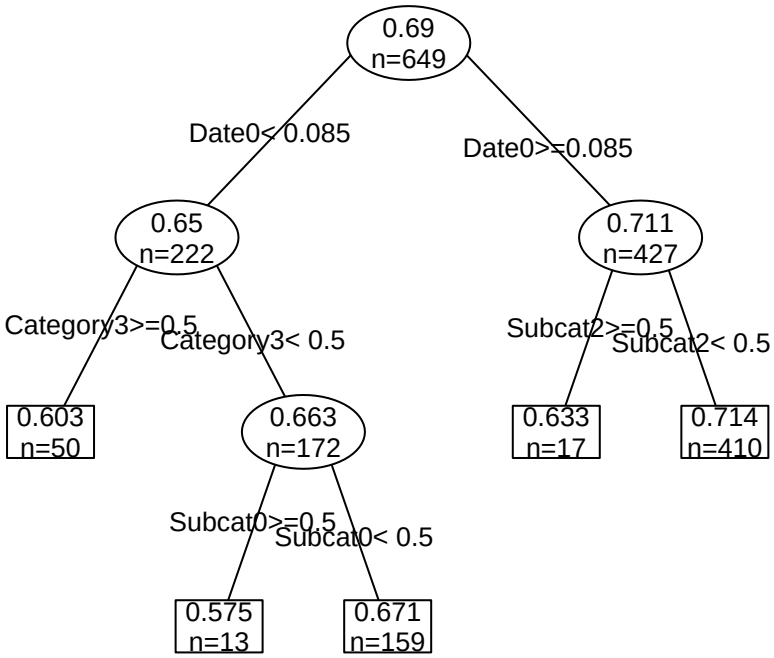

Figura 2.2: Regression Tree Model for PERIL

às variáveis *Category3* e *Subcat2*. Em seguida, encontram-se três nós-folha. Cada valor do nó-folha representa os possíveis resultados e o número de amostras no PERIL, que satisfazem os testes lógicos conjugados nos níveis anteriores da árvore. Por último, testes lógicos para a variável *Subcat0* são apresentados para geração de outros dois nós-folha.

Árvores são obtidas geralmente em dois passos. Inicialmente, uma grande árvore é desenvolvida, e em seguida essa árvore é podada eliminando os nós inferiores através de um processo de previsão estatística. Esse processo tem o objetivo de evitar o *overfitting*. Isso tem a ver com o fato de que uma árvore larga demais irá memorizar os dados do treinamento quase perfeitamente, mas irá capturar relações espúrias do conjunto de dados oferecido (*overfitting*), e portanto irá ter um desempenho ruim quando exposta a uma nova amostra de dados para a qual previsões são necessárias. O problema de *overfitting* ocorre em muitas técnicas de modelagem, particularmente quando as suposições

acerca da função a ser aproximada têm menor importância. Esses modelos, mesmo que tenham uma ampla faixa de aplicação (devido a essas condições não prioritárias), sofrem com esse problema de *overfitting*, necessitando, portanto, de uma previsão baseada em estatística para diminuir o impacto desse efeito [37].

2.5 Artificial Neural Networks

Uma Rede Neural Artificial (RNA) é um processador distribuído maciçamente paralelo constituído por unidades de processamento simples, que tem a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso [40]. Ela adota estimativas de regressão não-paramétricas constituída de elementos de processamento interconectados entre dados de entrada e saída, e tem uma excelente capacidade de aprendizado e generalização.

Outra definição descreve uma RNA como um sistema constituído de elementos de processamento, chamados neurônios, os quais estão dispostos em camadas (uma camada de entrada, uma ou várias camadas intermediárias e uma camada de saída) e são responsáveis pela não-linearidade e pela memória da rede [41].

As RNA são modelos que procuram simular o comportamento e o funcionamento do cérebro humano. Assim como existem neurônios biológicos, componentes essenciais para o processamento das informações do cérebro, a RNA é formada por unidades que objetivam realizar as mesmas funções do neurônio biológico. Esses componentes são denominados neurônios artificiais e foram propostos em 1943 por Mc-Culloch e Pitts [42].

Em seguida ao trabalho de McCulloch e Pitts surgiu a regra de aprendizagem proposta por Donald Hebb [43] que se constitui a base de todas as regras de aprendizagem. Em seu famoso livro, o autor procurou encontrar um mecanismo neural capaz de explicar como as informações podiam ser armazenadas e recuperadas nos neurônios. A regra de aprendizagem era enunciada da seguinte forma: "Quando um neurônio recebe um estímulo de outro neurônio, e se ambos estão altamente ativos, o peso entre eles deve ser fortalecido, caso contrário enfraquecido". A propósito, esse neurônio é chamado de Perceptron. Em 1960, Widrow e Hoff [44] apresentaram uma regra de aprendizagem para uma extensão do Perceptron, desenvolvida previamente por Frank Rosenblatt [45], chamada de ADALINE (Adaptive Linear Neuron). Esta regra baseada no método dos mínimos quadrados ficou conhecida como regra delta. Paul Werbos em 1974 [46] desenvolveu o algoritmo de *backpropagation*, sendo esse algortimo posteriormente popularizado através da publicação feita por Rumelhart e McClelland em 1985 [47].

O modelo do neurônio de McCulloch e Pitts procura representar o neurônio biológico utilizando uma regra de propagação e uma função de ativação. Considere $x_1, x_2, x_3, ..., x_n$, como sendo as variáveis de entrada x_j, em que $j = 1, ..., n$ do neurônio de saída i. A entrada líquida net_i é dada pela seguinte regra de propagação:

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

onde, w_{ij} são os pesos sinápticos e θ é o limiar de ativação do neurônio.

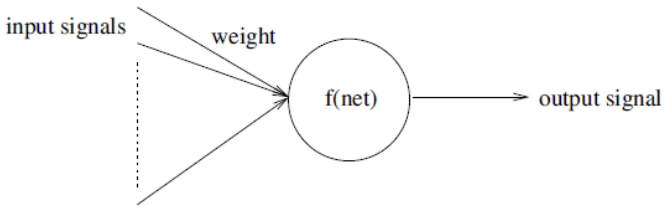

Figura 2.3: Representação gráfica da MLP com três camadas

A saída y_i é dada por $f(net_i)$, em que f é a função de ativação. Existem várias funções de ativação propostas, dentre elas as mais comuns são: linear, degrau, rampa, sigmoide logística, tangente hiperbólica e gaussiana representadas respectivamente por [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

A rede perceptron de múltiplas camadas é uma generalização da rede perceptron simples pela adição de pelo menos uma camada intermediária, conhecida como camada escondida. Em uma rede em camadas, os neurônios estão dispostos em cada uma delas. Na MLP, a primeira delas é a camada de entrada, na qual as variáveis de entrada são conectadas diretamente a um neurônio, exclusivamente. A próxima camada, denominada intermediária, liga completamente os neurônios da camada anterior aos neurônios da camada de saída. Por fim, a camada de saída representa a saída da RNA. Cada entrada em um neurônio tem um peso a ele associado a ser ajustado pelo algoritmo de treinamento. Um modelo comum de MLP contém um neurônio de *bias*, representando o parâmetro independente de variáveis para a função de ativação. A MLP é um grafo direto, no qual as entradas de dados são propagadas a partir da camada de entrada para a*s* camada*s* escondida*s* e da*s* camada*s* escondida*s* para a camada de saída. O fluxo de

dados no caminho para frente numa MLP é conhecido como ”fase *forward*”. O fluxo de dados na direção oposta é a ”fase *backward*”.

Uma das principais preocupações da ANN é o dilema da estabilidade-plasticidade, no qual, embora a aprendizagem contínua seja desejada numa RNA, a aprendizagem futura fará com que a RNA perca sua memória quando os pesos alcançaram um estado de equilíbrio [**?**]. O algoritmo *Backpropagation*, a ser definido adiante, é comumente usado como o método de treinamento, pois nos permite ajustar os pesos da rede de múltiplas camadas, através da Regra Delta Generalizada [47].

A vantagem de ter camadas intermediárias é que a rede neural passa a resolver problemas que não são linearmente separáveis, possibilitando, assim, a aproximação de qualquer função contínua, com apenas uma camada, e qualquer função matemática, quando houver mais de uma camada [49]. A Figura 2.4 ilustra a forma gráfica da MLP, apresentando as entradas, saídas e as camadas de entrada, intermediária e de saída.

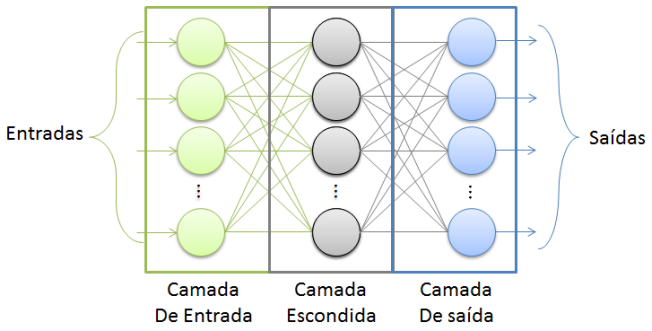

Figura 2.4: Representação gráfica da MLP com três camadas

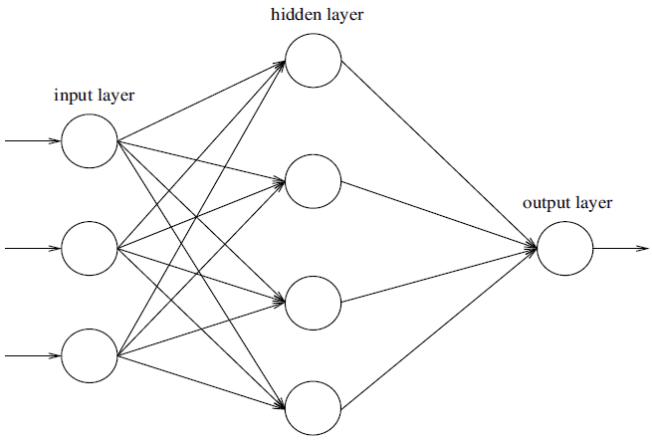

Figura 2.5: Representação gráfica da MLP com três camadas

O funcionamento da MLP é dividido em duas fases: *forward* e *backward*. Na fase *forward*, um neurônio de uma camada está ligado a todos os neurônios da camada seguinte. Os sinais da entrada são propagados da camada de entrada para a camada escondida e da camada escondida para a camada de saída; cada neurônio processa as entradas e apresenta uma saída. Nessa fase é possível calcular o erro entre a saída desejada para

a rede e a saída apresentada pela MLP. Na fase *backward* o erro é retropropagado e os pesos são ajustados, utilizando o algoritmo de ajustes de pesos, inicialmente aleatórios, chamado *Backpropagation* [41].

Backpropagation Algorithm

A primeira etapa do Backpropagation consiste na inicialização dos pesos com valores aleatórios. A partir daí, para cada exemplo da base de treinamento é realizado o seguinte processo: a propagação do sinal; a fase *forward* e retropropagação do erro; e a fase *backward*. Na fase *forward* é possível calcular o erro entre a saída desejada para a rede e a saída apresentada pela MLP. Na fase *backward* o erro é retropropagado e os pesos são ajustados. Faz-se necessário calcular a sensibilidade para cada neurônio. A sensibilidade para as camadas é dada por:

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

em que, w_{ij}^m são os pesos sinápticos que serão ajustados, δ_i^m representa a sensibilidade, o índice i representa o número de neurônios da camada que recebe o sinal e possui $N_{neurons}$ e $f^{m-1'}(net_j^{m-1})$ é a derivada da função de ativação dos neurônios da camada "M-1"(camada que emite o sinal) em relação à entrada líquida, net_j^{m-1}.

O ajuste dos pesos é dado por:

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

em que α é a taxa de aprendizagem do algoritmo, β é a taxa de momento e $\Delta w_{ij}^m(t-1)$ é a correção realizada no peso w_{ij}^m na iteração $t-1$. Quanto maior o valor de α maiores serão as correções realizadas a cada iteração. Já o momento β tem o objetivo de deixar o algoritmo menos susceptível a ficar preso em mínimos locais (no algoritmo original,$\beta = 0$), devido a superfície da função erro ser bastante complexa.

Durante o treinamento os exemplos vão sendo apresentados à MLP de forma aleatória e os pesos vão sendo ajustados pelo algoritmo *backprogation* até que se chegue a uma condição de parada. A parada do treinamento é uma decisão crítica, pois uma parada prematura do treinamento poderá acarretar o *underfitting* (os pesos da MLP não são ajustados de forma satisfatória), e caso o treinamento se torne excessivamente longo, pode ocorrer o *overfitting* (a rede memoriza os exemplos de treinamento e ocorre perda na capacidade de generalização) [50]. Objetivando evitar essas duas condições, e consequentemente uma melhor capacidade de generalização, a base de dados para treinamento da rede pode ser dividida em duas partes: uma parte de fato para treinamento e uma outra parte para validação. A parte para treinamento é apresentada à rede e serve para realizar o ajuste dos pesos pelo algoritmo de treinamento. A parte correspondente ao conjunto de validação serve para realizar uma avaliação do treinamento da rede. Quando o erro médio quadrático para o conjunto de validação começar a aumentar, significa que a MLP está memorizando o conjunto de treinamento e neste momento é adequado interromper o treinamento.

As redes MLP são uma boa ferramenta para a construção de classificadores, já que além de conseguir lidar com o mapeamento entrada-saída não linear, apresentam alto poder de generalização [51].

Em [52], algumas técnicas inteligentes são utilizadas para a análise do risco de projetos de software. O artigo conclui que uma técnica híbrida de redes neurais e algoritmos genéticos obtém uma precisão de 85% na previsão do projeto obter sucesso, ser desafiado ou falhar totalmente.

2.5.2 Support Vector Machine

A Máquina de Vetor de Suporte, do inglês *Support Vector Machine* (SVM), é uma técnica de aprendizado de máquina aplicável a problemas de reconhecimento de padrões nos quais se busca atingir alto potencial de generalização [49] [41]. O objetivo da SVM é encontrar um hiperplano particular, denominado de hiperplano ótimo, que maximize a margem de separação, conforme pode ser visualizado na Figura 2.6. Na Figura 2.6, observamos dois vetores de suporte capazes de separar linearmente as saídas no hiperplano em duas classes. A variável r é a distância algébrica desejada dos vetores de suporte para o hiperplano ótimo de separação das classes. Quanto maior essa distância, maior a capacidade de generalização da máquina de vetor de suporte.

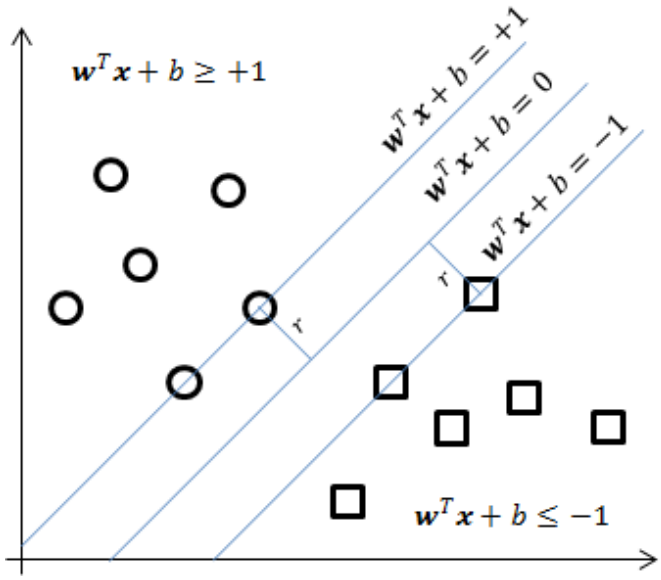

Figura 2.6: Vetores de Suporte e Hiperplano de Separação ótimo

As SVMs foram desenvolvidas tendo por base a teoria do aprendizado estatístico, detalhada por Vapnik em 1995 e 1998 [53] [54] e inicialmente proposta por Vapnik e Chervonenkis em 1971 [55].

O desenvolvimento das SVMs teve por objetivo obter algoritmos com alta capacidade de generalização. De acordo com a teoria do aprendizado estatístico este erro de generalização chamado de Risco Funcional pode ser calculado caso se conheça a

distribuição de probabilidade da população [54]. Entretanto, para problemas de aprendizado supervisionado, em geral não se dispõe da distribuição de probabilidade da população,uma vez que em problemas reais o que se utiliza é uma amostra da população sendo possível apenas o cálculo de uma função erro, como por exemplo o erro médio quadrático. Esta função erro calculada da amostra de dados é chamada de Risco Empírico. Logo, a depender do tamanho da amostra utilizada para treinamento, a minimização do erro de treinamento, não significa necessariamente uma minimização do erro de generalização ou Risco Funcional.

Um conceito importante da teoria do aprendizado estatístico é o da dimensão VC (Vapnik-Chervonenkis). A dimensão VC é um índice escalar que mede a complexidade intrínseca de uma classe de funções. Uma definição para a dimensão VC é de que esta representa o número máximo de exemplos de treinamento que uma máquina de aprendizagem é capaz de classificar corretamente, para todas as possíveis combinações binárias desses dados.

Desta forma, de acordo com a teoria do aprendizado estatístico proposta por Vapnik [54], uma rede neural ou qualquer outra máquina de aprendizagem durante o aprendizado supervisionado obtém sua capacidade máxima de generalização quando durante o treinamento se minimiza o Risco Funcional. A minimização do Risco Funcional é equivalente a minimização do Risco Empírico e do Risco Estrutural associado a complexidade do modelo. O Risco Funcional é limitado, com probabilidade $1-\delta$ pela Equação 2.16.

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

O Risco Estrutural por sua vez pode ser calculado, conforme mostrado por Vapnik [**?**] como:

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

onde h é a dimensão VC.

A formulação inicial das SVMs para problemas linearmente separáveis foi chamada de SVMs de margens rígidas (máximas). A diferença destas para uma rede *perceptron* está na forma de seleção do hiperplano de separação das classes. Enquanto numa rede *perceptron* se busca qualquer hiperplano que satisfaça o problema, numa SVM de margem rígida se procura o hiperplano ótimo, ou seja, aquele cuja margem de separação é máxima.

Desta forma, a determinação dos pesos das SVMs se tranforma em um problema de otimização com restrição, o que exige um maior esforço computacional. A otimização dos pesos das SVMs com restrição de margem máxima tem sido resolvida através do método de multiplicadores de Lagrange.

Posteriormente, foram elaboradas as SVMs de margens flexíveis (suaves) com o objetivo de lidar com exemplos de treinamento com erros ou ruídos e finalmente as SVMs não lineares que utilizam na camada escondida funções de base diferentes.

Para um melhor entendimento das SVMs considere um problema de classificação linear com vetor de entrada x_i $(i = 1, ..., n)$, onde n é o número de exemplos e saídas y_i(+1 ou -1). A seleção dos parâmetros deve ocorrer de tal forma que:

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

ou de forma resumida

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

A margem (d) é a distância entre os dois planos paralelos e pode ser expressa como uma distância euclidiana, na Equação 2.21.

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

onde, $||w|| = \sqrt{w_1^2 + w_2^2 + ... + w_n^2}$ é a norma Euclidiana do vetor de pesos.

Portanto, maximizar a margem se transforma em um problema de otimização com restrição para minimizar a seguinte função objetivo $\min_w \frac{||w||^2}{2}$ sujeito a Equação 2.20.

Esse é um problema de otimização não-linear (a função objetivo é quadrática) com restrições lineares que pode ser resolvido por meio dos multiplicadores de Lagrange.

2.5.3 Radial Basis Function Network

As Redes de Função de Base Radial *Radial Basis Function* (RBF) surgiram em 1988 como uma possível alternativa às redes MLP. Na sua formulação tradicional é composta por três camadas: uma camada de entrada, uma camada escondida e uma camada de saída. As principais diferenças entre as redes RBF e MLP são:

1. Os neurônios da camada intermediária têm apenas funções de base radial como função de ativação, que são funções localizadas de tal maneira que apenas algumas unidades escondidas ficarão ativadas ao receberem um dado conjunto de exemplos de entrada;

2. Nas redes MLP as funções de ativação têm como entrada líquida uma média ponderada entre os exemplos de entrada e o conjunto de pesos. Por outro lado, nas redes RBF a utilização destas funções de ativação de base radial fazem com que sua ativação seja obtida a partir de uma norma ponderada da diferença entre o valor de entrada e o centro da função de base radial;

3. A camada de saída é composta por unidades de processamento lineares.

As funções de ativação mais utilizadas são [41] [48]:

1. Função Linear

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

2. Função Cúbica

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

3. Função de base Gaussiana

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

4. Função Logística

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

5. Função de base Multi-quadrática

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

6. Função de base Multi-quadrática inversa

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

7. Função de base lâmina Spline fina

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

onde, x_j são exemplos de entrada, μ_i e σ_i representam respectivamente o centro e a largura (dispersão) da i-ésima função de base radial e θ é um bias ajustado.

O cálculo da resposta da rede RBF correspondente aos neurônios da camada de saída é realizado pela Equação 2.29.

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

O treinamento das redes RBF pode ser realizado de diversas formas. Dentre as abordagens propostas, o treinamento mais empregado é chamado de treinamento híbrido, por utilizar uma aprendizagem não supervisionada para determinar os parâmetros das

funções de base radial da camada escondida e um aprendizado supervisionado para ajustar os pesos que ligam a camada escondida a de saída. Portanto, desde que as funções de base radial, após determinação de seus parâmetros, sejam consideradas fixas, o ajuste dos pesos que ligam a camada escondida para a camada de saída para a rede RBF fica equivalente a uma rede ADALINE. Neste caso, ao se utilizar neurônios lineares na camada de saída e uma função objetivo como o erro médio quadrático, o treinamento destas redes é bastante rápido, uma vez que dispomos de um sistema de equações lineares para ser resolvido, o que nos permite utilizar técnicas lineares de inversão de matriz.

Durante a primeira fase de treinamento, aprendizagem não supervisionada, os centros das funções de base radial podem ser determinados por algoritmos de clusterização, tais como algoritmos k-médias [56] e mapas auto-organizáveis de Kohonen [50].

O método k-média tem por objetivo encontrar um conjunto de K centros μ_i (i=1,2,...,K) representativos das funções de base radial em função do conjunto de exemplos de entrada x_j com $j = 1,2,...,N$. O algoritmo divide o conjunto de exemplos de entrada em K subconjuntos $S_i$cada um deles contendo N_i exemplos. O objetivo do método é minimizar a Equação 2.30.

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

onde, μ_i é a média dos pontos pertencentes ao conjunto S_i calculada por

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

Inicialmente, os exemplos de entrada são localizados aleatoriamente dentro de cada subconjunto K e então os centros μ_i são calculados. Posteriormente, os exemplos são redistribuídos de acordo com a sua maior proximidade com relação a estes centros. Este processo é então repetido até que não ocorram mais mudanças de exemplos entre os grupos. Este método permite também determinar o valor da largura (dispersão) σ_i da função de base radial por meio da distância euclidiana.

Logo, após a finalização do treinamento não-supervisionado onde foram determinados os parâmetros das funções de base radial, se realiza a segunda parte do treinamento que consiste em um treinamento supervisionado onde se determinam os pesos que ligam a camada escondida à camada de saída. Nessa fase, o treinamento é similar a uma rede tradicional com a utilização de uma função objetivo tal como o erro médio quadrático. Diversas técnicas de otimização podem ser utilizadas para o aprendizado dos pesos da camada de saída, responsáveis pela combinação linear das ativações da camada escondida, tais como: a regra delta, o método dos mínimos quadrados, a técnica de pseudo-inversão e o método OLS (*Orthogonal Least Squares*) [57].

As redes RBF são aproximadores universais de função tal como a rede MLP. Em geral as redes RBF têm um tempo de treinamento inferior ao de uma rede MLP, uma vez que o treinamento híbrido da RBF permite a utilização de técnicas lineares de rápida convergência para determinação dos pesos (entre as camadas escondida e a de saída) durante o aprendizado supervisionado.

As redes RBF por serem redes de aprendizado local, necessitam de uma quantidade maior de exemplos para o seu treinamento, para que se obter uma precisão similar a das redes MLP, que são redes de ajuste global. Isto implica que as redes MLP em geral são melhores aproximadores de funções, pois o ajuste global tende a fornecer uma maior capacidade de generalização. Entretanto, em problemas de classificação, as redes MLP tendem a cometer maiores erros de classificação ”falso positivo”do que as redes RBF [50].

2.5.4 Learning Rules

Uma propriedade de importância primordial para uma rede neural é a sua habilidade de aprender a partir de seu ambiente e de melhorar o seu desempenho. A melhoria do desempenho ocorre com o tempo de acordo com alguma medida preestabelecida. Uma rede neural aprende acerca do seu ambiente através de um processo iterativo de ajuste de seus pesos sinápticos e níveis de bias. Idealmente, a rede se torna mais instruída sobre o seu ambiente após cada iteração do processo de aprendizagem.

Gradient Descent Learning

O Gradiente Descendente (GD) requer a definição de uma função erro (ou objetivo) para medir o erro do neurônio na aproximação do alvo, a soma quadrática dos erros é normalmente usada 2.32. O objetivo do GD é encontrar os valores de peso que minimizem a função de erro obtido através do cálculo da inclinação da função de erro no espaço de pesos com o objetivo de mover o vetor de pesos ao longo do gradiente negativo, tal como ilustrado por um único erro na Figura 2.7 [48].

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

onde e_i e c_i são respectivamente a saída esperada e calculada para o i-ésimo padrão, e P_T é o número total de pares de vetores entrada-saída no conjunto de treinamento.

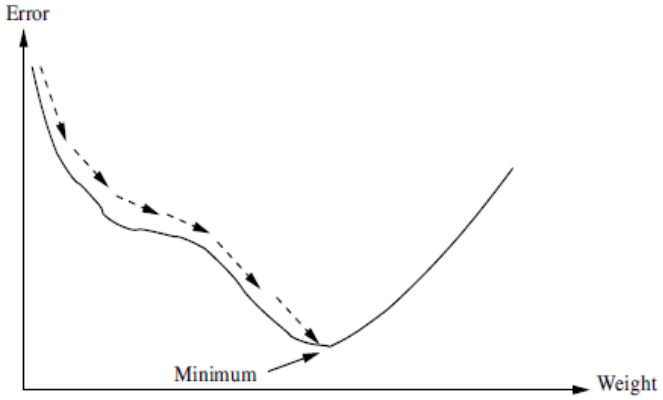

Figura 2.7: Vetores de Suporte e Hiperplano de Separação ótimo

Dado um padrão único de treinamento, os pesos são atualizados usando:

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

onde

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

e η é a taxa de aprendizagem (isto é, o tamanho dos passos dados na direção contrária ao gradiente). A regra de aprendizagem Widrow-Hoff apresenta uma solução para as funções de passo e rampa, enquanto a regra de aprendizagem delta generalizada assume funções contínuas que sejam pelo menos diferenciáveis uma vez [48].

Widrow-Hoff Learning

A regra de aprendizagem Widrow-Hoff assume que se a função objetivo $f = net_i$, então $\frac{\partial f}{\partial net_i}$. Essa regra de aprendizagem também é conhecida como o algoritmo *least-mean-square* (LMS), foi um dos primeiros algoritmos usados para treinar redes neurais em camadas com múltiplos neurônios lineares adaptativos (Madaline) [48].

Error Correction Learning

A regra delta é uma generalização da regra de Widrow-Hoff [44] que utiliza funções de ativação que tenham derivada real em todo seu domínio e ao menos um valor mínimo. A regra de aprendizagem é dada por:

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

ou seja, isso significa que o ajuste feito em um peso sináptico de um neurônio é proporcional ao produto do sinal de erro pelo sinal de entrada da sinapse em questão [51]. Assim, o ajuste a ser aplicado aos pesos é:

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

Esse algoritmo garante a minimização do erro ao longo do tempo, através do ajuste dos pesos, ou seja, sua convergência garante que a adaptação dos pesos seja realizada num número finito de iterações. A prova de convergência pode ser encontrada em Beale e Jackson [58].

Memory based Learning

Na aprendizagem baseada em memória, todas as (ou a maioria das) experiências passadas são armazenadas explicitamente em uma grande memória de exemplos de entrada-saída classificados corretamente: $\{(x_i, d_i)\}_{i=1}^{N}$, onde x_i representa um vetor de entrada e d_i representa a resposta desejada correspondente. Sem perda de generalidade, pode-se restringir a resposta desejada a um escalar. Em um problema de classificação de padrões binário, por exemplo, há duas classes/hipóteses a serem consideradas, representadas por ζ_1 e ζ_2. Neste exemplo, a resposta desejada d_i assume o valor 0(ou -1) para a classe ζ_1

e o valor 1 para a classe ζ_2. Quando se deseja classificar um vetor de teste $x_t este$ (não visto antes), o algoritmo responde buscando e analisando os dados de treinamento em uma "vizinhança local"de $x_t este$.

Todos os algoritmos de aprendizagem baseada em memória envolvem dois ingredientes essenciais:

1. O critério utilizado para definir a vizinhança local do vetor de teste $x_t este$;
2. a regra de aprendizagem aplicada aos exemplos de treinamento na vizinhança local de $x_t este$.

Em um tipo mais efetivo de aprendizagem baseada em memória conhecido como a regra do vizinho mais próximo, a vizinhança local é definida como o exemplo de treinamento que se encontra na vizinhança imediata do vetor de teste $x_t este$. Em particular, diz-se que o vetor

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

é o vizinho mais próximo de $x_t este$ se

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

onde $d(x_i, x_t este)$ é a distância euclidiana entre os vetores x_i e $x_t este$. A classe associada com a distância mínima, ou seja, o vetor x'_N é apresentada como a classificação de $x_t este$. Esta regra é independente da distribuição fundamental responsável pela geração dos exemplos de treinamento.

Gradiente Conjugado Escalonado

A otimização do gradiente conjugado oferece uma escolha entre a simplicidade do gradiente descendente e a rápida convergência quadrática do método de Newton. Vários algoritmos de aprendizagem do gradiente conjugado foram desenvolvidos, a maioria deles são baseados no pressuposto de que a função de erro de todos os pesos da região de solução pode ser aproximada com precisão pela Equação 2.39 [48]:

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

onde H é a matriz Hessiana. Desde que a dimensão da matriz Hessiana seja o número total de pesos na rede, o cálculo das direções conjugadas na superfície de erro torna-se computacionalmente inviável. Algoritmos de gradiente conjugado computacionalmente viáveis calculam as direções do gradiente conjugado sem explicitamente computar a matriz Hessiana, e realizam atualizações nos pesos ao longo dessas direções.

Um aspecto importante nos métodos de gradiente conjugado são os vetores de direção $\{p(0), p(1), ..., p(t-1)\}$. Esses vetores são criados para ser conjugados com o vetor de pesos w. Isto é, $p^T(t_1) w p(t_2) = 0$ para $t_1 \neq t_2$. Um novo vetor de direção é gerado a cada iteração pela adição do vetor de gradiente negativo calculado, da função de erro, a uma combinação linear dos vetores de direção anteriores. O algoritmo de gradiente

conjugado padrão assume uma função de erro quadrática, nos casos em que os algoritmos convergem em não mais que n_w passos, onde n_w é o número total de pesos e vieses. O algoritmo do gradiente conjugado Fletcher-Reeves não assume uma função de erro quadrática. O algoritmo é reiniciado após n_w iterações se ainda não foi encontrada uma solução [48].

Os fatores de escala também pode ser calculado através de métodos Polak-Ribiere e Hestenes-Stiefer. Moller [59] propôs o algoritmo de gradiente conjugado escalonado (GCS) como um algoritmo de aprendizado *batch*. Os tamanhos do passo são determinados automaticamente e o algoritmo é reiniciado após n_w iterações se uma boa solução não foi encontrada.

Quasi-Newton Learning

As variações em algoritmos de minimização são o gradiente e a matriz Hessiana. O gradiente deve ser conhecido com precisão assim como as direções de descidas devem ser calculadas a partir delas mesmas e aproximações ao gradiente não oferecem a precisão necessária. Por outro lado, a matriz Hessiana pode ser aproximada pela técnica das secantes. Uma vez que a matriz Hessiana é a matriz Jacobiana de um sistema de equações não-lineares $\nabla F(x) = 0$, ela pode ser aproximada. porém, a matriz Hessiana tem duas propriedades importantes: ela é sempre simétrica e, frequentemente, positiva. A incorporação dessas duas propriedades na aproximação secante é um aspecto importante dos métodos *Broyden-Fletcher-Goldfarb-Shanno back-propagation* (BFGS-BP) e *One Step Secant back-propagation* (OSS-BP) discutidos adiante. Eles são mais frequentemente chamados de atualizações secantes positivas definidas [60].

Broyden-Fletcher-Goldfarb-Shanno back-propagation method

No método de Newton uma aproximação quadrática é utilizada ao invés de uma aproximação linear da função $F(x)$. A próxima solução aproximada é obtida num ponto que minimize a função quadrática na Equação 2.40.

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

Assim, a sequência obtida é a Equação 2.41.

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

A principal vantagem do método de Newton é que ele tem uma taxa de convergência quadrática, enquanto que o descendente mais acentuada tem uma taxa de convergência linear, que é muito mais lento. No entanto, cada passo do método de Newton requer uma grande quantidade de computação. Supondo-se que a dimensionalidade do problema é N, uma operação de ponto flutuante O(N^3) é necessária para calcular a direção da busca d^k. Um método que utiliza uma matriz Hessiana aproximada para o cálculo da direção de busca é o método quasi-Newton. Seja H_k uma matriz simétrica $N \times N$ que aproxima a matriz Hessiana A_k; então a direção da pesquisa para o método quasi-Newton é obtida pela minimização da função quadrática, descrita pela Equação 2.42 [60].

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

À medida que a matriz H_k é aproximada à matriz Hessiana da função $F(x)$ em $x = x_k$, ela precisa ser atualizada a cada iteração, incorporando as informações de gradiente mais recentes [60].

One-step Secant backpropagation method

Uma desvantagem da atualização do BFGS-BP na Equação 2.42 é que ela requer o armazenamento para uma matriz de tamanho $N \times N$ e cálculos de ordem O(N^2). Apesar do armazenamento disponível ser um problema menor agora do que foi a uma década atrás, o problema computacional ainda existe para um grande valor de N. É possível usar uma aproximação secante com computação de O(N). Nesse método, a nova direção de busca é obtida a partir de vetores calculados dos gradientes. Se g_{k+1} é o gradiente atual, a nova direção de busca p_{k+1} é obtida a partir da Equação 2.43 [60],

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

onde $g_{k+1} = \nabla F(x_{k+1})$ e os dois escalares B_k e C_k são a combinação dos produtos escalares dos vetores definidos anteriormente s_k, g_{k+1} e y_k (último passo, gradiente atual e diferenças dos gradientes)

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

e

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

A linha de busca de *backtracking* é utilizada como o algoritmo de otimização do quasi-Newton do OSS-BP. No início do aprendizado, a direção de busca é $-g_0$ e é reiniciada para $-g_{k+1}$ a cada N passos. O multiplicador $\left(1 + \frac{y_k^T y_k}{s_k^T y_k}\right)$ aumenta a última taxa de aprendizado que obteve sucesso e o primeiro passo da tentativa é executado. Se a energia da rede é maior do que o valor limite superior, então uma nova tentativa é experimentada por meio de interpolação quadrática sucessiva até que a exigência seja cumprida. A taxa de aprendizagem é reduzida à metade após cada tentativa mal sucedida [60].

Levenberg-Marquardt Algorithm

O Algoritmo de Levenberg-Marquardt adaptativamente varia as atualizações de parâmetros entre a atualização do gradiente descendente e a atualização de Gauss-Newton [61],

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

onde pequenos valores de parâmetros do algoritmo λ resulta em uma atualização Gauss-Newton e grandes valores de λ resultam numa atualização do gradiente descendente. O parâmetro λ é inicializado para ser grande. Se uma iteração acontece de resultar numa aproximação pior, λ é aumentado. À medida que a solução se aproxima do mínimo, λ é diminuído, assim o método de Levenberg-Marquardt se aproxima do método de Gauss-Newton, e a solução tipicamente converge rapidamente para o mínimo local [61].

O algoritmo sugeriu uma relação de atualização como na Equação 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

Nesta seção, apresentamos uma classe de redes adaptativas que são funcionalmente equivalentes aos sistemas de inferência Fuzzy, analisados durante esse estudo. A arquitetura é denominada como ANFIS, que significa *Adaptive Network-based Fuzzy Inference System* ou semanticamente equivalente, Sistema de Inferência Neuro-fuzzy Adaptativo. Nós descrevemos como decompor o conjunto de parâmetros para facilitar a regra de aprendizagem híbrida para arquiteturas ANFIS representando os modelos fuzzy de Sugeno e Tsukamoto. Também demonstramos que, sob certas pequenas restrições, a Rede com Função de Base Radial é funcionalmente equivalente a arquitetura ANFIS para o modelo fuzzy de Sugeno [62].

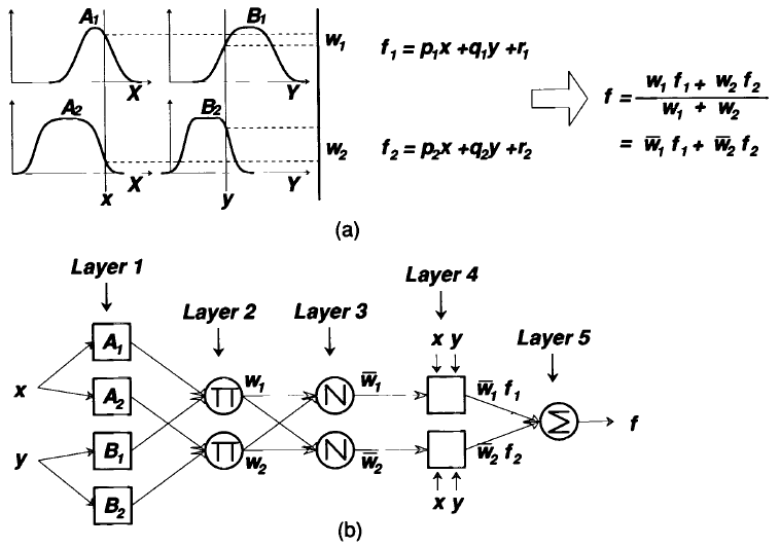

Figura 2.8: (a) Um modelo fuzzy de Sugeno de primeira ordem com duas entradas e duas regras; (b) arquitetura equivalente ANFIS

ANFIS Architecture

Por simplicidade, nós assumimos que o sistema de inferência fuzzy em consideração tem duas entradas x e y e uma saída z. Para um modelo fuzzy de Sugeno de primeira ordem, um conjunto de regras comum com duas regras fuzzy if-then são as sequintes:

1. Regra 1: Se x é A_1 e y é B_1, então $f_1 = p_1x + q_1y + r_1$,
2. Regra 2: Se x é A_2 e y é B_2, então $f_2 = p_2x + q_2y + r_2$.

A Figura 2.8(a) ilustra o mecanismo de raciocínio para esse modelo de Sugeno; a arquitetura ANFIS equivalente correspondente é como mostrada na Figura 2.8(b), em que nós da mesma camada têm funções similares, como descrito a seguir (Aqui definimos a saída do i-ésimo nó na camada l como $O_{1,i}$).

Camada 1: Cada nó i nessa camada é um nó adaptativo com um função:

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

em que x (ou y) é a entrada para o nó i e A_i (ou B_{i-2}) é um rótulo linguístico (tal como "pequeno"ou "grande") associado a esse nó. Em outras palavras, $O_{1,i}$ é o grau de pertinência a um conjunto fuzzy $A(A_1, A_2, B_1 or B_2)$ e ele especifica o grau em que a entrada de dados x (ou y) satisfaz o quantificador A. Aqui, a função de pertinência para A pode ser qualquer função de pertinência parametrizada apropriada, como a função sino generalizada [62]:

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

Camada 2: Cada nó nessa camada é um nó fixo rotulado Π, cuja saída é o produto de todos os sinais de entrada:

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

Cada nó saída representa a força de disparo de uma regra. Em geral, quaisquer outros operadores de norma T que executam a operação fuzzy "E"podem ser usados como função de nó nessa camada.

Camada 3: Cada nó nessa camada é um nó fixo rotulado N. O i-ésimo nó calcula a taxa i-ésima força de disparo da regra para a soma de todas as forças de disparo das regras:

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

Por conveniência, as saídas dessa camada são chamadas **forças de disparo normalizadas**.

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

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

em que $\overline{w_i}$ é uma força de disparo normalizada da camada 3 e p_i, q_i, r_i é o conjunto de parâmetros para esse nó. Parãmetros nessa camada são referenciados como "parâmetros consequentes".

Camada 5: O nó único nessa camada é nó fixo rotulado $\sum$, que computa a saída geral como o somatório de todos os sinais de entrada:

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

Assim, nós construímos uma rede adaptativa que é funcionalmente equivalente a um modelo difuso Sugeno. Note que a estrutura da rede adaptativa não é única; podemos combinar as camadas 3 e 4 para obter uma rede equivalente com apenas quatro camadas. Do mesmo modo, pode-se realizar a normalização do peso na última camada. No caso extremo, podemos até diminuir toda a rede em um único nó adaptativo com o mesmo conjunto de parâmetros. Obviamente, a atribuição de funções de nó e a configuração de rede são arbitrárias, desde que cada nó e cada camada execute funcionalidades significativas e modulares [62].

A extensão do ANFIS Sugeno para Tsukamoto é direta, em que a saída de cada regra $(f_i, i = 1,2)$ é induzida conjuntamente por um conseqüente função de pertinência e uma força de disparo. Para o sistema de inferência fuzzy Mamdani com a composição max-min, um ANFIS correspondente pode ser construído se aproximações discretas são utilizadas para substituir as integrais no esquema de defuzificação centróide. No entanto, o ANFIS resultante é muito mais complicado do que qualquer ANFIS Sugeno ou Tsukamoto. A complexidade extra na estrutura e cálculo de Mamdani ANFIS com composição max-min não implica, necessariamente, uma melhor capacidade de aprendizagem ou a aproximação de energia [62].

Capítulo 3

Metodologia

Nessa dissertação, propõe-se uma metodologia para a análise de risco em projetos de software, a partir de dados históricos de registros de riscos, por meio da utilização de redes neurais artificiais. Para desenvolver essa metodologia, primeiramente, necessitamos de uma base de dados extensa e real de riscos em projetos de software. No entanto, há uma dificuldade em encontrar publicamente bases de dados representativas e confiáveis nessa área. Uma base de dados de risco chamada PERIL [2] mostrou atender as necessidades básicas para essa pesquisa, além de já estar totalmente classificada (mais detalhes na Seção 3.1). Nesse estudo, precisa-se de uma base de dados extensa de registros de riscos oriundos de diversos projetos espalhados pelo Mundo e em diversos períodos de tempo, com o objetivo de mostrar evidências do que ocorre no gerenciamento desses projetos.

Segundo, é necessário realizar o pré-processamento dos dados para que as variáveis de entradas sejam transformadas, normalizadas e selecionadas para o estudo. Terceiro, avaliar o desempenho das ferramentas de estado da arte quanto ao erro de previsão: Simulação de Monte Carlo e Análise PERT.

Quarto, implementar cada modelo de previsão para que pudéssemos selecionar o melhor modelo em redes neurais artificiais: Perceptron de Múltiplas Camadas, Máquinas de Vetor de Suporte, Redes de Função de Base Radial e o Sistema de Inferência Adaptativo Neuro-Fuzzy. Alguns desses modelos selecionados apresentam alguns parâmetros e devido a diversidade de possíveis valores para cada parâmetro, necessita-se otimizar os parâmetros dos modelos. Nesse estudo, implementa-se uma variação da meta-heurística Otimização por Enxame de Partículas (PSO - *Particle Swarm Optimization*) com coeficiente de constrição de Clerk [48] para realizar a tarefa de otimização dos parâmetros das RNAs. O espaço de busca do problema de otimização desses parâmetros é multidimensional, complexo e apresenta diversos mínimos locais. Na Figura 3.1, observa-se um esquema ilustrando que um algoritmo de otimização executa os quatro modelos em suas diversas configurações paramétricas para que seja possível eleger o modelo mais coerente com a base de dados adotada.

Quinto, um teste de validação dos resultados é realizado para verificar se os modelos baseados em redes neurais são mais eficazes que os modelos de estado da arte em análise de riscos: Simulação de Monte Carlo, Análise PERT, Modelo de Regressão Linear Múltipla, Modelo de Regressão em Árvore. Os dois últimos modelos de regressão

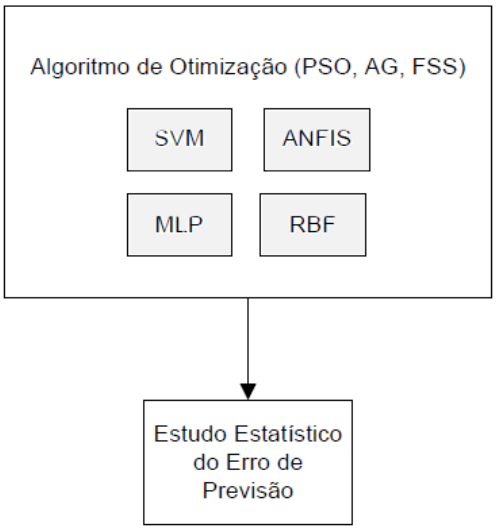

Figura 3.1: Esquema de Seleção da Melhor Rede Neural Artificial

linear foram considerados como linha de base, por serem modelos mais simples, já que realizam a regressão linear. Os dois últimos modelos foram implementados e também são avaliados quanto ao erro de previsão.

A seleção do melhor modelo é feita em termos da precisão na previsão. É difícil obter uma métrica representativa da precisão de um modelo. No entanto, Engelbrecht [48] e Saxena [25] sugerem que a Raiz do Erro Médio Quadrático(REMQ), do inglês *Root Mean Square Error* é uma medida conveniente e aplicável para a maioria dos problemas. A precisão, nesse caso, significa o grau de proximidade de uma saída calculada para a esperada. A REMQ é representada pela Equação 3.1:

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

onde $e_i = f_i - y_i$, f_i é o resultado calculado, y_i é o resultado esperado e n é o número de pares de dados.

Todas as técnicas estudadas nesse trabalho estimam a saída para o impacto de riscos. A REMQ é calculada trinta vezes para cada abordagem. Um Teste Estatístico de Wilcoxon Não-pareado [63] pode ser necessário para determinar qual é uma abordagem mais precisa para a base de dados (nesse estudo, o PERIL). O Teste de Wilcoxon Não-pareado é utilizado porque não há evidências que as amostras sejam oriundas de uma população normalmente distribuída, como também não há relação de ordem entre os valores de diferentes amostras.

A validação cruzada [64] é utilizada para evitar a ocorrência de *overfitting* ou *underfitting* durante o treinamento das redes neurais. Nesse caso, o treinamento com parada prematura é utilizado para identificar o início do *overfitting* porque esse método tem provado ser capaz de melhorar a capacidade de generalização da Rede Neural Artificial em comparação com o treinamento exaustivo [51] [48] [65]. Portanto, o método de validação cruzada é utilizado para cada abordagem, excluindo a Simulação de Monte

Carlo e Análise PERT, para promover maior capacidade de generalização. Quando adotamos o uso da validação cruzada, significa que necessitamos particionar a base de dados em três partes: conjunto de treinamento, conjunto de validação cruzada e conjunto de testes. O conjunto de treinamento é utilizado na fase de treinamento das redes neurais, momento em que o aprendizado ocorre. O conjunto de validação cruzada é processado ao mesmo tempo, com o conjunto de treinamento e determina a parada no treinamento. Já o conjunto de testes, é utilizado para produzir a métrica de precisão do modelo (REMQ).

Por fim, a previsão do impacto do risco e a definição de um intervalo de confiança para uma amostra do conjunto de treinamento são obtidas utilizando o modelo de previsão mais preciso após os testes de validação. É necessário produzir um intervalo de confiança da previsão para que os gerentes de projetos e analistas de risco possam estabelecer o nível de confiança de acordo com a sua necessidade. Afinal de contas, esse é o resultado que eles esperam.

A Figura 3.2 apresenta um fluxograma com a metodologia estabelecida para esse estudo. O procedimento inicia com a seleção da base de dados que serve como conjunto de dados de entrada, nesse caso o PERIL, e finaliza com a atividade ”Definição do Intervalo de Confiança”. Todas as atividades são executadas sequencialmente, exceto as atividades ”Avaliação dos Modelos de Estado da Arte”e ”Seleção da Melhor Rede Neural Artificial”que são executados paralelamente.

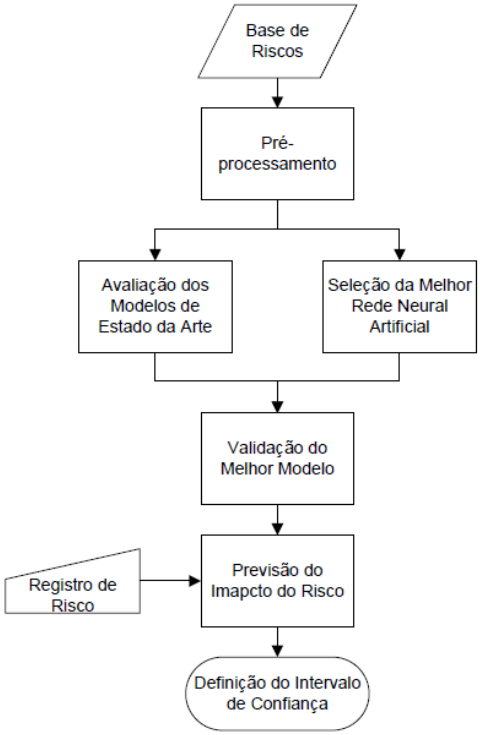

Figura 3.2: Fluxograma do estudo realizado

3.1 PERIL Database

Um melhor gerenciamento de riscos inicia com a identificação de problemas potenciais, atribuído como fatores de risco. A adoção dos métodos disponíveis como: revisar lições aprendidas, *brainstorming*, entrevistas e opinião especializada são alternativas relativamente eficientes, no entanto, na maioria das situações elas envolvem alto custo. Uma proposta de baixo custo, extensiva e acessível é utilizar a base de dados *Project Experience Risk Information Library*(PERIL) [2]. A base de dados PERIL provê informações e experiências de outras gestão de risco de projetos.

Por mais de uma década, em *Workshops* de Gerenciamento de Riscos, Kendrick coletou dados anônimos de centenas de líderes de projetos lidando com seus problemas em projetos passados. Ele compilou essas informações na base de dados PERIL, que sumariza tanto uma descrição do que houve de errado quanto o impacto que ele teve em cada projeto. A base provê uma perspectiva preocupante do que projetos futuros podem encarar e é valiosa na ajuda para a identificação do que pelo menos poderiam ter sido riscos invisíveis ou *black swans* [2].

Em projetos, os riscos identificados podem ser classificados como ”conhecidos”, aqueles antecipados durante o planejamento, ou ”desconhecidos”, identificados futuramente durante a execução do projeto. O propósito dessa base de dados é prover um *framework* para identificar riscos, de modo a aumentar o número de ”conhecidos”e diminuir o número de ”desconhecidos”.

Algumas características da PERIL são:

- Os dados não estão relacionados, eles representam somente uma pequena fração de dezenas de milhares de projetos realizados pelos líderes de projetos, cujos riscos foram coletados;
- Apresentam viés, a informação não foi coletada aleatoriamente;
- Representam somente os riscos mais significativos;
- São de todo o mundo, com a maioria nas Américas;
- Não identificam oportunidades;
- Contém seis centos e quarenta e nove registros, cujo impacto relativo é baseado no número de semanas de atraso no cronograma;
- Projetos comuns tiveram um cronograma inicialmente planejado entre seis meses e um ano;
- Tamanho da equipe raramente foi maior que vinte pessoas.

Os registros de risco são categorizados em escopo, cronograma e recurso. O escopo é decomposto nas subcategorias mudança e defeito. O cronograma é decomposto em dependência, estimativa e atraso. Os recursos são decompostos em dinheiro, *outsourcing* e pessoas. Um benefício da PERIL é que o autor contempla ”*black swans*”: riscos com grande impacto, difíceis de prever e com ocorrência rara [66].

Kendrick decidiu padronizar o impacto usando como métrica o tempo, medido em semanas de atraso no projeto. Essa tática faz sentido à luz da obsessão de hoje por cumprimento de prazos e foi uma escolha fácil, pois, de longe, o impacto mais sério relatado nesses dados foi o atraso no prazo. Focar-se em tempo também é apropriado porque entre as restrições triplas de projeto - escopo, tempo e custo -, tempo é o único é completamente fora de nosso controle. Afinal, quando se vai, se foi [16].

Tabela 3.1: Raw project numbers in the PERIL database

	Americas	Asia	Europe/Middle East	Total
IT/Solution Project	256	57	18	331
Product Development Project	224	66	28	318
Total	**480**	**123**	**46**	**649**

Na Tabela 3.1, apresenta-se o número de projetos em TI e de desenvolvimento de produtos nas Américas, Ásia e Europa.

Tabela 3.2: 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

A Tabela 3.2 apresenta o número de casos, o impacto cumulativo e médio em semanas para cada categoria e sub-categoria de causa-raiz; além do significado de cada subcategoria.

Uma desvantagem dessa base de dados, é que ela somente contabiliza riscos que tiveram impacto negativo no projeto. As oportunidades não foram identificadas e maximizadas com esse estudo. No entanto, um dos grandes benefícios é que o autor apresenta alguns riscos como *black swans* [16]: representando a idéia de riscos com amplo

impacto, difíceis de prever e raros de ocorrer. Se o risco tiver impacto negativo, é conhecido como catástrofe, ao passo que, se tiver impacto positivo, é conhecido como recompensa.

3.1.1 Black Swans

Denominar alguns riscos como "*black swans*"têm sido popularizado desde os textos de Nassim Nicholas Taleb [66]. A noção de um "*black swan*"originou-se na Europa antes de ser popularizada pelo Mundo. Já que todos os cisnes observados na Europa eram brancos, um cisne negro era considerado impossível de existir. Porém, foi como um choque quando uma espécie de cisne negro foi descoberta na Austrália. Esse fato deu origem ao uso metafórico do termo "*black swan*"para descrever algo erroneamente acreditado ser impossível.

O conceito de Taleb acerca de "*black swan*"é um evento raro, difícil de prever e de grande impacto. No entanto, é aplicável à gestão de risco do projeto. Nos projetos, é comum que os líderes de projeto para descontar principais riscos do projeto, porque eles são estimados para ter probabilidades extremamente baixas. No entanto, esses riscos ocorrem - a PERIL é cheia deles - e a severidade dos problemas que eles causam significa que ignorá-los pode ser imprudente. Quando esses riscos ocorrem, o mesmo gerente de projetos que inicialmente os negaram começam a percebê-los como muito mais previsíveis - às vezes até mesmo inevitável [16].

Na PERIL, há 127 casos representando os maiores atrasos no cronograma. Como a base de dados mostra, estes riscos mais danosos não são tão raros quanto devem ser pensados, e não devem ser tão difíceis de ser previstos por gerentes se eles dedicarem a atenção apropriada para o processo de gerenciamento de riscos [16].

Em muitas situações, a tarefa mais difícil é identificar e estimar "*black swans*"devido a sua característica: emergente, inesperado, imprevisível e com alto impacto. Portanto, *"black swans"* também são considerados nesse estudo.

3.2 Data Preprocessing

PERIL contém valores nominais e numéricos. Logo, variáveis nominais foram expressadas através de variáveis binárias. Nesse estudo, utilizam-se doze variáveis binárias para representar as variáveis nominais. Segundo, o impacto que representa a saída real, são números inteiros. Foi observado que a função de distribuição de probabilidade do impacto ajusta-se às funções log-normal e gamma. Portanto, foi realizada uma normalização gamma [38]. Terceiro, a seleção das variáveis mais significativas para o estudo foi realizado após o resultado da análise promovida pelo algoritmo *Random Forest* proposto no livro de Luís Torgo [37].

A Figura 3.3 e 3.4 apresentam os histogramas das variáveis de entrada. Todos os dados encontram-se binarizados como pode ser observado nos gráfico em barras, que contém o número de ocorrências para cada intervalo de valores.

A Figura 3.5 apresenta a saída normalizada pela função gamma para a PERIL num histograma. A forma da função de distribuição de probabilidade é exibida como uma

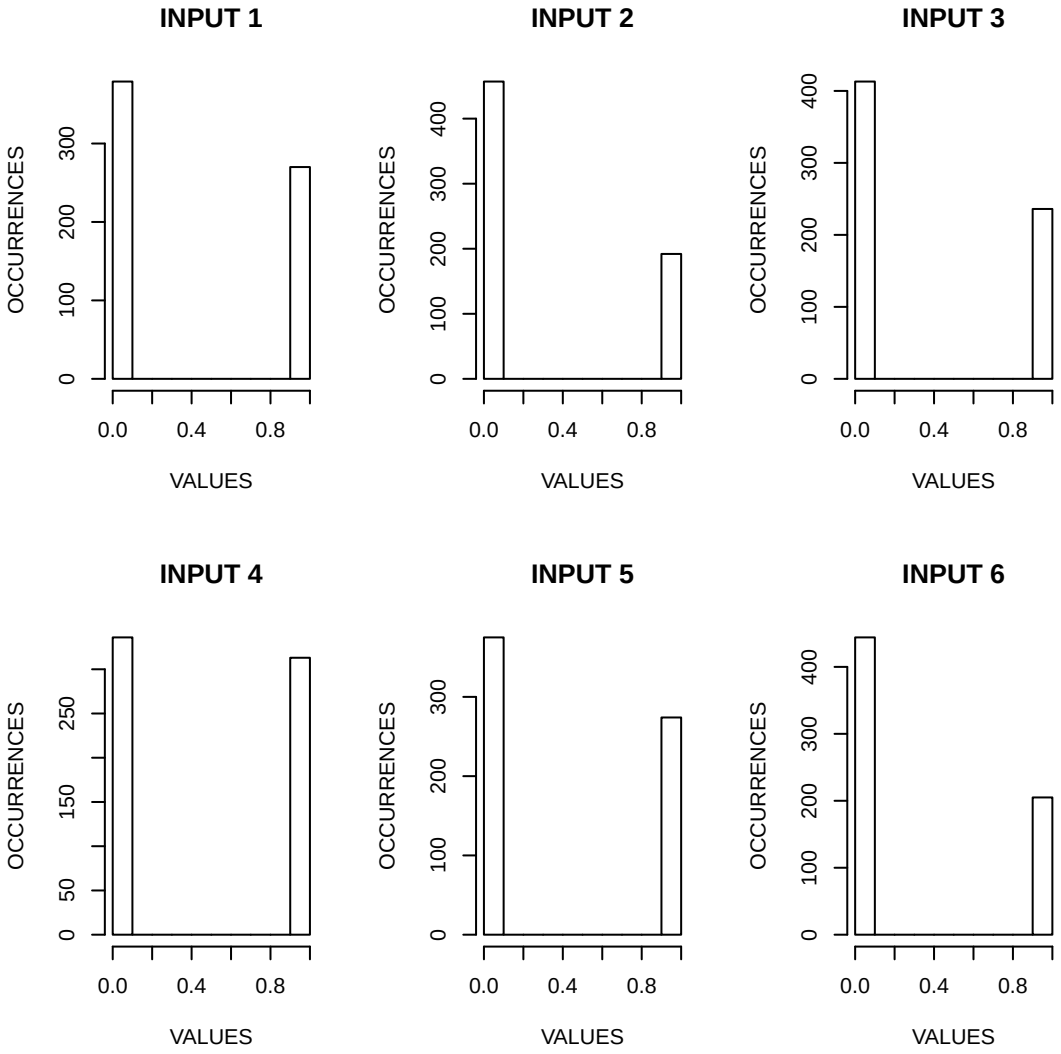

Figura 3.3: First six input variables

curva sobre o histograma.

Para o nosso objetivo, PERIL foi dividida em três subconjuntos disjuntos - treinamento, validação cruzada e teste, correspondendo a cinquenta, vinte e cinco porcento e o restante da base de dados, respectivamente. O método de validação-cruzada com divisão de amostras foi utilizado tanto para MRLM e MRA quanto para MLP, SVM, RBF e ANFIS; sendo que para esses a parada prematura também foi utilizada no treinamento [67].

3.3 Modelos de Estado da Arte

Nesta seção, são descritos as configurações adotadas para a experimentação dos modelos de estado da arte.

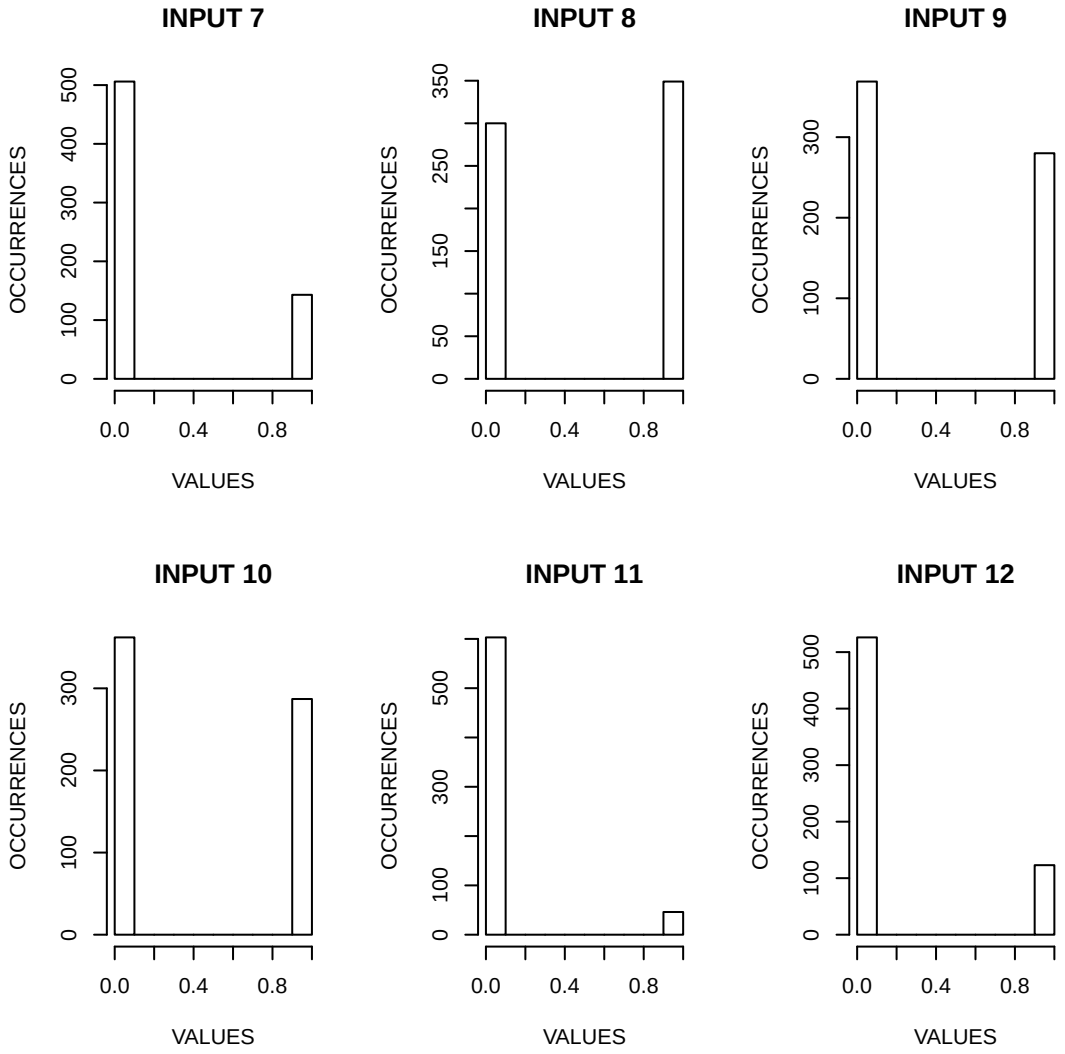

Figura 3.4: Last six input variables

3.3.1 Monte Carlo Simulation

A Simulação de Monte Carlo utilizou a base de dados inteira com o objetivo de aumentar o desempenho do modelo durante a previsão. Além disso, foram filtradas somente as saídas possíveis para uma dada entrada para que uma nova saída pudesse ser calculada, através disso, o desempenho da previsão também aumentou.

3.3.2 Análise PERT

A Análise PERT também utilizou a base de dados inteira e somente as saídas possíveis para uma determinada entrada foi utilizada no cálculo da nova saída, tal qual a Simulação de Monte Carlo.

A partir dessa configuração pudemos otimizar esses modelos e observar que há uma falta de habilidade na capacidade de previsão do impacto de riscos a partir de uma base

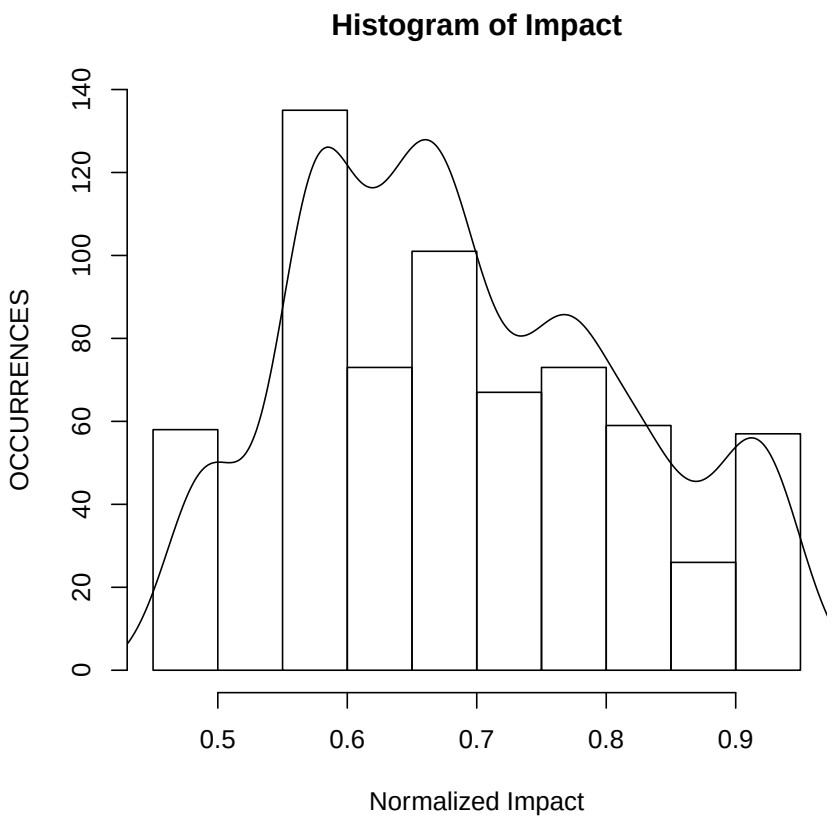

Figura 3.5: Histogram of impact and shape of the distribution fitting function

de dados através de modelos estatísticos.

3.4 Modelos de Regressão Linear

Nesta seção, são descritos as configurações adotadas para a experimentação com os modelos de regressão linear.

O código-fonte dos modelos de regressão linear foram adaptados de Torgo [37] para realizar o treinamento, validação cruzada, teste e avaliação da Raiz do Erro Médio Quadrático. Os modelos de RLM e MAR foram comparados estatisticamente para que se pudesse definir um modelo de regressão linear padrão para estudos futuros com a base de dados correspondente.

3.4.1 Modelo de Regressão Linear Múltipla

Após a otimização do Modelo de Regressão Linear através da seleção das melhores variáveis de entrada utilizando o critério *Akaike Information Criterion*(AIC) sete das onze variáveis foram selecionadas, além do termo independente.

3.4.2 Modelo de Árvore de Regressão

O modelo de árvore de regressão constrói uma árvore de classificação das variáveis de entrada, nesse caso pode não ser necessário utilizar todas as variáveis de entrada para a construção do modelo.

Na Seção 4, três modelos de árvore de regressão foram analisados com o modelo de regressão linear múltipla.

3.5 Otimização por Enxame de Partículas

Para o PSO, os parâmetros descritos na Tabela 3.3 foram utilizados. A variação do PSO utilizada é aquela que implementa o coeficiente de constrição de Clerk, como explicado anteriormente.

Tabela 3.3: Parâmetros do PSO.

Parameter	Value
Coeficiente Cognitivo	2.05
Coeficiente Social	2.05
Fator de Inércia	0.8
Número de partículas	30
Número de ciclos	600

Cada partícula no PSO representa uma configuração candidata. A função de avaliação das partículas é a REMQ de trinta avaliações da rede neural artificial analisada cujos parâmetros são cada partícula. Os parâmetros das redes MLP, SVM e RBF são determinados após a execução desse algoritmo.

3.6 Redes Neurais Artificiais

Nesta seção, descrevemos as configurações das redes neurais artificiais utilizadas nesse estudo.

3.6.1 MLPs

Algumas variações do modelo da MLP foram utilizadas nesse estudo. Diversos parâmetros podem ser alterados como: a quantidade de camadas escondidas, quantidade de neurônios escondidos em cada camada escondida, taxa de aprendizado, momento, número máximo de ciclos de treinamento e regra de aprendizado. Um modelo MLP utilizado nesse estudo é apresentado na Figura 3.6. Nesse modelo tem-se dez neurônios escondidos em uma única camada escondida.

A quantidade de camadas escondidas estudadas foram uma e duas camadas. O número de neurônios na(s) camada(s) escondida(s), a taxa de aprendizado e o momento

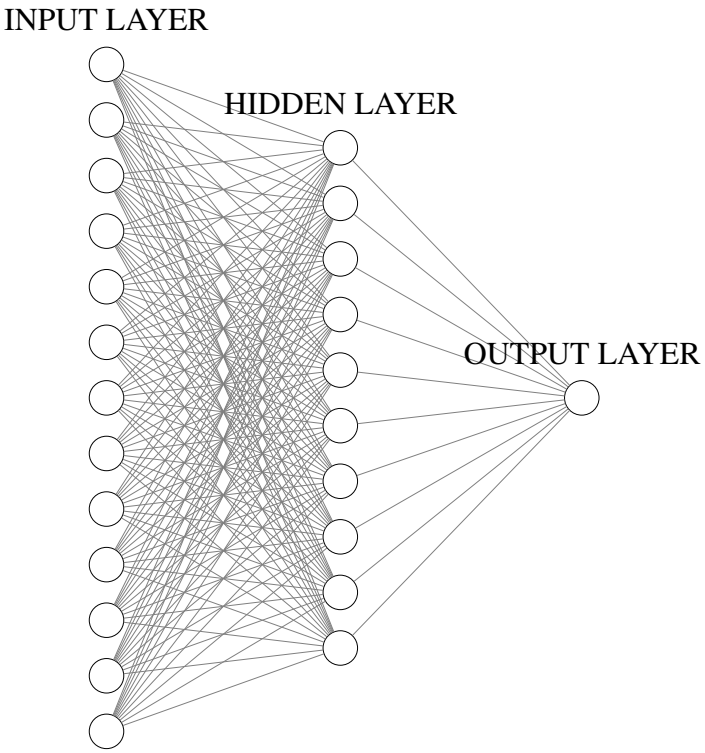

Figura 3.6: Um modelo MLP utilizado no estudo.

foram determinados por um algoritmo de otimização como o PSO para aumentar a precisão na estimativa de erros. Para as análises, o número máximo de ciclos de treinamento foi configurado para seiscentos.

A taxa de aprendizado, momento e a quantidade de neurônios escondidos variam de acordo com os valores apresentados na Tabela 3.4.

Tabela 3.4: 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

Por fim, as regras de aprendizado utilizadas nesse estudo são *Backpropagation*, Levenberg-Marquardt, BFGS Quasi-Newton, *Resilient Backpropagation*, *Polak-Ribiére Conjugate Gradient*, Gradiente Conjugado Escalonado e *One Step Secant*.

Em particular, uma MLP, chamada ”MLPRegressor”, que tem uma camada escondida e cuja regra de aprendizado tem o objetivo de minimizar o erro médio quadrático mais uma penalidade quadrática através do método BFGS Quasi-Newton teve um melhor desempenho que as demais variações.

3.6.2 SVM

O algoritmo SVM para regressão utilizado é o SMOReg. Nesse algoritmo RegSMOImproved é o algoritmo de otimização e PolyKernel é a função de kernel como descrito em [68]. O pseudo-código para esse algoritmo é apresentado no Algoritmo 3.6.2.

```
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
```

No Algoritmo 3.6.2, os dados são lidos a partir de um arquivo, dividido nos subconjuntos treinamento, validação cruzada e teste. Instanciar o modelo de regressão SMOReg, executar o treinamento do modelo, gerar a saída calculada e calcula o REMQ a partir da saída real e calculada. Esse algoritmo é utilizado como a função de otimização para o algoritmo de otimização por enxames.

3.6.3 RBF

A rede neural RBF utilizada é a RBFRegressor, ela minimiza o erro quadrático através do método BFGS. Os centros iniciais das gaussianas são encontrados utilizando SimpleKMeans, um algoritmo que implementa K-Médias. O sigma inicial é configurado para a maior distância entre qualquer centro e o vizinho mais próximo no conjunto de centros. O parâmetro de cume é usado para penalizar o tamanho dos pesos na camada de saída, o qual implementa uma combinação linear simples. O número de funções de base pode também ser especificado. Para esse estudo somente um sigma global é utilizado para todas as funções de base. O pseudo-código para esse algoritmo é apresentado no Algoritmo 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
```

No Algoritmo 3.6.3, os dados são lidos a partir de um arquivo, dividido nos subconjuntos treinamento, validação cruzada e teste. O modelo de regressão SMOReg é instanciado, o treinamento do modelo é executado, a saída calculada é gerada e a REMQ é obtida a partir da saída real e calculada. Esse algoritmo é utilizado como a função de otimização para o algoritmo de otimização por enxames.

3.6.4 ANFIS

O ANFIS é um sistema neuro-fuzzy implementado no Matlab por Sugeno [62]. ANFIS usa um algoritmo de aprendizado híbrido para identificar parâmetros do sistema de inferência fuzzy Sugeno. Ele aplica uma combinação do método dos mínimos quadrados e o método do gradiente descendente *backpropagation* para o treinamento dos parâmetros da função de pertinência do sistema de inferência fuzzy. O sistema de inferência fuzzy utilizado foi o genfis2, já que há um número grande de variáveis de entrada. O pseudocódigo para esse algoritmo é apresentado no Algoritmo 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
```

No Algoritmo 3.6.4, os dados de entrada e saída são lidos a partir de um arquivo. O sistema de inferência fuzzy é instanciado, o treinamento do modelo é executado, o erro é gerado e a REMQ é obtida a partir da saída real e calculada. Esse algoritmo é utilizado como a função de otimização para o algoritmo de otimização por enxames.

Capítulo 4

Experiments

Os experimentos realizados utilizaram os modelos descritos no Capítulo 3. A base de dados utilizada é o PERIL. Os experimentos são estabelecidos analisando os modelos de regressão que são linha de base para o estudo, seguindo para as técnicas comumente utilizadas na academia e na indústria. Após isso, o modelo estado da arte é analisado e por último, mas não menos importante, nos concentramos nas técnicas propostas pela metodologia desse estudo apresentadas na Seção 3.6. Os resultados para cada experimento são apresentados no Capítulo 5.

O objetivo global é utilizar a metodologia proposta no Capítulo 3 para determinar uma abordagem mais precisa para a estimativa do impacto de riscos. A métrica utilizada para verificar o atingimento desse objetivo é o erro de previsão, REMQ. Uma questão a ser respondida é: As RNAs são técnicas mais precisas para a estimativa do impacto do risco no gerenciamento de projetos de software?

As hipóteses para os experimentos são:

- H_0: Não há diferença entre usar as Redes Neurais Artificiais e os modelos de Estado da Arte para a estimativa do impacto de riscos;
- H_1: As Redes Neurais Artificiais são mais precisas que os modelos de Estado da Arte para a estimativa do impacto de riscos;
- H_2: As Redes Neurais Artificiais são menos precisas que os modelos de Estado da Arte para a estimativa do impacto de riscos.

Os objetos de controle são os códigos-fonte desenvolvidos para o experimento. Os critérios de aleatoriedade, agrupamento e balanceamento para facilitar a análise estatística. O objeto experimental é a base de dados de risco, no nosso caso a PERIL. Por fim, Oo resultados são analisados estatisticamente, portanto um teste de hipótese será conduzido tão logo eles sejam obtidos.

4.1 Regressão Linear Múltipla e Modelo de Regressão em Árvore

O primeiro experimento definido para este trabalho é estabelecer uma linha de base para que seja possível comparar o desempenho de outras abordagens com a base de dados selecionada. Os modelos MRLM e MAR são analisados, aquele que produzir o menor erros de previsão (REMQ) será selecionado.

Nessa análise, o código-fonte para análise dos modelos de regressão linear foram adaptados de Torgo [37] com o objetivo de realizar o treinamento, a validação cruzada, a geração das saídas previstas e o cálculo do REMQ.

4.2 Simulação de Monte Carlo e Análise PERT

O segundo experimento é analisar o desempenho das técnicas convencionais utilizadas na academia e na indústria, inclusive determinadas como boas práticas pelo PMBOK [4]. Como explicado anteriormente, essas abordagens foram configuradas para obterem o melhor desempenho possível.

4.3 Perceptron de Múltiplas Camadas e suas variações

O terceiro experimento tem o objetivo de analisar quais das variações da MLP obtém o menor REMQ de previsão. Há numerosas combinações possíveis de configurações da MLP, no entanto somente um subconjunto delas foi avaliado. A melhor configuração da MLP é utilizada no experimento subsequente.

Esse experimento e o próximo são os experimentos mais significativos desse trabalho. A importância da avaliação de diversas alternativas a MLP tradicional baseada no algoritmo *backpropagation* é de fundamental importância já que nenhum dos trabalhos anteriores refinaram esse estudo. Além disso, eles não investigaram se outras abordagens como RBF e SVM poderiam ter um melhor desempenho.

4.4 MLP, SVM, RBF e ANFIS

O quarto experimento tem o objetivo de eleger qual a melhor técnica baseada em Redes Neurais Artificiais para a previsão do impacto de riscos a partir da base de dados PERIL. As melhores configurações para cada uma das abordagens é selecionada e o REMQ é calculado para cada técnica.

4.5 Validação do Melhor Modelo

Por fim, um teste estatístico dos resultados da melhor abordagem com os oriundos do segundo experimento é realizado para observar se o modelo baseado em Redes Neurais

apresenta maior precisão que os tradicionais. Sendo validada a hipótese nula, de que as redes neurais artificiais são mais precisas e poderiam atender à real necessidade da indústria, conclui-se que através da metodologia proposta nesse trabalho é possível obter uma RNA mais precisa para a estimativa do impacto de riscos.

Capítulo 5

Results

5.1 Regressão Linear Múltipla e Modelo de Regressão em Árvore

O primeiro experimento definido para este trabalho é estabelecer uma linha de base para que seja possível comparar o desempenho de outras abordagens com a base de dados selecionada.

Inicialmente, esse experimento consiste na seleção entre MRLM e MAR como linha de base. Isso pode ser realizado após discutir a informação apresentada na Tabela 5.1 e na Figura 5.1.

A Tabela 5.1 mostra a estatística descritiva do REMQ normalizado para ambos os algoritmos. Os valores de média, desvio padrão, mínimo e máximo são calculados para um Modelo de Regressão Linear Múltipla e três Modelos de Regressão em Árvore: MRLM (cv.lm.v1), MRA_1 (cv.rpart.v1), MRA_2 (cv.rpart.v2) and MRA_3 (cv.rpart.v3).

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

	MRLM	MRA_1	MRA_2	MRA_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

Na Figura 5.1, os *boxplots* da REMQ normalizado após as previsões para MRLM, MRA_1, MRA_2 e MRA_3 são apresentados. O menor REMQ é obtido para o MRLM, esse modelo de regressão também tem menor desvio padrão. A partir dessa informação, pode-se afirmar que o MRLM é um modelo eficiente e preciso e é definido como modelo de linha de base.

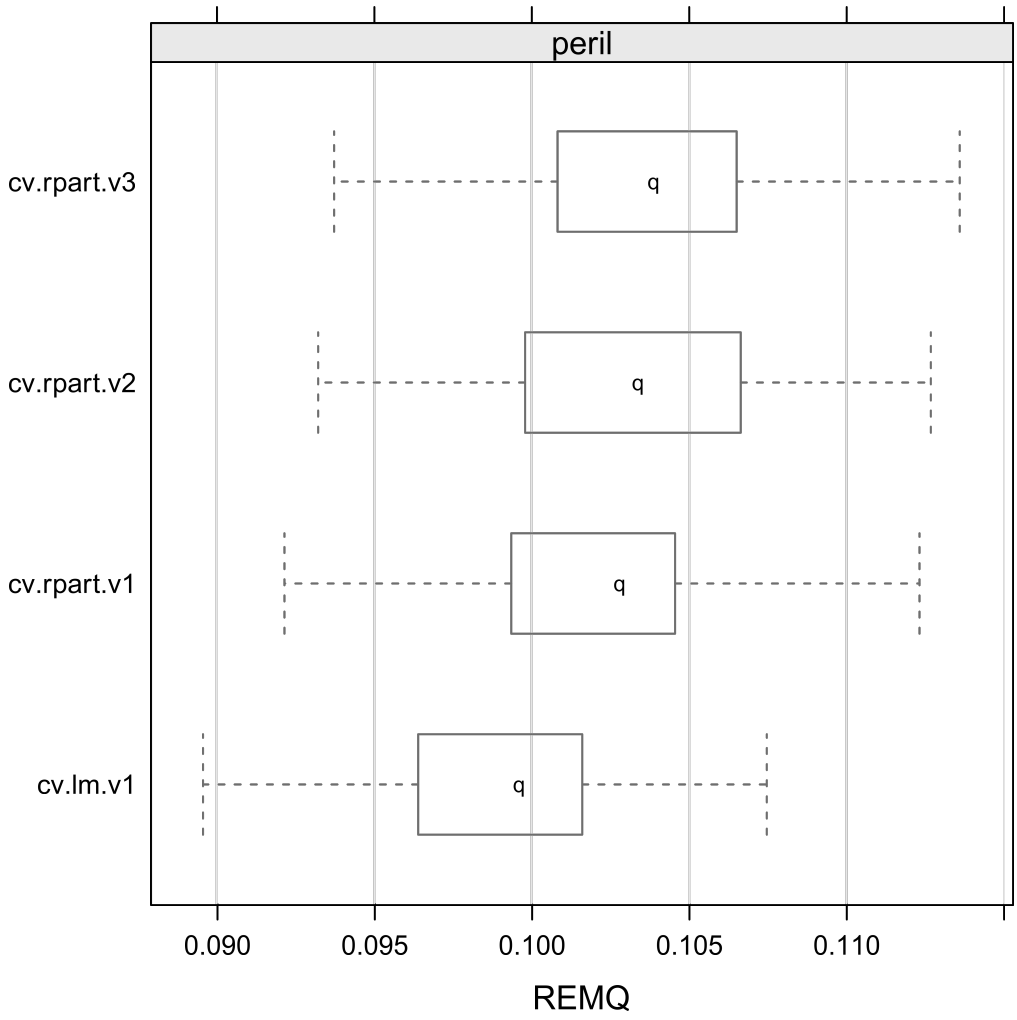

Figura 5.1: Boxplots for normalized errors of linear regression models.

5.2 Simulação de Monte Carlo e Análise PERT

O segundo experimento é analisar o desempenho das técnicas convencionais utilizadas na academia e na indústria, inclusive determinadas como boas práticas pelo PMBOK [4]. Como explicado anteriormente, essas abordagens foram configuradas para obterem o melhor desempenho possível.

A Tabela 5.2 mostra a estatística descritiva do REMQ normalizado para ambos os algoritmos. Os valores de média, desvio padrão, mínimo e máximo, calculados para Simulação de Monte Carlo e para a Análise PERT, mostram que a Análise PERT apresenta valores bem inferiores de média, mínimo e máximo. No entanto, o desvio padrão da Simulação de Monte Carlo é menor, provando ser um método mais preciso.

Na Figura 5.2, os *boxplots* da REMQ normalizado após as previsões para SMC e PERT são apresentados. A partir dessa informação, pode-se afirmar que a Análise PERT

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**

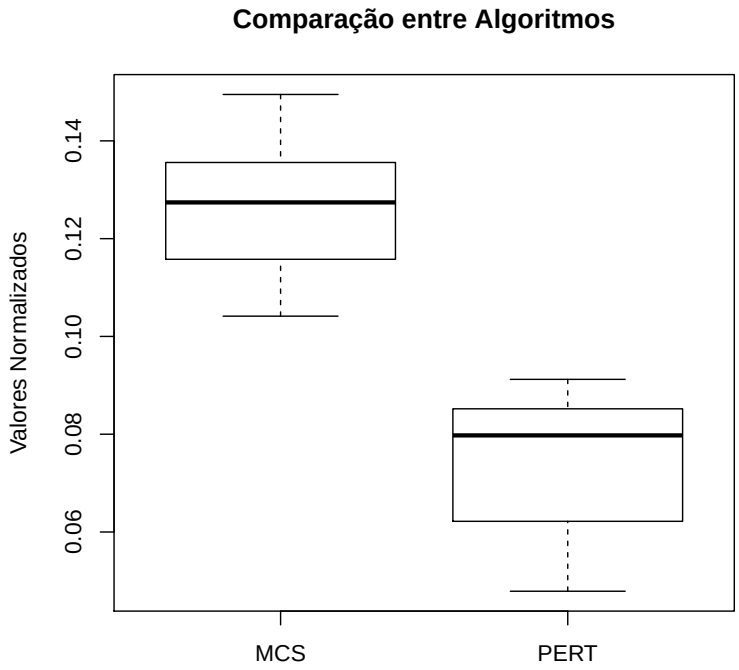

Figura 5.2: Boxplots for normalized errors of linear regression models.

é um modelo de estado da arte mais eficiente e ligeiramente preciso, nessa comparação.

5.3 Perceptron de Múltiplas Camadas e suas variações

Nesse experimento, algumas MLPs foram avaliadas, o principal diferencial entre elas é a regra de aprendizagem. Os algoritmos: *Backpropagation*, Levenberg-Marquardt, Broyden-Fletcher-Goldfarb-Shanno *Backpropagation*, Gradiente Conjugado Escalonado, *Resilient-propagation*, *Resilient-propagation*, *One-step Secant backpropagation* e Quasi-Newton foram alguns dos selecionados.

Na Tabela 5.3, é exibida a estatística descritiva do REMQ normalizado para as abordagens desenvolvidas para esse experimento. Os valores de média, desvio padrão, mínimo e máximo são calculados para cada uma das MLPs. ”BP”apresenta os resultados para uma MLP com algoritmo de aprendizagem *Backpropagation*; ”LM”para uma MLP com o algoritmo Levenberg-Marquardt; ”BFGS”para uma MLP com o algoritmo Broyden-Fletcher-Goldfarb-Shanno *Backpropagation*; ”SCG”para uma MLP com

o algoritmo Gradiente Conjugado Escalonado; ”RP”para uma MLP com o algoritmo *Resilient-propagation*; ”RPCG”para uma MLP com o algoritmo *Resilient-propagation* combinado com o Gradiente Conjugado; ”OSS”para uma MLP com o algoritmo *One-step Secant backpropagation*; por fim, ”Reg”para uma MLP chamada ”MLPRegressor”com o algoritmo BFGS Quasi-Newton. Conforme os valores médio, mínimo e máximo, observa-se que a alternativa ”Reg”é mais eficiente, mesmo tendo o maior desvio padrão segundo esse experimento.

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**

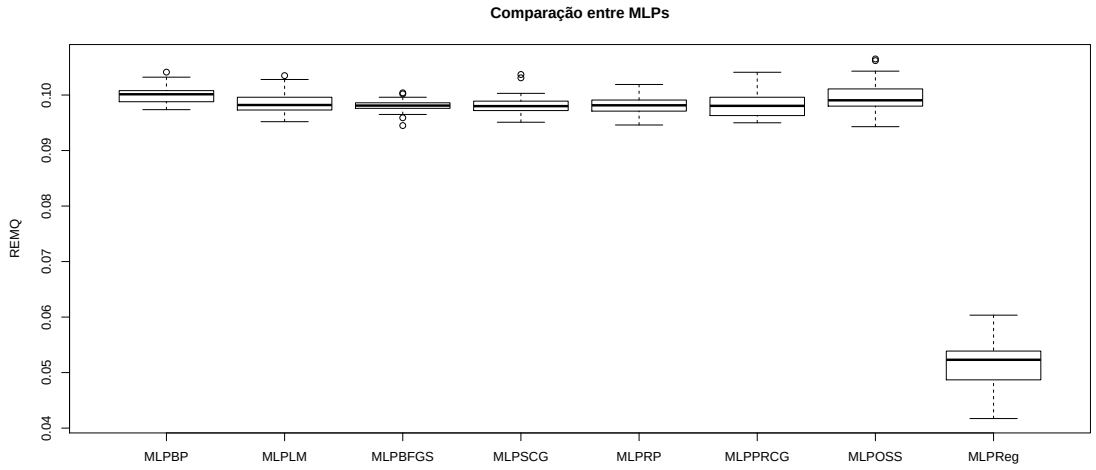

Figura 5.3: Boxplots for normalized errors of multiple linear perceptron networks.

Na Figura 5.3, os *boxplots* da REMQ normalizado após as previsões para as oito MLPs são apresentados. A partir dessas informações, pode-se afirmar que a MLP chamada ”MLPRegressor”é uma rede neural mais eficiente porém imprecisa, de acordo com o experimento.

5.4 MLP, SVM, RBF e ANFIS

Após avaliar uma rede neural eficiente no experimento anterior, o quarto experimento tem o objetivo de eleger qual a melhor técnica baseada em Redes Neurais Artificiais, dentre as implementadas, para a previsão do impacto de riscos a partir da base de dados PERIL.

A Tabela 5.4 mostra a estatística descritiva do REMQ normalizado para as RNAs estudadas. Os valores de média, desvio padrão, mínimo e máximo, calculados para

um modelo Neuro-Fuzzy ANFIS, uma SVM (chamada SMORegressor), uma rede RBF (chamada RBFRegressor) e para uma ”MLPRegressor”mostram que o último algoritmo apresenta valores bem inferiores de média, mínimo e máximo. No entanto, o desvio padrão do ANFIS é menor. 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 5.4: Descriptive statistics for normalized errors of MLPs 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**

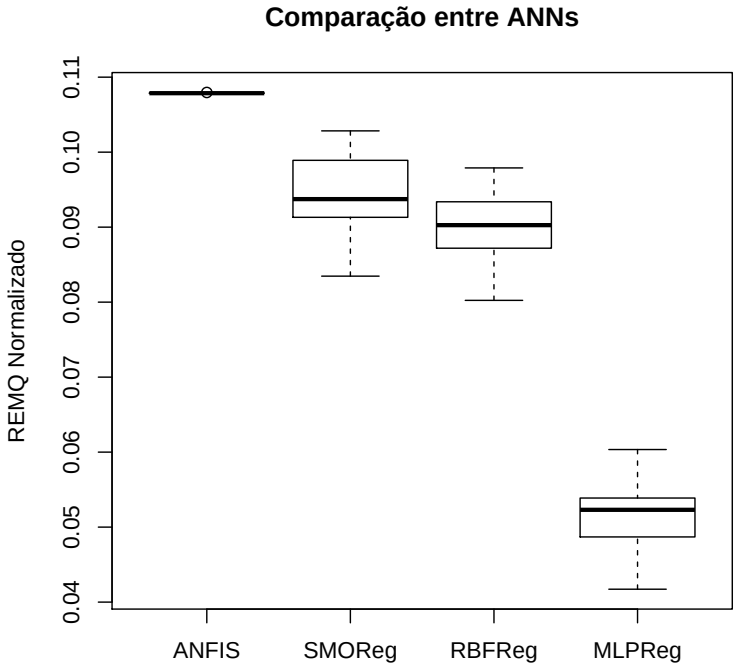

Figura 5.4: Boxplots for normalized errors of multiple linear perceptron networks.

Na Figura 5.4, os *boxplots* da REMQ normalizado após as estimativas de impacto das RNAs são apresentados. A partir dessas informações, pode-se afirmar que o ”MLPRegressor”é uma Rede Neural Artificial mais eficiente e ligeiramente precisa, de acordo com essa comparação.

Na Figura 5.5, é apresentada a convergência do ANFIS, em termos da REMQ normalizado, para cada época durante o treinamento até ser interrompido, de acordo com a validação cruzada. Observa-se que o algoritmo encontra um mínimo local e permanece preso até o treinamento ser interrompido. Logo, conclui-se que o ANFIS apresenta algumas limitações de convergência para essa base de dados.

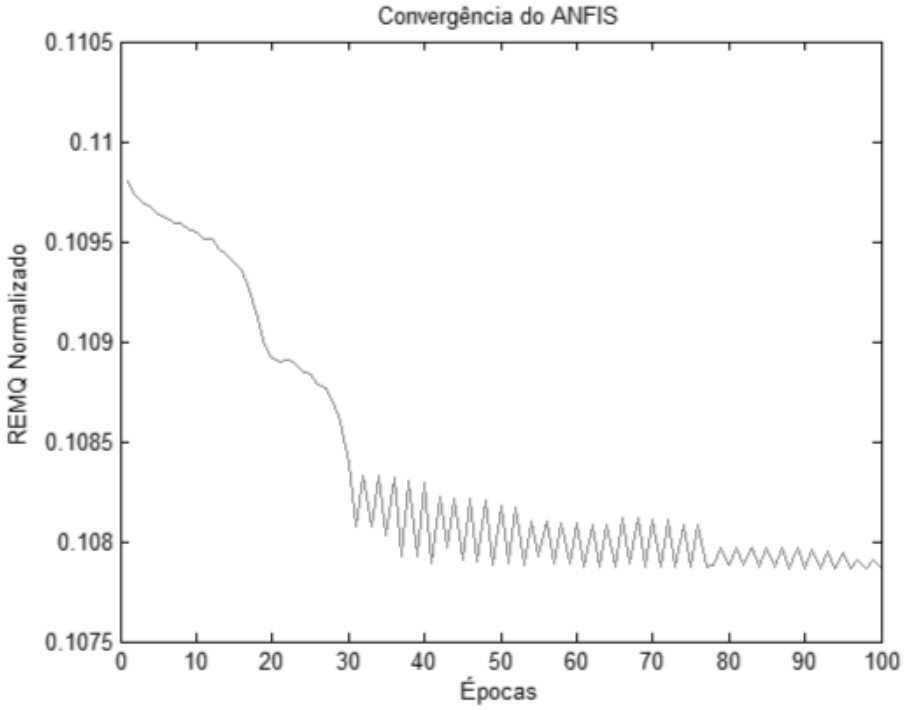

Figura 5.5: Boxplots for normalized errors of multiple linear perceptron networks.

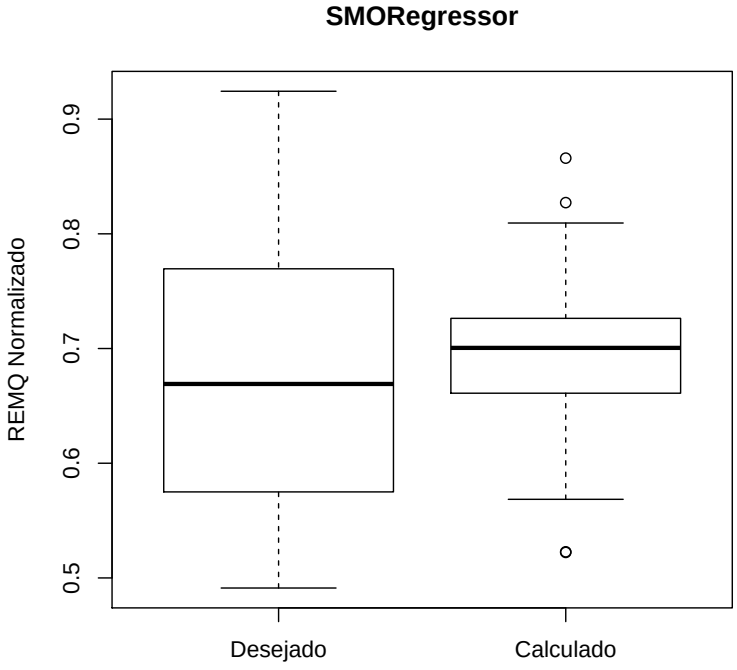

Figura 5.6: Boxplots for normalized errors of multiple linear perceptron networks.

Na Figura 5.6, os *boxplots* dos impactos esperados e calculados pelo algoritmo ”SMORegressor”de cinqunta amostras são apresentados. O resultado ideal é que os

boxplots das duas amostras sejam o mais semelhantes possível, respeitando a mediana, o intervalo inter-quartil e os valores de mínimo e máximo. A partir da informação apresentada, pode-se concluir que o *boxplot* oriundo dos impactos calculados tenta representar mas distorce em todas as medidas os impactos esperados.

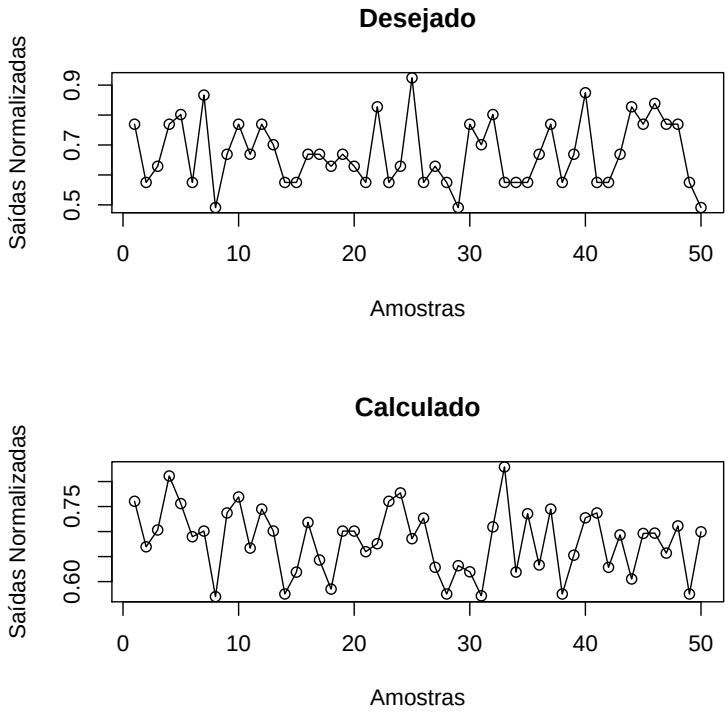

Figura 5.7: Boxplots for normalized errors of multiple linear perceptron networks.

Na Figura 5.7, as cinquenta primeiras amostras do subconjunto de testes e da saídas calculadas são apresentadas, graficamente. A partir desses gráficos, pode-se observar que o algoritmo ”SMORegressor”tem dificuldade em estimar os resultados esperados, quase não apresentando relação com eles.

Na Figura 5.8, os *boxplots* dos impactos esperados e calculados pelo algoritmo ”RBFRegressor”de cinqunta amostras são apresentados. O resultado ideal é que os *boxplots* das duas amostras sejam o mais semelhantes possível, respeitando a mediana, o intervalo inter-quartil e os valores de mínimo e máximo. A partir da informação apresentada, pode-se concluir que o *boxplot* oriundo dos impactos calculados tenta representar mas distorce no quartil inferior e nos valores máximos os impactos esperados.

Na Figura 5.9, as cinquenta primeiras amostras do subconjunto de testes e da saídas calculadas são apresentadas, graficamente. A partir desses gráficos, pode-se observar que o algoritmo ”RBGRegressor”tem dificuldade em estimar os resultados esperados, principalmente os valores máximos e mínimos.

Na Figura 5.10, os *boxplots* dos impactos esperados e calculados pelo algoritmo ”MLPRegressor”de cinquenta amostras são apresentados. O resultado ideal é que os *boxplots* das duas amostras sejam o mais semelhantes possível, respeitando a mediana,

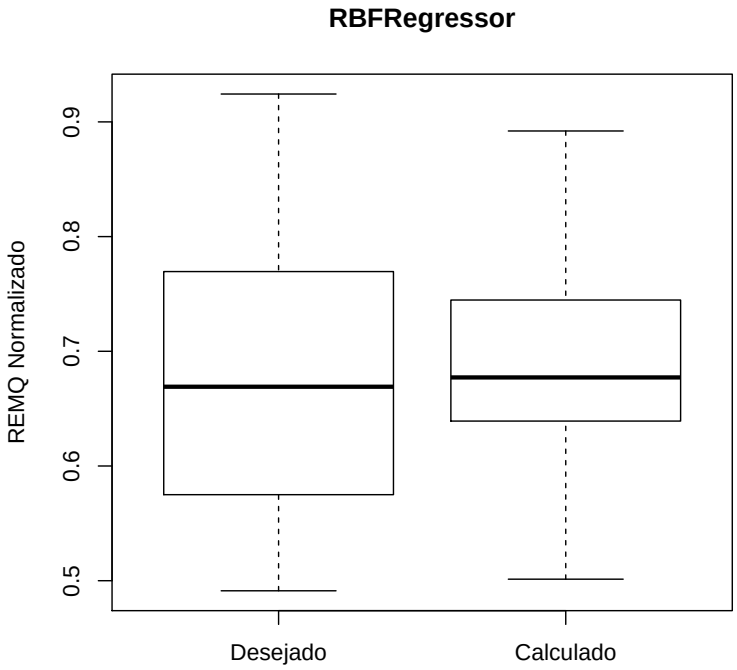

Figura 5.8: Boxplots for normalized errors of multiple linear perceptron networks.

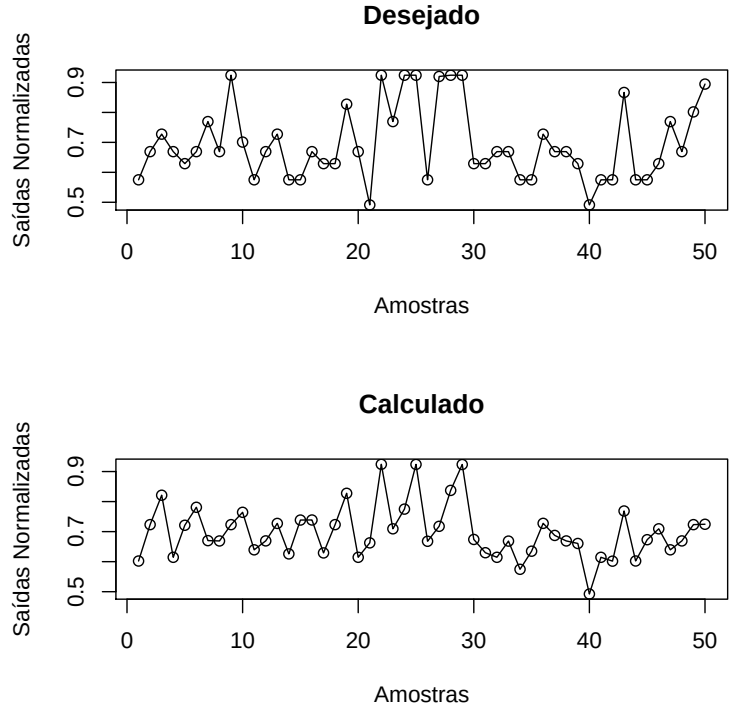

Figura 5.9: Boxplots for normalized errors of multiple linear perceptron networks.

o intervalo inter-quartil e os valores de mínimo e máximo. A partir da informação apre-

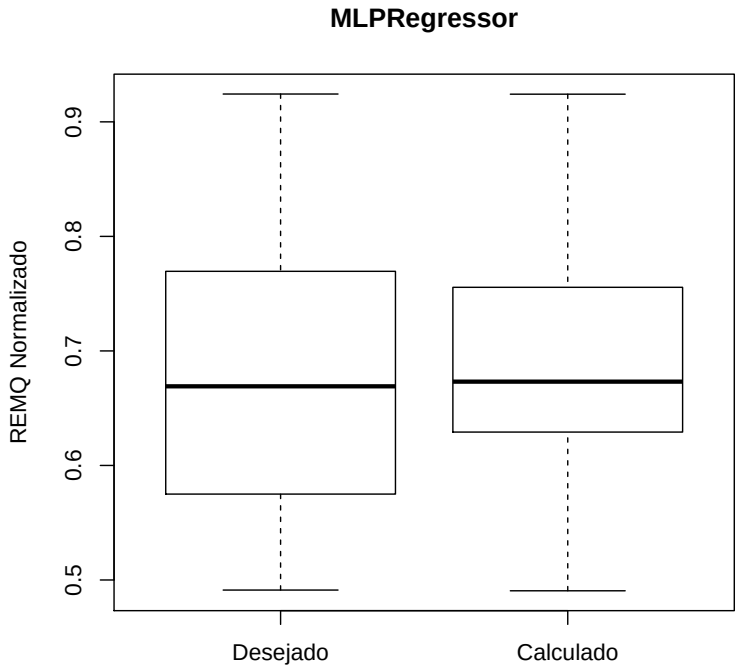

Figura 5.10: Boxplots for normalized errors of multiple linear perceptron networks.

sentada, pode-se concluir que o *boxplot* oriundo dos impactos calculados representa, porém com distorções principalmente no quartil inferior, os impactos esperados.

Na Figura 5.11, as cinquenta primeiras amostras do subconjunto de testes e da saídas calculadas são apresentadas, graficamente. A partir desses gráficos, pode-se observar que o algoritmo ”MLPRegressor”tem dificuldade em estimar os resultados esperados, porém tende a acompanhar os resultados.

5.5 Validação do Melhor Modelo

A Tabela 5.6 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.

Na Figura 5.12, os *boxplots* da REMQ normalizado após as estimativas de impacto das RNAs são apresentados. A Simulação Monte Carlo(SMC) é imediatamente descartada da análise por apresentar erros maiores que o Modelo de Regressão Linear Múltipla (MLR); a Análise PERT (PERT) apresenta-se como uma abordagem simples e rápida para a estimativa do impacto de riscos, porém como é uma técnica estatística está sujeita a algumas situações que geram maiores erros na estimativa; já, o MLPRegressor (MLPReg) apresenta os menor valores de média, desvio padrão, mínimo e máximo de erro.

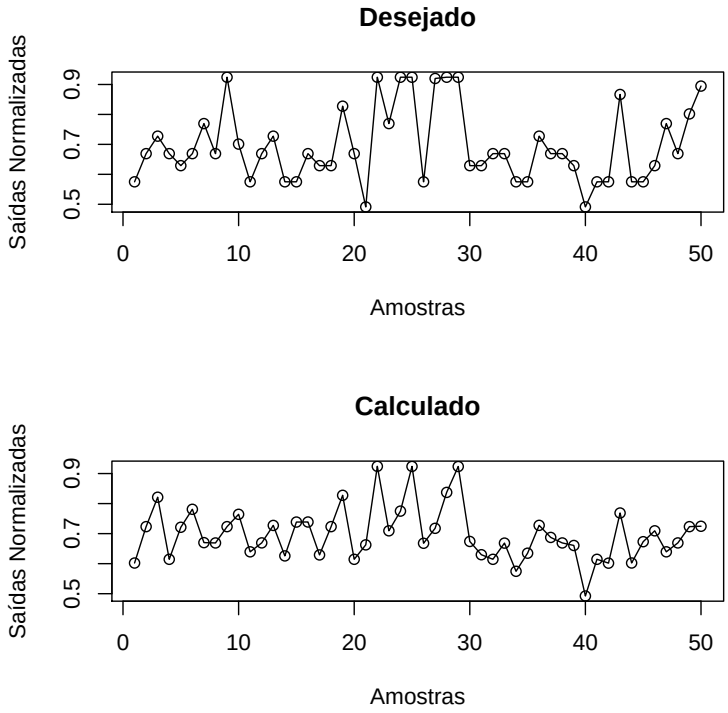

Figura 5.11: Boxplots for normalized errors of multiple linear perceptron networks.

Tabela 5.5: Descriptive statistics for normalized errors of MLPs 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**

A partir dessas informações, pode-se afirmar que o ”MLPRegressor”é uma Rede Neural Artificial mais eficiente e precisa, comparada com o desempenho das outras RNAs.

A Tabela 5.4 apresenta os resultados dos Testes de Wilcoxon não-pareados para as RNAs estudadas. O símbolo Δ significa que a técnica à esquerda apresenta melhores resultados que a acima; diferentemente, o símbolo ∇ siginifica que a técnica acima apresenta melhores resultados que à esquerda. Após analisar a tabela, podemos afirmar que a ”MLPRegressor”é melhor que a Análise PERT que, por sua vez, é melhor que o Modelo de Regressão Linear Múltipla.

5.6 Previsão do Impacto e Definição do Intervalo de Confiança

O último experimento consiste na aplicação prática da metodologia proposta nesse estudo, com base no PERIL. Como alguns passos descritos na metodologia foram seguidos nos experimentos anteriores, resta somente a geração de um intervalo de confiança

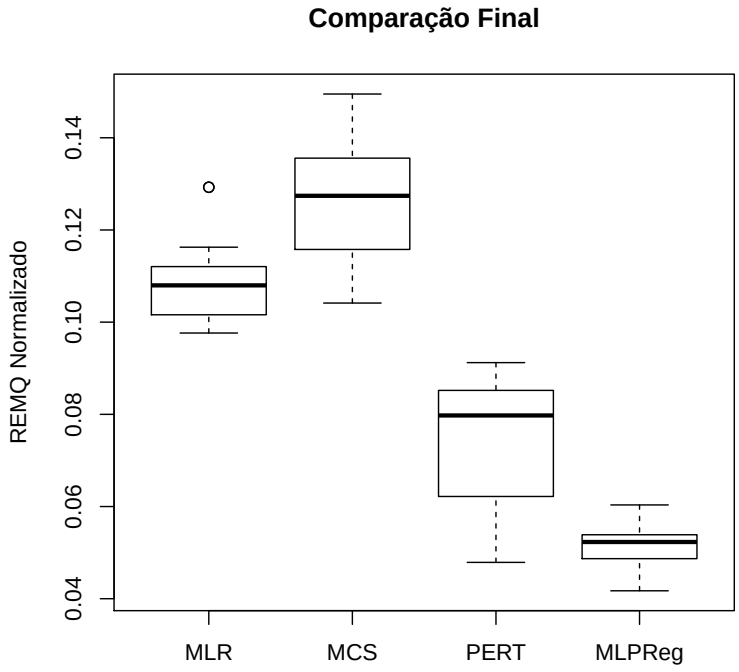

Figura 5.12: Boxplots for normalized errors of multiple linear perceptron networks.

Tabela 5.6: Descriptive statistics for normalized errors of MLPs models.

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

que é a informação compreendida pelos gerentes de projetos, analistas de projetos e de riscos.

A aplicação do intervalo de confiança irá determinar a qualidade dos resultados gerados. A partir dos resultados apresentados pelo modelo, será gerado um intervalo que, para uma determinada previsão, terá 95 % de chances de conter o valor real. Para isso, será utilizado o método de máxima verossimilhança.

O método de máxima verossimilhança considera que existem duas fontes de incertezas para um modelo de previsão. O primeiro, o σ_v, é a variância do ruído, e o segundo, σ_w, é a variância da incerteza. O σ_v representa a variância dos erros gerados pelo conjunto de validação cruzada na fase de treinamento. Esses valores seguem uma distribuição normal e possuem média zero. Já o σ_w é referente à variância de incerteza do modelo e é calculada a partir da utilização do modelo para prever os erros gerados pela própria rede.

Assume-se que essas duas fontes de erro são independentes. Então o cálculo da variância total do modelo é dado pela Equação 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.

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

onde, n é a quantidade de valores; $Erro_i$ é o erro referente à entrada i; $\overline{Erro}$ é a média dos erros.

Para calcular o σ_w, armazena-se is erros para todas as entrada utilizadas no treinamento da rede, incluindo dados de treinamento e validação cruzada. com esses dados devidamente armazenados e normalizados, cria-se um novo modelo, com novos pesos e novas ligações. Esse modelo terá como objetivo, ou seja, valores desejados, os erros gerados pelo modelo anterior. O σ_w será calculado utilizando a mesma fórmula utilizada para o σ_v. A diferença está na quantidade de dados, pois, para o σ_w serão utilizados todos os erros de todos os conjuntos, não só o de validação cruzada.

Tendo o σ_v e σ_w calculados, pode-se calcular o σ_total. Este será utilizado para calcular o intervalo de confiança a partir da Equação 5.3.

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

onde, x é o valor calculado pela rede e t é o valor extraído da tabela de t de Student para o maior grau de liberdade possível para um intervalo de 95% de chances de conter o valor real, como exibido na Tabela 5.1.

Tabela 5.7: Descriptive statistics for normalized errors of MLPs models.

$P(t_n \leq x)$				
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 paper has investigated the use of Artificial Neural Networks algorithms, like SVM and MLP, for estimation of risk impact in software project risk analysis. We have carried out a statistical analysis using PERIL. The results were compared to MLRM and Monte Carlo Simulation, a traditional approach proposed by [4]. We have considered improving risk impact estimation accuracy during software project management, in terms of MAE mean and standard deviation. We have observed that MLP had minor standard deviation estimation error, and showed to be a promissory technique. Moreover, SVM had minor estimation error outcomes using PERIL. 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.