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, Julho de 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, Julho de 2014

Agradecimentos

Agradeço a Deus pela oportunidade da vida para estar me desenvolvendo pessoalmente e profissionalmente. Agradeço a minha família, aos meus amigos e, em especial a Isabelle, pelo apoio e contribuição para que eu possa estar desempenhando tantas atividades ao mesmo tempo.

Agradeço a Escola Politécnica de Pernambuco, aos meus professores e aos meus colegas de sete anos de convivência e experiências marcantes. Em especial, agradeço ao meu professor orientador, Prof. Dr. Sérgio Murilo, por estar me apoiando e colaborando positivamente nos momentos acadêmicos mais marcantes da minha carreira, como a orientação do trabalho de conclusão de curso de graduação e a orientação para o desenvolvimento da minha pesquisa e da escrita desta dissertação. Agradeço também ao meu professor coorientador, Prof. Dr. Mêuser Valença, pelas discussões, pelos ensinamentos e pela disposição em colaborar com o desenvolvimento da minha pesquisa.

Agradeço a todas pessoas que conviveram comigo nos últimos quatro anos que desejarem o meu crescimento, que apoiaram a realização dos meus sonhos. Vocês me ajudaram a me tornar mais sábio e me proporcionarem aprender com ótimos exemplos de vida.

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

Jorge da Capadócia
Jorge Ben Jor

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

Roger W. Kivell
WWII Navy Frogman (Navy Seal)

Resumo

Muitos projetos de desenvolvimento de software finalizam com falha. Segundo estudos já realizados, somente um quarto dos projetos de software têm sucesso imediato, e bilhões de dólares são perdidos anualmente por meio de falhas ou projetos que não cumprem a entrega dos benefícios prometidos. Além disso, ainda há algumas dificuldades na interpretação do conceito de risco e na aplicação de métodos para a análise de risco de forma eficiente e precisa, conforme descrito nesta dissertação. Isso ocorre devido a vários fatores: falta de habilidade para manusear as ferramentas existentes; falta de conhecimento dos benefícios obtidos pela gestão de risco, bem como as limitações encontradas nas ferramentas sugeridas como boas práticas; a ausência de estudos na indústria que comprovem o valor de cada técnica e a dificuldade na obtenção de uma base de dados de riscos adequada.

Na Simulação de Monte Carlo, por exemplo, as simulações podem levar a resultados enganosos se entradas inapropriadas, derivadas da parametrização subjetiva, forem inseridas no modelo. Além disso, como ela não pode modelar correlações entre riscos, então os números que surgem em cada sorteio são aleatórios. Na análise PERT, por sua vez, como há necessidade da opinião especializada de um especialista, basta inserir valores de entrada errôneos para gerar resultados incoerentes.

Nesse trabalho, é utilizada uma base de dados especial, chamada PERIL, e é proposta uma metodologia para análise de risco cujo objetivo é encontrar uma Rede Neural Artificial que seja mais eficiente e precisa que as técnicas convencionais até então utilizadas, independentemente da base de dados a ser empregada. Perceptron de Múltiplas Camadas, Máquina de Vetor de Suporte, Redes com Função de Base Radial, um Sistema Adaptativo *Neuro-Fuzzy*, Modelos de Regressão Linear, Simulação de Monte Carlo e Análise PERT foram as técnicas avaliadas. Em seguida, são conduzidos experimentos para determinar qual o método mais eficiente e preciso para a estimativa do impacto de risco, baseado em informações da PERIL. Por fim, um intervalo de confiança pôde ser obtido para uma determinada previsão.

Os resultados mostram que uma das variações da MLP obteve melhores resultados que qualquer outra técnica, conforme se esperava. Além disso, comprova-se que a Simulação de Monte Carlo apresenta os piores resultados ainda que se compare com modelos de regressão linear simples. Por fim, a análise PERT mostra ser bastante eficiente quando se utiliza a opinião especializada e uma base de dados de riscos como a PERIL.

Palavras-chave:
Gerenciamento de Projetos, Análise de Riscos, PERIL, Redes Neurais Artificiais, Simulação de Monte Carlo, Análise PERT.

Abstract

Many software projects end in failure. According to previous studies, only a quarter of software projects have immediate success, and billions of dollars are lost annually through failures or projects that do not meet the delivery of promised benefits. Moreover, there are still some difficulties in the interpretation of the concept of risk and application of methods for risk analysis efficiently and accurately. This is due to several factors: lack of ability to handle existing tools; lack of knowledge of the benefits from risk management as well as limitations found in the tools suggested as good practice; the lack of studies in the industry proving the value of each technique and the difficulty in obtaining a database of proper risk.

In Monte Carlo simulation, for example, simulations can give misleading results if inappropriate inputs, derived from subjective parameterization, are incorporated into the model. Besides, as it can not model correlations between risks, then the numbers appearing in each drawing are random. In PERT analysis, in turn, as there is a need for specialized opinion from an expert, just enter erroneous input values to generate inconsistent results.

In this work, a special database, called PERIL is used, and is proposed a methodology for risk analysis whose goal is to find an Artificial Neural Network which is more efficient and accurate than standard techniques previously used, regardless of the database to be employed. Multilayer Perceptron, Support Vector Machine, Radial Basis Function Networks, an Adaptive Neuro-Fuzzy System, Linear Regression Models, Monte Carlo Simulation and PERT Analysis techniques were evaluated. Then, experiments are conducted to determine which is a more efficient and accurate method for risk impact estimation based on PERIL. Finally, a confidence interval could be obtained for a given prediction.

The results show that a MLP variation has better results than any other technique, as expected. Furthermore, it was proven that Monte Carlo Simulation shows the worst results still compared with Linear Regression Models. Finally, PERT analysis shows to be quite efficient when using expert judgment and a risk database as PERIL.

Keywords:
Project Management, Risk Analysis, PERIL, Artificial Neural Networks, Monte Carlo Simulation, PERT Analysis.

Sumário

Lista de Figuras

Lista de Siglas

PERT	*Program Evaluation and Review Technique*
PERIL	*Project Experience Risk Information Library*
MLP	*Multilayer Perceptron*
SVM	*Support Vector Machine*
RBF	*Radial Basis Function*
ANFIS	*Adaptive Neuro-Fuzzy Inference System*
RNA	Rede Neural Artificial
PSO	*Particle Swarm Optimization*
PMBOK	*Project Management Body of Knowledge*
SMC	Simulação de Monte Carlo
MRLM	*Modelo de Regressão Linear Múltipla*
NFS	*Neuro-Fuzzy System*
REMQ	Raiz do Erro Médio Quadrático
MRL	Modelo de Regressão Logística
EVM	Valor Monetário Agregado
ADALINE	*Adaptive Linear Neuron*
ANN	*Artificial Neural Network*
GD	Gradiente Descendente
LMS	*Least Mean Square*
MADALINE	Múltiplos Neurônios Lineares Adaptativos
GCS	Gradiente Conjugado Escalonado
BFGS – BP	*Broyden-Fletcher-Goldfarb-Shanno Backpropagation*
OSS – BP	*One Step Secant Backpropagation*
AIC	*Akaike Information Criterion*

Lista de Símbolos

τ_e	Duração estimada de uma atividade	13
σ^2	Variância da duração da atividade	14
α	Coeficiente de regressão independente	15
β	Coeficiente de regressão dependente	15
net_i	Entrada líquida do neurônio i	17
$f(net_i)$	Função de ativação do neurônio i	17
δ_i^m	Sensibilidade para os neurônios i da camada m	19
$w_{ij}^m(t+1)$	Peso entre par de neurônios (i, j)	19
$R_{funcional}$	Risco funcional do modelo SVM	21
$R_{estrutural}$	Risco estrutural do modelo SVM	21
d	Distância entre planos na SVM	22
$\phi_i(x)$	Funções de ativação para redes RBF	23
y_j	Função de resposta na camada de saída da RBF	24
F	Função objetivo do algoritmo K-Médias	24
μ_i	Média dos pontos de um subconjunto no K-Médias	24
ε	Função de erro no GD	25
$v_i(t)$	Equação de atualização de pesos no GD	26
Δw_{ij}	Regra de aprendizagem por correção de erro	26
$d(x_i, x_{teste})$	Distância euclidiana entre x_i e x_{teste}	27
$\varepsilon_T(D_T, w)$	Função erro dos pesos da solução no SCG	28
$F(x_{k+1})$	Função de minimização na aprendizagem do BFGS-BP	29
p_{k+1}	Nova direção de busca no OSS-BP	29
$\varepsilon_T(D_T, w)_{LM}$	Função de erro na aprendizagem por LM	30
$O_{1,i}$	Grau de pertinência para a camada 1	31
$\mu_A(x)$	Função de pertinência ao conjunto *fuzzy A*	31
$O_{2,i}$	Grau de pertinência para a camada 2	32
$O_{3,i}$	Grau de pertinência para a camada 3	32
$O_{4,i}$	Grau de pertinência para a camada 4	32
$O_{5,i}$	Saída geral do ANFIS	32
σ_v	Variância dos erros de validação	62

Lista de Algoritmos

Lista de Tabelas

Capítulo 1

Introdução

Quão arriscados são os projetos de software? Diversos estudos sobre a efetividade de técnicas de previsão de custo, escopo, cronograma; *surveys* com profissionais em *software* na indústria; e análise de portfólio de projetos foi realizada para responder essa questão [2]. No entanto, não há um consenso.

Todos os projetos envolvem risco. Há sempre pelo menos algum nível de incerteza no resultado de um projeto, independentemente do que o gráfico de Gannt pareça indicar. Projetos de alta tecnologia são particularmente arriscados, por uma série de motivos. Primeiro, há uma grande variedade de projetos técnicos. Esses projetos tem objetivos e aspectos únicos que os diferem significativamente dos trabalhos anteriores, além de apresentar um ambiente de projetos que evolue rapidamente. Além disso, projetos técnicos são frequentemente ”enxutos”, ou seja, desafiados a trabalhar com financiamento, pessoal e equipamentos inadequados. Para piorar a situação, há uma expectativa generalizada que não corresponde à realidade de que por mais rápido que tenha sido o último projeto, o próximo deve ser ainda mais rápido [3].

Projetos que tiveram sucesso geralmente conseguiram isso porque duas das ações tomadas pelos seus líderes foram determinantes. Primeiro, eles reconheceram que alguns dos trabalhos em qualquer projeto, mesmo projetos de alta tecnologia, não são novos. Nos trabalhos por eles desenvolvidos as notas, registros e lições aprendidas em projetos anteriores puderam ser utilizados como um roteiro para identificar, e em muitos casos evitar, muitos problemas potenciais. Segundo, eles planejaram com afinco o trabalho do projeto, especialmente as partes que exigiam inovação, para possibilitar a compreensão dos desafios futuros e antecipar muitos dos riscos [3].

Alguns benefícios da boa gestão de riscos de projetos de software são apresentados abaixo. Tais fatores podem determinar o sucesso dos projetos [4] [5].

- redução de custos incorridos com mudanças no software;
- desenvolvimento de um plano de respostas a eventos inesperados, conhecido como plano de contingência de riscos;
- previsão da probabilidade da ocorrência de eventos indesejados;
- seguimento das linhas de base de custo, cronograma e qualidade.

1.1 Motivação

Em 2009, o CHAOS Report [6] mostrou que 32% dos projetos de software alcançaram sucesso, foram entregues no prazo, de acordo com o orçamento estabelecido e com os requisitos prometidos; 44% dos projetos foram desafiados, em que o prazo ou o orçamento ou os requisitos não foram cumpridos; não menos importante, 24% dos projetos falharam e foram cancelados. Isso ocorre devido aos riscos envolvidos nas atividades do projeto e a um gerenciamento de risco de software ausente ou deficiente [7].

Schmidt e outros autores [8] notaram que muitos dos projetos de desenvolvimento de *software* terminavam com falha. Eles mostraram que cerca de 25% de todos os projetos de software são cancelados e cerca de 80% de todos os projetos de software ultrapassaram seus orçamentos, excedendo-os em 50% na média. Paul Bannerman [9] afirma que pesquisas na indústria sugerem que somente um quarto dos projetos de software tem sucesso imediato, e bilhões de dólares são perdidos anualmente por meio de falhas ou projetos que não cumprem a entrega dos benefícios prometidos. Além disso, o autor mostra evidências de que isso é um assunto global, impactando organizações do setor privado e público [10].

A previsão de possíveis eventos a curto, médio e longo prazos muitas vezes é falha. Ao analisar os riscos e as incertezas, os gerentes de projeto comumente confiam na própria intuição, em vez de utilizarem a lógica e uma análise detalhada. No entanto, o pensamento intuitivo é frequentemente alvo de ilusões, que causam erros mentais previsíveis e decisões eventualmente não embasadas. O método para conciliar o efeito dessas ilusões psicológicas é uma avaliação sistemática dos riscos e esforços na mitigação dos mesmos através de métodos analíticos.

É difícil gerenciar algo que não pode ser medido. Gerentes de projeto devem quantificar a probabilidade de risco, os resultados, e seu efeito cumulativo em um projeto. Além disso, é importante avaliar as várias opções de mitigação: o custo de cada opção e o tempo necessário para a sua realização [11].

Existe uma dificuldade na interpretação do conceito de risco, principalmente quanto a aplicação desse conhecimento no desenvolvimento e utilização de técnicas eficientes para a análise de risco no gerenciamento de projetos de *software*. A gestão de riscos e incertezas em projetos de software, é fundamental para a disciplina de gerenciamento de projetos. Entretanto, em momentos econômicos de crise torna-se muito mais difícil realizar o gerenciamento de riscos, devido aos custos incorridos.

Esta é uma área de pesquisa que vem evoluindo e, portanto, novas e melhores metodologias para identificação, medição e controle de itens de risco de *software* precisam ser desenvolvidas. Segundo Keshlaf e Riddle [12], mesmo que existam muitas abordagens ainda há uma grande lacuna com relação ao que é praticado pelas indústrias de *software*.

1.2 Descrição do Problema

Embora o gerenciamento de risco na gestão de projetos de *software* seja um processo saudável, sua utilização ainda está aquém das expectativas. Algumas causas disso são o acúmulo de responsabilidades dos gerentes de projetos, a baixa importância atribuída a essa área, a falta de conhecimento em gestão de riscos, os custos envolvidos nas atividades de gestão de risco, a falta de habilidade para lidar com as técnicas e ferramentas específicas. Como consequência, o projeto está sujeito à influência negativa de riscos sem haver um plano de contingência, o que pode ocasionar o fracasso do projeto. Conforme identificado por Kwak e Ibbs [13], o gerenciamento de risco é a disciplina menos praticada dentre as diferentes áreas do conhecimento no gerenciamento de projetos. Os autores mencionam que, provavelmente, um motivo é que desenvolvedores de software e gerentes de projetos consideram gerenciar processos e atividades que envolvam incerteza como trabalho e custo extras. De acordo com o *benchmarking* realizado em 2009 pelo Project Management Institute, em 20% dos projetos os seus gerentes não realizam todos os processos de planejamento e em apenas 35% dos projetos o gerenciamento de riscos é realizado de acordo com uma metodologia formal, estruturada por políticas, procedimentos e formulários. Além disso, 46% dos gerentes realizam atividades de gerenciamento em tempo parcial.

Barry Boehm [14] definiu risco como a possibilidade de perda ou dano. Essa definição pode ser expressa pela fórmula de exposição ao risco. Mesmo que Boehm cite a exposição ao risco como a técnica mais efetiva para a priorização do risco depois de sua análise, Paul Bannerman [9] considera essa definição limitada e inapropriada. Na teoria clássica da decisão, risco reflete a variação na distribuição de probabilidade de possíveis resultados, seja negativo ou positivo, associado a uma decisão particular. Esse estudo leva em consideração a definição do Project Management Institute [5] em que risco em projeto é um evento ou condição específica que, se ocorrer, tem um efeito positivo ou negativo em um ou mais objetivos do projeto. Uma definição complementar proposta por Haimes [15] também é considerada, a qual expressa o risco como uma medida da probabilidade e severidade de efeitos adversos.

Um fator de risco é uma variável associada com a ocorrência de um evento inesperado. Fatores de risco são correlacionados, não necessariamente causais e, se um deles ocorrer, implicará em um ou mais impactos. De acordo com Haimes [15], riscos podem comumente surgir como o resultado não apenas de um processo estocástico não percebido ocorrendo no tempo e no espaço, mas também, baseado em fatores de riscos determinísticos. A estimativa do risco pode ser alcançada baseada em informações históricas ou conhecimento de projetos anteriores similares, ou ainda de outra fonte de informação [5].

Risco é um conceito que muitos consideram difícil de ser compreendido por envolver duas métricas complexas: uma medida da probabilidade e da severidade de efeitos adversos [16]. Uma limitação dessa definição é a dificuldade prática em se estimar a probabilidade e o impacto de diversos fatores de risco, especialmente em projetos de *software*. As probabilidades somente podem ser definidas de um modo preciso para atividades que são repetidas muitas vezes sob circunstâncias controladas. No entanto, a natureza única de muitas atividades de projetos de *software* não permite a estimativa

precisa de suas probabilidades. Outra limitação dessa definição é que ela abrange somente ameaças conhecidas ou previsíveis, oferecendo opções limitadas para gerenciar ameaças não percebidas, além de não reconhecer ameaças imprevisíveis. Essa é uma consequência da definição de risco em termos de probabilidade e impacto; uma vez que para se avaliar a probabilidade e o impacto é necessário ter a capacidade de se prever uma eventualidade. Existe ainda uma outra questão em que se questiona se as melhores decisões são baseadas na quantificação numérica, determinada pelos padrões do passado, ou na avaliação subjetiva das incertezas. Não é possível quantificar o futuro com certeza, mas através da probabilidade, é possível prevê-lo a partir do passado. Apesar de ser difícil encontrar um projeto de *software* padrão, é possível classificar atividades e definir padrões que possibilitem a estimativa. Para Bannerman [9], a solução comum para esse problema em projetos de *software* consiste em observar o risco de um modo mais geral, em termos da incerteza, e avaliá-lo qualitativamente [9].

Haimes [16] considera duas premissas na pesquisa de análise de riscos, que também serão consideradas neste estudo. Uma, que o risco é comumente quantificado através da fórmula matemática de expectativa. No entanto, mesmo que essa fórmula permita uma medida valiosa do risco, ela falha em reconhecer e/ou acentuar consequências de eventos extremos. Tom Kendrick apresenta no seu livro [1] um *framework* para a identificação e gerenciamento de catástrofes. A outra premissa, por sua vez, afirma que uma das tarefas mais difícil em análise de sistemas é saber como modelar o risco. Portanto, novas propostas para a análise quantitativa e modelagem de sistemas sob o ponto de vista de seus riscos, poderão contribuir para o avanço científico da área.

A necessidade de gerenciar riscos (eventos indesejados) cresceu exponencialmente com a complexidade dos sistemas. Gerenciar esses eventos nesses sistemas complexos torna difícil identificar e estimar a ocorrência de eventos indesejados esperados ou inesperados por conta da imensa quantidade de fatores de riscos envolvidos e suas relações. Há uma necessidade crescente por métodos mais sistemáticos e ferramentas para suplementar o conhecimento individual, julgamento e experiência. Essas características humanas são muitas vezes suficientes para enfrentar riscos de menor complexidade e isolados. Por exemplo, uma parte dos problemas mais sérios encontrados na aquisição de um sistema são os resultados de riscos que são ignorados, devido a sua baixa probabilidade, até que eles já tenham criado consequências mais sérias [17].

O Guia de Boas Práticas para o Gerenciamento de Projetos, PMBOK [5], apresenta a Simulação de Monte Carlo (SMC) como uma boa prática para a análise de risco de projetos. No entanto, existem algumas limitações na adoção dessa abordagem que o torna inviável [18]. As simulações podem levar a resultados enganosos se entradas inapropriadas, derivadas da parametrização subjetiva, são inseridas no modelo. Comumente, o usuário deve estar preparado para realizar os ajustes necessários se os resultados que são gerados parecem fora de rumo. Além disso, Simulação de Monte Carlo não pode modelar correlações entre riscos. Isso significa que os números que surgem em cada sorteio são aleatórios e, em consequência, um resultado pode variar de seu valor mais baixo, em um período, para o mais alto no próximo. Portanto, abordagens alternativas devem ser consideradas para prever a probabilidade de risco e impacto, levando em consideração as características de risco do projeto e as limitações da Simulação de Monte Carlo. Assim, a análise de risco deve ser uma tarefa mais precisa e mais fácil, do ponto de vista

dos usuários. Esse trabalho considera Redes Neurais Artificial (RNAs) uma alternativa valiosa a ser considerada na análise de risco de projetos de *software*.

1.3 Objetivos

1.3.1 Questões de Pesquisa

Como analisar os riscos no gerenciamento de projetos de *software*, também considerando as catástrofes?
Como analisar quantitativamente os riscos no gerenciamento de projetos de *software*?
Quais dados de registros de riscos de projetos de *software* estão disponíveis para realizar os estudos?
Como desenvolver um método para previsão de riscos em gerenciamento de projetos de *software* eficiente para o suporte a tomada de decisão?

1.3.2 Objetivo Geral

O objetivo principal dessa dissertação é definir uma metodologia para determinar qual é uma abordagem mais eficiente e precisa para a análise de riscos em projetos de software. Algumas técnicas avaliadas são Simulação de Monte Carlo (SMC), Análise PERT, Modelo de Regressão Linear Múltipls (MRLM), Modelo de Regressão em Árvore (MRA), as alternativas de Redes Neurais Artificiais (RNA's) - especificamente, Perceptron de Múltiplas Camadas (MLP), Máquina de Vetor de Suporte(SVM), Redes de Função de Base Radial (RBF) - ou um Sistema *Neuro-Fuzzy* para diminuir o erro na estimativa de impacto dos riscos e a variância das estimativas.

1.3.3 Objetivos Específicos

- Desenvolver uma metodologia para previsão do impacto de riscos através da adoção de Redes Neurais Artificiais para gerenciar os riscos num projeto de software;
- Determinar uma linha de base para a estimativa do impacto de riscos, a partir da base de dados selecionada;
- Avaliar abordagens tradicionais de estimativa de impacto de riscos no que se refere ao erro de previsão;
- Avaliar diversas Redes Neurais Artificias para obter uma melhor configuração para a base de dados selecionada;
- Determinar um limiar de erro satisfatório para a estimativa de impacto de riscos;
- Apresentar resultados compreensíveis para gerentes e analistas de projetos e riscos.

O primeiro objetivo consiste em determinar uma metodologia para a previsão do impacto de riscos com o propósito de alcançar a maior eficiência possível, através da minimização do erro de previsão. Já o terceiro, envolve estudar Simulação de Monte Carlo (SMC) e Análise PERT para compará-las com Modelos de Regressão em Árvore (MRA) e Modelo de Regressão Linear Múltipla (MRLM), que servirão como linha de base (objetivo 2), com o propósito de comprovar se é uma boa prática mesmo adotar SMC e Análise PERT. O terceiro objetivo específico é alcançado após experimentar diversas Redes Neurais Artificiais como *Multilayer Perceptron* (MLP), *Support Vector Machine* (SVM) e *Radial Basis Function* (RBF). Além disso, um Sistema *Neuro-fuzzy* (NFS) também foi incluído nesse estudo. Por fim, foi necessário realizar uma pesquisa com profissionais e acadêmicos para identificar um limiar aceitável de erro na estimativa do impacto de um risco.

Em resumo, a metodologia adotada neste estudo é realizar uma série de experimentos para avaliar os erros de previsão dos impactos dos riscos oriundos da base de dados do PERIL [3], um *framework* para identificar riscos no gerenciamento de projetos de software. As técnicas selecionadas estimarão o resultado de impactos de risco. A Raiz do Erro Médio Quadrático (REMQ) será calculada trinta vezes para cada abordagem, e, então, um teste de hipótese pode ser necessário para afirmar qual deles é um método mais preciso que se ajusta as particularidades dessa base de dados. Mais detalhes sobre essa metodologia é apresentado no Capítulo 4.

É concluído que uma variação da MLP chamada de "MLPRegressor", é uma abordagem vencedora para estimar o impacto de riscos. Além disso, observou-se que todas as alternativas de Redes Neurais Artificiais são melhores que os Modelos de Regressão Linear e Simulação de Monte Carlo. Portanto, não foi descoberto qualquer motivo para atribuir SMC e PERT como métodos recomendados para a análise de riscos, de acordo com os experimentos conduzidos.

O restante da dissertação está organizado nos seguintes capítulos: Capítulo 2 aborda gerenciamento de risco de projetos, conceitos de análise de risco qualitativa e quantitativa, Simulação de Monte Carlo, análise PERT, Modelos de Regressão Linear e conceitos de Redes Neurais Artificiais e suas características. O Capítulo 3 descreve o banco de dados do PERIL, métodos de pré processamento de dados para preparar os dados para esse estudo e define as configurações dos algoritmos. O Capítulo 4 descreve cada experimento. O Capítulo 5 apresenta o resultado dos experimentos. Finalmente, o Capítulo 6 apresenta as conclusões e as sugestões de trabalhos futuros.

O trabalho descrito nessa dissertação teve alguns dos seus resultados publicados no artigo:

- 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

Referencial Teórico

2.1 Trabalhos Relacionados

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 (RNAs), Análise de Discriminantes (AD), Árvore de Decisão (ADE), Algoritmos Genéticos (AG), Otimização por Enxame de Partículas (PSO), Teoria dos Conjuntos *Fuzzy* (TCF), Sistema *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 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 Gerenciamento de Risco em Projetos

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) [5].

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 [5], 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 [4] [29].

Boehm [14], 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 [3], o gerenciamento de risco de projetos deve focar em riscos no sentido restrito.

Já 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 [3].

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 [3].

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 [3].

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. Quando realizada de forma 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, ela revela fraquezas num plano de projeto e desencadeia mudanças, novas atividades e realocação de recursos que melhoram o resultado do projeto [3].

2.2.1 Análise Qualitativa de Riscos

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 Análise Quantitativa de Riscos

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 [14] 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 [5]:

- 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 [5].

Alguns trabalhos propõem novas ferramentas de análise quantitativa de riscos. Entre eles, Virine [11] apresenta o método 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 SMC.

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 [3].

2.3 Técnicas Convencionais para Análise de Risco

2.3.1 Simulação de Monte Carlo

A Simulação de Monte Carlo (SMC) 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 [5].

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 SMC 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 Análise PERT

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

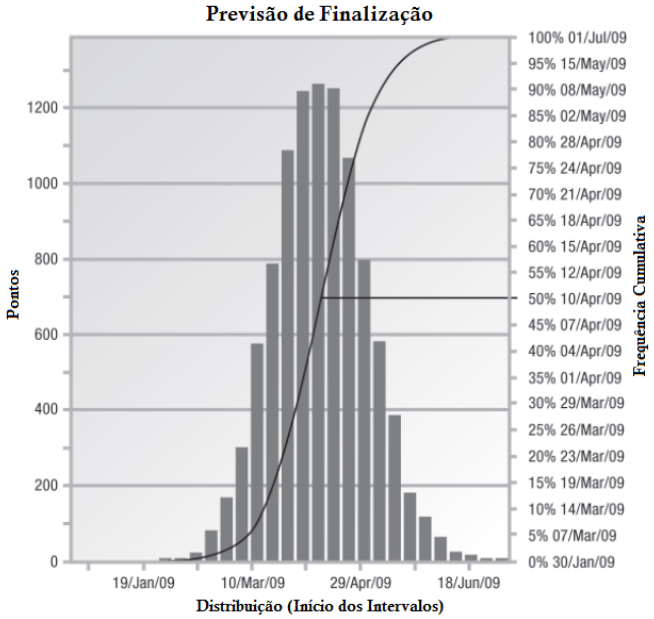

Figura 2.1: Exemplo de Resultado da Simulação Monte Carlo

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:

$$\sigma^2_{100}(\tau_e) = [(b-a)/6]^2 \tag{2.2}$$

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:

$$\sigma^2_{90}(\tau_e) = [(b-a)/3.2]^2 \tag{2.3}$$

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 Técnicas Estatísticas e de Computação Inteligente para Análise de Risco

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 consequência 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 Modelo de Regressão Linear Múltipla

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 e 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 Modelo de Regressão em Árvore

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

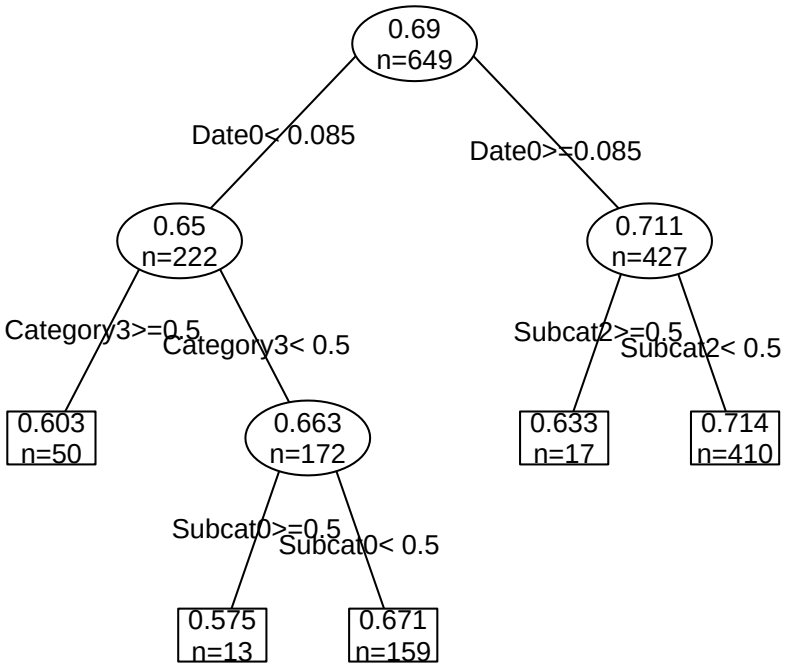

Figura 2.2: Regression Tree Model for PERIL

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, 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 Redes Neurais Artificiais

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 entrada 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.

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 Perceptron de Múltiplas Camadas

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 2.3 ilustra a forma gráfica da MLP, apresentando as entradas, saídas e as camadas de entrada, intermediária e de saída.

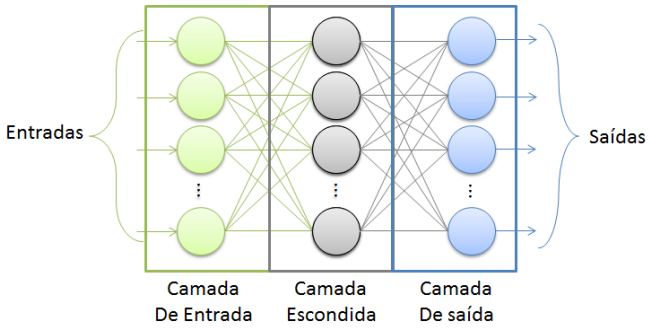

Figura 2.3: Representação gráfica da MLP com três camadas

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].

Algoritmo *Backpropagation*

A primeira etapa do *Backpropagation* consiste na inicialização dos pesos com valores aleatórios. A partir daí, para cada exemplo da base de treinamento são realizados os seguintes passos: 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 as entradas 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. O subconjunto para treinamento é apresentada à rede e serve para realizar o ajuste dos pesos pelo algoritmo de treinamento. O subconjunto de validação serve para realizar uma avaliação do treinamento da rede. Quando a REMQ para o subconjunto 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 Máquina Vetor de Suporte

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 2.4. Na Figura 2.4, 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.

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

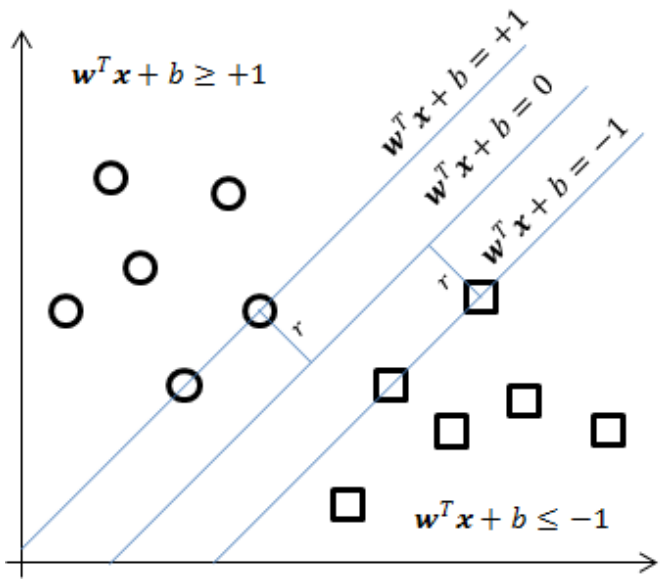

Figura 2.4: Vetores de Suporte e Hiperplano de Separação ótimo

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 [54] 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 amostras 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 Rede com Função de Base Radial

As Redes com 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 $\phi_i(x)$ 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 de fina Spline

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

onde, x_j são amostras 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 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 amostras 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 Regras de Aprendizado

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.

Aprendizado por Gradiente Descendente

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 2.5 [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.

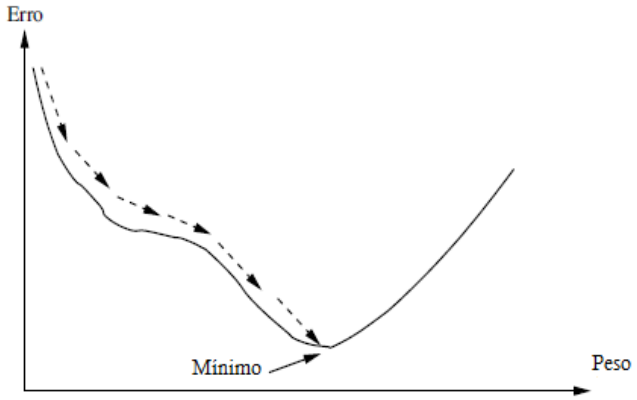

Figura 2.5: Vetores de Suporte e Hiperplano de Separação ótimo

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].

Aprendizado por Widrow-Hoff

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].

Aprendizado por Correção de Erro

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].

Aprendizado baseado em Memória

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_{teste} (não visto antes), o algoritmo responde buscando e analisando os dados de treinamento em uma ”vizinhança local”de x_{teste}.

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_{teste};
2. A regra de aprendizagem aplicada aos exemplos de treinamento na vizinhança local de x_{teste}.

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_{teste}. 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_{teste} se

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

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

Aprendizado por 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.

Aprendizado por Quasi-Newton

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].

Aprendizado por Broyden-Fletcher-Goldfarb-Shanno *Backpropagation*

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].

Aprendizado por Backpropagation e Secante de Um-passo

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].

Aprendizado por Levenberg-Marquardt

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

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

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.

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

2.6 Sistema *Neuro-Fuzzy*

2.6.1 Sistema de Inferência *Fuzzy* baseado em Rede Adaptativa

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].

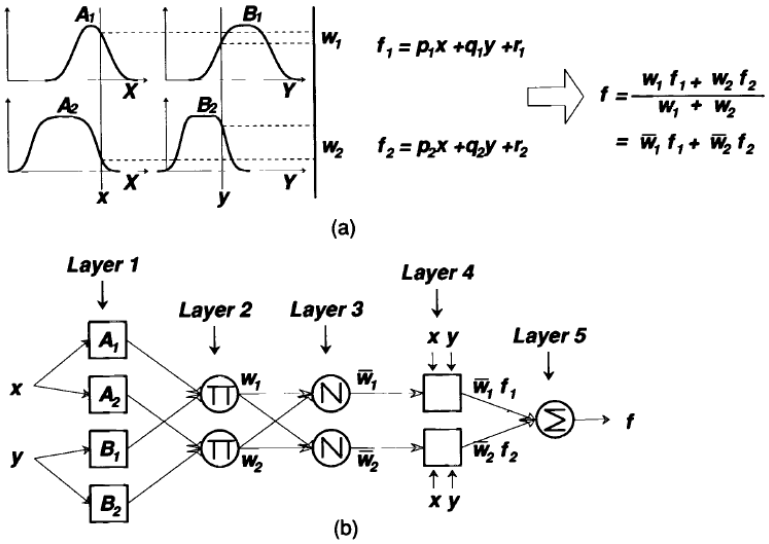

Figura 2.6: (a) Um modelo *fuzzy* de Sugeno de primeira ordem com duas entradas e duas regras; (b) arquitetura equivalente ANFIS

Arquitetura do ANFIS

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 2.6(a) ilustra o mecanismo de raciocínio para esse modelo de Sugeno; a arquitetura ANFIS equivalente correspondente é como mostrada na Figura 2.6(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 :

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

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* $A(A_1, A_2, B_1 orB_2)$ 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 :

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

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 :

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

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

Camada 4: Cada nó *i* nesta camada é um nó adaptativo com uma função de nó

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

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, é proposta 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 o desenvolvimento dessa metodologia, primeiramente necessita-se de uma base de dados extensa e real de riscos em projetos de *software*. No entanto, há uma dificuldade em se encontrar publicamente bases de dados representativas e confiáveis. Uma base de dados de risco chamada PERIL [3] 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 de quais são os principais modos de falha em projetos e um padrão para explorar respostas aos riscos.

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

Em quarto lugar, deve-se implementar modelos de previsão baseados em RNAs para que se possa selecionar o melhor modelo dentre estes: Perceptron de Múltiplas Camadas, Máquinas de Vetor de Suporte, Redes de Função de Base Radial e 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 a realização da tarefa de otimização dos parâmetros das RNAs. O espaço de busca do problema de otimização desses parâmetros é multi-dimensional, complexo e apresenta diversos mínimos locais.

Na Figura 3.1, observa-se um esquema ilustrando um algoritmo de otimização que executa os quatro modelos em suas diversas configurações paramétricas para possibilitar a escolha do modelo mais eficiente para a base de dados adotada.

Em quinto lugar, um teste de validação dos resultados é realizado para verificar a eficiência dos modelos baseados em redes neurais artificiais em relação aos modelos de estado da arte em análise de riscos, tais como os modelos baseados em Simulação de Monte Carlo, Análise PERT, Modelo de Regressão Linear Múltipla, Modelo de Re-

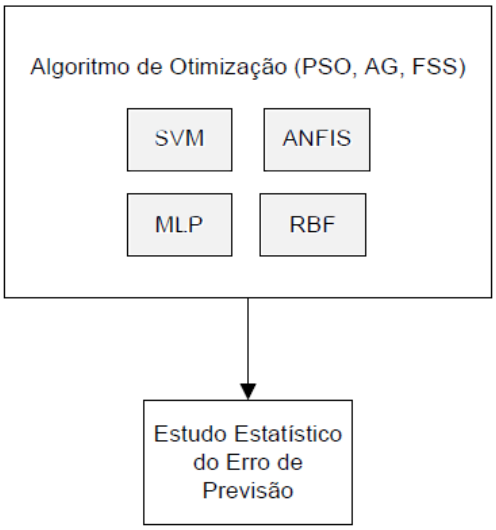

Figura 3.1: Esquema de Seleção da Melhor Rede Neural Artificial

gressão em Árvore. Como não existe estudos anteriores para a estimativa do impacto para essa base de dados, os dois últimos modelos de regressão linear foram considerados como linha de base, por serem modelos mais simples. Os dois últimos modelos foram implementados e também são avaliados quanto ao erro na previsão.

A seleção do melhor modelo é feita em termos da precisão na estimativa dos impactos dos riscos. 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) seja 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 tuplas 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 e 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). Esse teste estatístico é utilizado porque não há evidências que as amostras sejam oriundas de uma população normalmente distribuída, como também não há relação de ordem nos valores pertencentes às amostras.

A validação cruzada [64] é utilizada para evitar a ocorrência de *overfitting* ou *underfitting* durante o treinamento das RNA's. Nesse caso, um treinamento com parada prematura é utilizado para identificar o início do *overfitting*, já que esse método tem provado ser capaz de melhorar a capacidade de generalização da RNA em comparação com o treinamento exaustivo [51] [48] [65]. Portanto, o método de validação cruzada

é utilizado para cada abordagem, com exceção da Simulação de Monte Carlo e Análise PERT, por promover uma maior capacidade de generalização. Quando se adota o uso da validação cruzada, é necessário o particionamento da 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 RNA's, momento em que o aprendizado ocorre. O conjunto de validação cruzada é processado no mesmo tempo junto 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 estabelecer 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-se com a seleção da base de dados utilizada 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 executadas paralelamente.

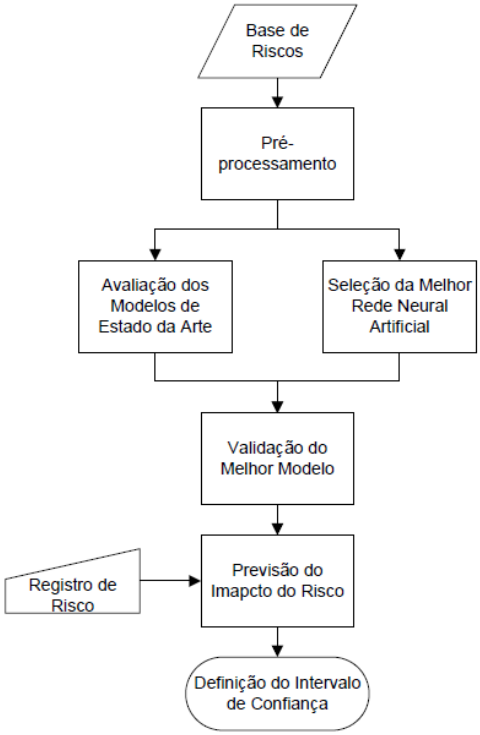

Figura 3.2: Fluxograma do estudo realizado

3.1 Base de dados PERIL

Um melhor gerenciamento de riscos se 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) [3]. A base de dados PERIL provê informações e experiências de outras gestões de risco de projetos.

Por mais de uma década, durante *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 enfrentar e é valiosa na ajuda para a identificação do que pelo menos poderiam ter sido riscos invisíveis ou *black swans*, isto é, riscos com grande impacto, difíceis de prever e com ocorrência rara [3].

Segundo Kendrick, 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 riscos ”conhecidos”e diminuir o número de riscos ”desconhecidos”[3].

Algumas características observadas na base de dados PERIL são:

- Os dados não estão relacionados, eles representam somente uma pequena fração de dezenas de milhares de projetos realizados por líderes de projetos, cujos riscos foram coletados;
- Apresentam viés, a informação não foi coletada aleatoriamente;
- Representam somente os riscos mais significativos;
- Foram coletadas no mundo todo, majoritariamente nas Américas;
- Não identificam oportunidades;
- Contém seiscentos 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 maior do 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*"que são riscos com grande impacto, difíceis de prever e de rara ocorrência [66].

Kendrick decidiu padronizar o impacto usando como métrica o tempo, medido em semanas de atraso no projeto. Essa estratégia 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 que está completamente fora de nosso controle. Afinal, o tempo sempre passa e quando não se é utilizado, não é possível voltar atrás [1].

Tabela 3.1: Número bruto de projetos no PERIL

	Américas	Ásia	Europa/Oriente Médio	Total
Projeto de TI/Soluções	256	57	18	331
Projeto de Des. de Produto	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: Impacto total de projetos pelas causas-raiz de categorias e subcategorias [1]

Subcat. Causas-raiz	Definição	Casos	Impacto Cumul.	Impacto Médio
Escopo: Mudanças	Revisão no escopo durante o projeto	177	1460	8,2
Recurso: Pessoas	Problemas de relacionamento interno	123	706	5,7
Escopo: Defeito	Falha em alcançar requisitos de entrega	93	654	7,0
Cronograma: Atrasos	Atraso devido a fatores sob o controle do projeto	102	509	5,0
Cronograma: Estimativas	Durações inadequadas alocadas para atividades do projeto	49	370	7,6
Recurso: *Outsourcing*	Problemas de relacionamento externo	47	316	6,7
Cronograma: Dependências	Atraso no projeto devido a fatores externos	41	262	6,4
Escopo: Mudanças	Financiamento insuficiente	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 analisadas nesse estudo. No entanto, um dos grandes benefícios é que o autor apresenta alguns riscos como *black swans* [1], então utilizando a PERIL responde-se a primeira questão dessa pesquisa. 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 *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* define-o como um evento raro, difícil de prever e de grande impacto. Mas não deixa de ser aplicável a gestão de risco do projeto. Nos projetos, é comum que os líderes de projeto descartem os principais riscos do projeto, devido as suas probabilidades serem 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 inevitáveis [1].

Na PERIL, há cento e vinte e sete 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 [1]. 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 Pré-processamento dos Dados

PERIL contém valores nominais e numéricos. As variáveis nominais foram expressas através de variáveis binárias. Nesse estudo, utilizam-se doze variáveis binárias para representar as variáveis nominais. 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]. A seleção das variáveis mais significativas para o estudo foi realizada 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 gráficos 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 curva sobre o histograma.

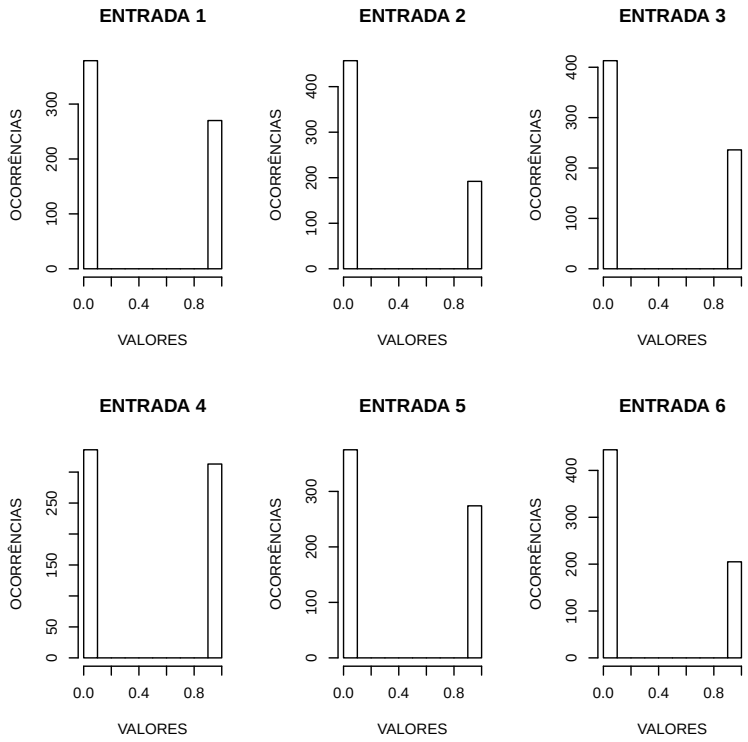

Figura 3.3: Primeiras seis variáveis de entrada

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 o MRLM e o MRA quanto para o MLP, SVM, RBF e o ANFIS; sendo que para os últimos o método da parada prematura também foi utilizada no treinamento [67].

3.3 Modelos de Estado da Arte

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

3.3.1 Simulação de Monte Carlo

A Simulação de Monte Carlo utilizou a base de dados completa 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 dessa configuração, o desempenho da estimativa do impacto dos

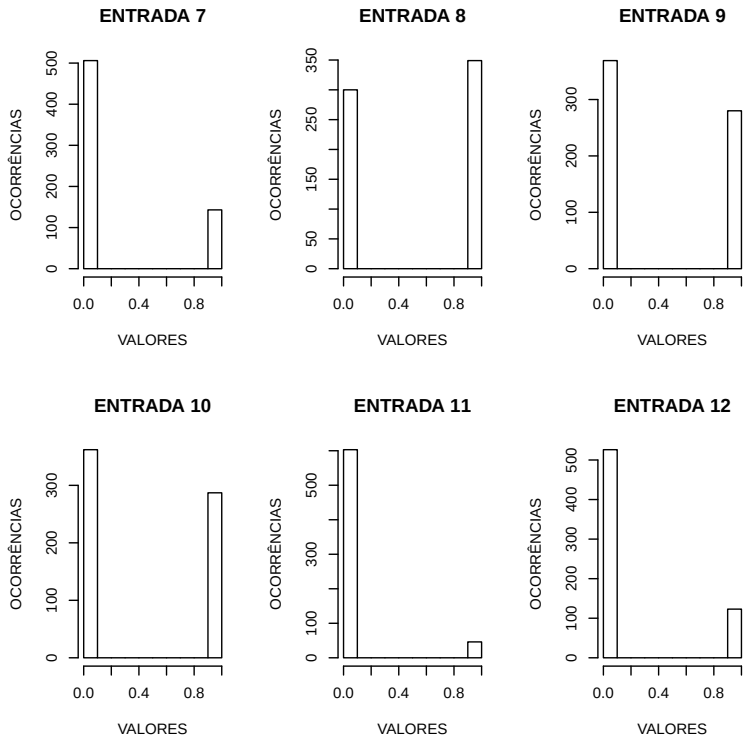

Figura 3.4: Últimas seis variáveis de entrada

riscos foi melhorado. Já que, a SMC é um método estatístico, a previsão de valores a partir de uma amostra pouco representativa como ocorre quando não se filtram as somente as saídas possíveis geram previsões mais errôneas.

3.3.2 Análise PERT

A Análise PERT também utilizou a base de dados completa e somente as saídas possíveis para uma determinada entrada foi utilizada no cálculo da nova saída, tal qual a SMC. A partir dessa configuração o resultado desse modelo foi melhorado, em comparação com o cenário em que a base de dados não foi filtrada.

3.4 Modelos de Regressão Linear

Nesta seção, são descritas 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 realização do treinamento, validação cruzada, teste e avaliação da Raiz do Erro Médio Quadrático. Os modelos MRLM 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.

Um estudo com esses modelos de regressão linear são necessários porque não há

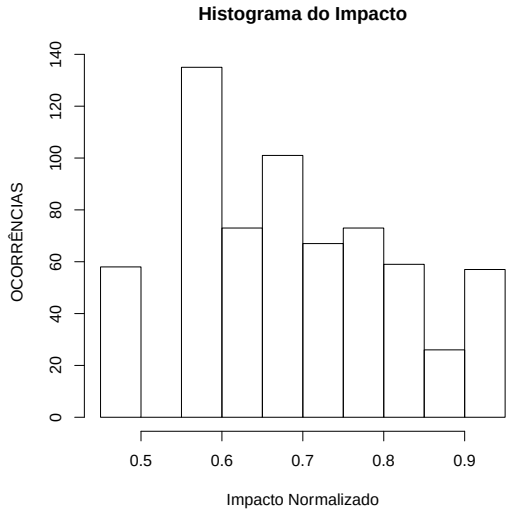

Figura 3.5: Histograma do impacto e forma da função de distribuição de ajuste

um estudo base para a estimativa de impactos de risco utilizando a PERIL. Portanto, durante os experimentos, a avaliação do melhor modelo de regressão linear tem como objetivo definir um limite superior de valores da REMQ. Isso significa que os modelos que obtiverem erros acima desse limite superior são insatisfatórios.

3.4.1 Modelo de Regressão Linear Múltipla

O MRLM é um modelo de regressão linear mais simples para a estimativa do impactos dos riscos apresentados na PERIL. 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 MRA constrói uma árvore de classificação das variáveis de entrada, podendo não ser necessário utilizar todas as variáveis de entrada para a construção do modelo. Ele tenta obter erros de previsão menores que o MRLM através da seleção das variáveis de entrada que têm maior correlação linear com a saída. 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 [68], como definido no início desse capítulo.

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, são descritas 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 tais como a quantidade de camadas escondidas, o número de neurônios escondidos em cada camada escondida, a taxa de aprendizado, o momento, o número máximo de ciclos de treinamento e a regra de aprendizado. Um modelo MLP utilizado nesse estudo é apresentado na Figura 3.6. Nesse modelo tem-se dez neurônios escondidos na 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 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, o momento e a quantidade de neurônios na camada escondida variam de acordo com os valores apresentados na Tabela 3.4.

Tabela 3.4: Intervalos de parâmetros para a MLP.

Parâmetros	Valor Mínimo	Valor Máximo
Momento	0.1	0.9
Taxa de Aprendizado	0.1	0.9
Neurônios Cam. Escondida	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*.

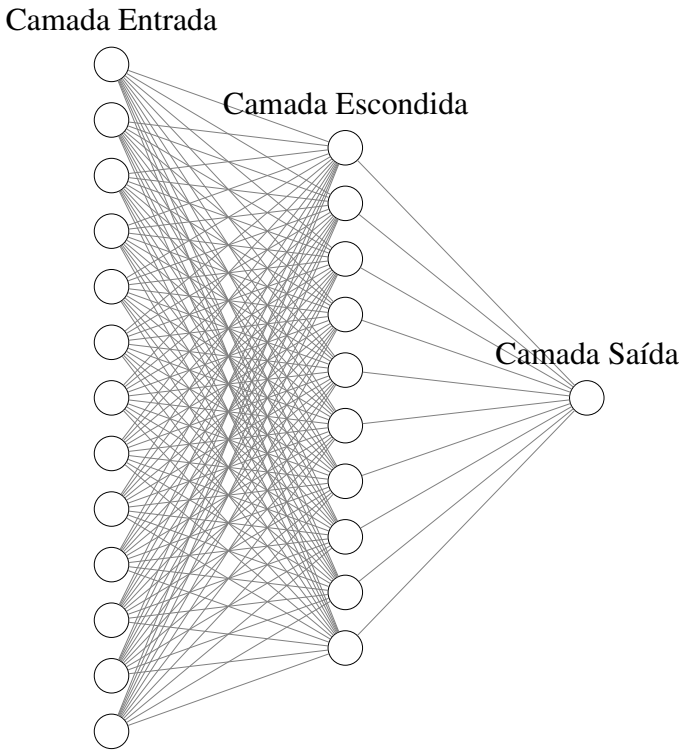

Figura 3.6: Um modelo MLP utilizado no estudo.

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 [69]. O pseudo-código para esse algoritmo é apresentado no Algoritmo 1.

Algoritmo 1: Algoritmo do SVM

```
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 1, 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 é calculado o REMQ a partir da saída real e da saída calculada. Esse algoritmo é utilizado como a função de otimização para o algoritmo de otimização por enxame.

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 2.

Algoritmo 2: Algoritmo do RBF

```
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 2, 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 das saídas real e calculada. Esse algoritmo é utilizado como a função de otimização para o algoritmo de otimização por enxame.

3.6.4 ANFIS

O ANFIS é um sistema *neuro-fuzzy* desenvolvido por Sugeno [62]. Ele utiliza 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 pseudo-código para esse algoritmo é apresentado no Algoritmo 3.

Algoritmo 3: Algoritmo do ANFIS

```
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, 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 das saídas real e calculada. Esse algoritmo é utilizado como a função de otimização para o algoritmo de otimização por enxame.

Capítulo 4

Experimentos

Os experimentos realizados utilizaram os modelos descritos no Capítulo 3 e a base de dados PERIL. Eles foram estabelecidos, primeiramente, analisando os modelos de regressão que foram a linha de base para o estudo, e em seguida as técnicas comumente utilizadas na academia e na indústria foram abordadas. O modelo do estado da arte, o ANFIS, foi analisado e por fim, mas não menos importante, as RNAs apresentadas na Seção 3.6 foram analisadas. 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 atingir esse objetivo é o erro de previsão, REMQ. As questões a serem respondidas foram apresentadas no Capítulo 1.

Três foram as possíveis hipóteses para os resultados dos experimentos:

- H_0: não há diferença entre usar os modelos em Redes Neurais Artificiais e os modelos de Estado da Arte para as estimativas dos impactos de riscos;
- H_1: os modelos em Redes Neurais Artificiais são mais precisos do que os modelos de Estado da Arte para as estimativas dos impactos de riscos;
- H_2: os modelos em Redes Neurais Artificiais são menos precisos do que os modelos de Estado da Arte para as estimativas dos impactos de riscos.

Os objetos de controle foram os códigos-fonte desenvolvidos para o experimento. Critérios de aleatoriedade, agrupamento e balanceamento foram adotados para facilitar a análise estatística. O objeto experimental foi a base de dados de risco PERIL. Por fim, os resultados foram analisados estatisticamente, através de testes de hipótese que foram conduzido tão logo os resultados foram obtidos.

4.1 Pré-processamento

Antes de se iniciar os experimentos, foi necessário preparar a base de dados PERIL para ser utilizada pelos modelos. Primeiro, inicia-se selecionando os registros de riscos classificados como comuns e *black swam*, totalizando setecentos e setenta e dois registros

de riscos. Esses registros foram apresentados no formato de uma tabela extensa completamente classificados através de um critério definido por Kendrick em seu livro [3]. A tabela contém oito colunas, dentre as quais uma delas é a descrição do evento de risco ocorrido e outra o impacto do risco; as seis restantes representam as classes definidas por Kendrick que são ”*Parameter*”, ”*Category*”, ”*Sub category*”, ”*Region*”, ”*Project*”e ”*Date*”.

A primeira dificuldade enfrentada foi transformar os dados nominais em numéricos. Decidiu-se ”binarizar”os dados de modo que para cada classe, apenas um dígito seja ”1”. Por exemplo, utilizando esse critério para ”binarizar”quatro classes é necessário um número binário de quatro dígitos. Após esse passo, obteve-se uma tabela com doze colunas. É importante lembrar que a coluna de descrição foi desprezada.

A partir daí, a segunda dificuldade foi identificada que é escolher qual o método de normalização apropriada para que o histograma dos impactos normalizados pudesse se aproximar do histograma da função normal, ou seja, apresentasse a forma de sino. Os métodos analisados foram a normalização linear, normalização estatística, normalização log-normal e normalização gamma. Para auxiliar a escolha do melhor método, investigou-se qual a função de distribuição de probabilidade se ajusta ao histograma dos impactos da PERIL. Nesse caso, verificou-se que as função de distribuição de probabilidade log-normal e gamma se aproximaram mais após uma análise visual. A escolha da normalização gamma foi comprovada, logo em seguida, após a obtenção do histograma dos impactos dos riscos normalizados.

Em seguida, as melhores variáveis de entrada foram escolhidas. Um número excessivo de variáveis de entrada pode degradar o desempenho de um modelo de previsão, provocando interferências na estimativa, aumento na complexidade do modelo e longos intervalos de processamento para obtenção dos resultados. Para realizar essa atividade, foi escolhido o método ”Random Forest”apresentado e descrito no livro de Torgo [37]. Esse método ordena as variáveis de entrada que estão correlacionadas com a saída. Quatro configurações foram geradas e analisadas: sem a remoção de variáveis de entrada, removendo as cinco variáveis menos importantes, removendo as dez variáveis menos importantes e removendo as quinze variáveis menos importantes.

Após a análise, as quatro configurações da base de dados através da estimativa do impacto de riscos utilizando uma MLP com regra de aprendizado *backpropagation*. Observou-se, que houve um aumento nos erros de previsão (REMQ) à medida que mais variáveis de entrada foram removidas. Então, nenhuma variável de entrada foi removida da base de dados pré-processada.

Por fim, um método de divisão de amostras foi utilizado para prevenir a possibilidade da ocorrência do *underfitting* e do *overfitting*. A base de dados pré-processada foi dividida em três subconjuntos balanceados: subconjunto de treinamento, de validação cruzada e teste. Esses subconjuntos eram recriados a cada simulação.

Portanto, após a binarização, a normalização utilizando a função gamma, a seleção das melhores variáveis de entrada e a divisão das amostras num subconjunto de validação cruzada, a base de dados PERIL encontrou-se preparada para ser utilizada nos experimentos subsequentes.

4.2 Regressão Linear Múltipla e Modelo de Regressão em Árvore

O primeiro experimento definido para este trabalho consistiu no estabelecimento de uma linha de base para que fosse possível comparar o desempenho de outras abordagens com a base de dados selecionada. Os modelos MRLM e MRA foram analisados e os seus erros de previsão (REMQ) foram comparados sendo selecionado aquele que apresentou o menor valor do erro. Os resultados foram produzidos rapidamente, devido a baixa complexidade desses modelos de regressão linear.

Nenhum trabalho dentre os encontrados na literatura estabeleceram uma linha de base para os estudos posteriores. Além disso, não foi desenvolvido um *benckmarking* para a estimativa do impacto de riscos. Logo, se faz necessário o estabelecimento de uma linha de base no início desse estudo.

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.

4.3 Simulação de Monte Carlo e Análise PERT

O segundo experimento consistiu em analisar o desempenho das técnicas convencionais utilizadas na academia e na indústria, inclusive determinadas como boas práticas pelo PMBOK [5]. Como explicado anteriormente, essas abordagens foram configuradas para obterem o melhor desempenho possível.

Esses modelos produziram os resultados mais rapidamente, devido ao fato deles utilizarem cálculos estatísticos extraídos da base de dados. Para esses modelos decidiu-se não dividir a base de dados, logo todas as setecentos e setenta e duas amostras foram utilizadas. Além disso, as amostras que apresentaram as mesmas variáveis de entrada foram filtradas para que se pudesse obter os menores REMQ possíveis.

4.4 Perceptron de Múltiplas Camadas e suas variações

O terceiro experimento teve como objetivo analisar quais das variações da MLP obteve 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 foi utilizada no experimento subsequente.

Nos primeiros resultados, uma MLP simples utilizando o algoritmo *backpropagation* apresentou uma REMQ média aproximadamente duas vezes maior (0,10007) que a MLPRegressor obtida nos últimos resultados (0,05168). Após um estudo mais detalhado, a investigação das variações de MLPs que contém diferentes regras de aprendizado foi sugerida e, posteriormente, aprovada, já que os resultados obtidos foram duas vezes inferiores.

Esse experimento e o próximo foram os experimentos mais significativos desse trabalho. A importância da avaliação de diversas alternativas à MLP tradicional, baseada

no algoritmo *backpropagation* foi 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.

4.5 MLP, SVM, RBF e ANFIS

O quarto experimento teve como objetivo eleger 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 foram selecionadas e a REMQ foi calculada para cada técnica.

Paralelamente a análise das variações da MLP, outras RNAs foram analisadas nesse caso a SVM e a RBF. Por último, o modelo de previsão do estado da arte ANFIS proposto por Saxena [25] foi implementado e a REMQ para previsões foram geradas.

Conforme ilustrado na Figura 3.1 do Capítulo 3, um algoritmo de otimização foi utilizado para que os parâmetros dos modelos fossem definidos de modo a reduzir a REMQ para cada modelo. Para a MLP, os parâmetros otimizados foram a quantidade de neurônios escondidos nas duas camadas escondidas, o fator de penalidade e o parâmetro de tolerância dos valores delta. Para o SVM, os parâmetros otimizados foram a constante de complexidade, um parâmetro na função de perda, um parâmetro para o erro de arredondamento, um parâmetro de tolerância para o critério de parada e o expoente para as funções gaussianas. Já para a RBF, os parâmetros otimizados foram o número de funções de base gaussiana, um parâmetros de tolerância para os valores delta, o fator de penalidade dos pesos nas saídas e o tipo da escala de otimização.

4.6 Validação do Melhor Modelo

Por fim, um teste estatístico dos resultados da melhor abordagem com os oriundos do segundo experimento foi realizado para observar se o modelo baseado em Redes Neurais apresentava maior precisão do que os tradicionais. Tendo sido validada a hipótese nula, de que as redes neurais artificiais são mais precisas e que poderiam atender à real necessidade da indústria, conclui-se que através da metodologia proposta nesse trabalho que é possível a obtenção de uma RNA mais precisa para as estimativas dos impactos de riscos.

Capítulo 5

Resultados

5.1 Regressão Linear Múltipla e Modelo de Regressão em Árvore

O primeiro experimento definido para este trabalho foi o estabelecimento de uma linha de base para que fosse possível comparar o desempenho de outras abordagens com a base de dados selecionada.

Inicialmente, esse experimento consistiu na seleção entre MRLM e MRA 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) e MRA_3 (cv.rpart.v3).

Tabela 5.1: Estatísticas descritivas para erros normalizados dos modelos de regressão linear.

	MRLM	MRA_1	MRA_2	MRA_3
Média	**0.09912**	0.10238	0.10305	0.10361
Desvio Padrão	**0.00391**	0.00423	0.00441	0.00426
Mínimo	**0.08956**	0.09214	0.09321	0.09372
Máximo	**0.10746**	0.11231	0.11267	0.11359

Na Figura 5.1, os *boxplots* da REMQ normalizada, após as previsões para MRLM, MRA_1, MRA_2 e MRA_3, são apresentados. Os menores valores de média, desvio padrão, mínimo e máximo da REMQ foi obtido para o MRLM. A partir dessa informação, pode-se afirmar que o MRLM é um modelo eficiente e preciso. Portanto, é definido como modelo de linha de base.

Os *boxplots* são elementos gráficos bastante utilizados em estudos estatísticos. Eles representam algumas medidas estatísticas como a mediana, o intervalo inter-quartil, os *whiskers* superiores e inferiores e os *outliers*. De modo geral, quanto menor o retângulo, menor o intervalo inter-quartil e o desvio padrão; e quanto menor a mediana, menor a média.

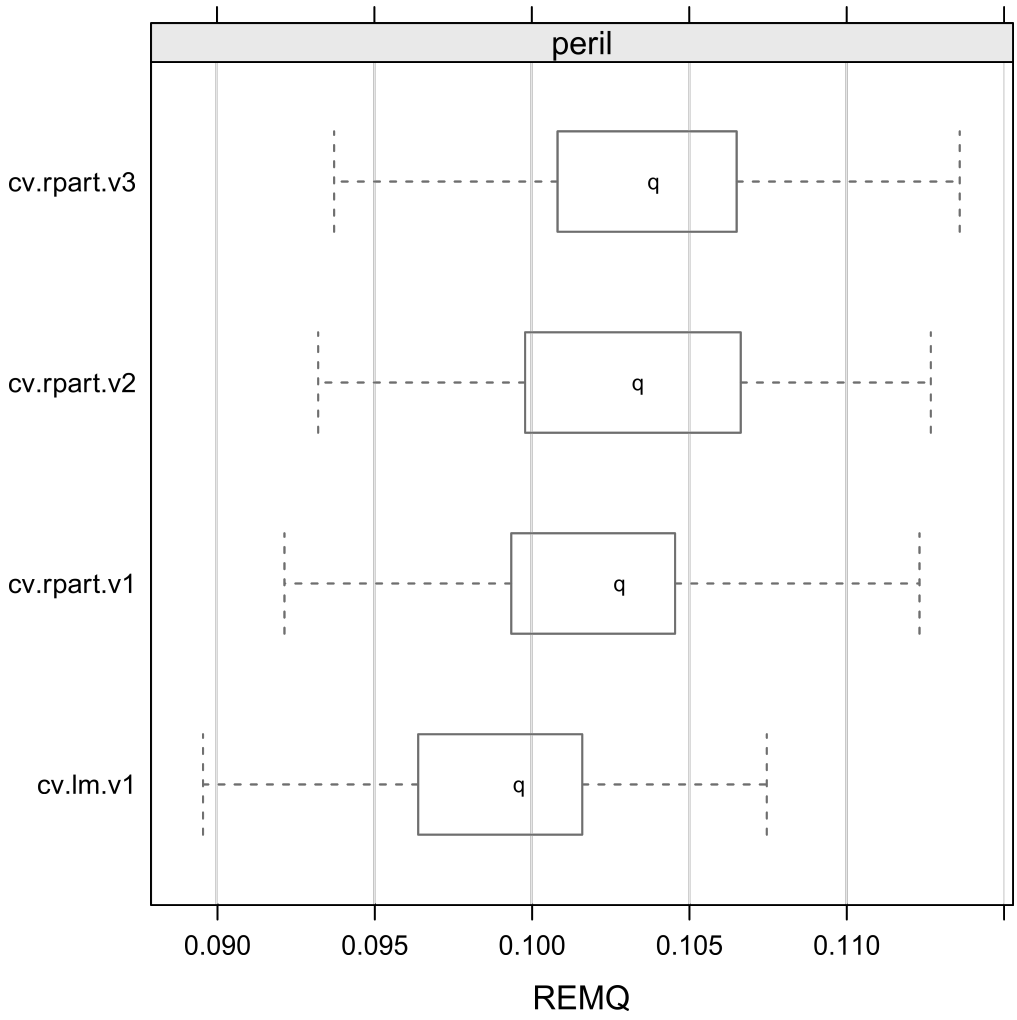

Figura 5.1: *Boxplots* para erros normalizados dos modelos de regressão linear.

5.2 Simulação de Monte Carlo e Análise PERT

O segundo experimento consiste em analisar o desempenho das técnicas convencionais, Simulação de Monte Carlo e Análise PERT, utilizadas na academia e na indústria, determinadas como boas práticas pelo PMBOK [5]. Como explicado anteriormente, essas abordagens foram configuradas para obterem o melhor desempenho possível.

A Tabela 5.2 mostra a estatística descritiva da REMQ normalizada 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 o último apresentou valores bem inferiores de média (0.07466), mínimo (0.04788) e máximo (0.09122). No entanto, o desvio padrão da Simulação de Monte Carlo é inferior (0.01250), provando ser um método mais preciso.

Na Figura 5.2, os *boxplots* da REMQ normalizada, após as previsões para SMC

Tabela 5.2: Estatísticas descritivas dos erros normalizados para modelos do estado da arte.

	SMC	PERT
Média	0.12640	**0.07466**
Desv. Padrão	**0.01250**	0.01438
Mínimo	0.10410	**0.04788**
Máximo	0.14950	**0.09122**

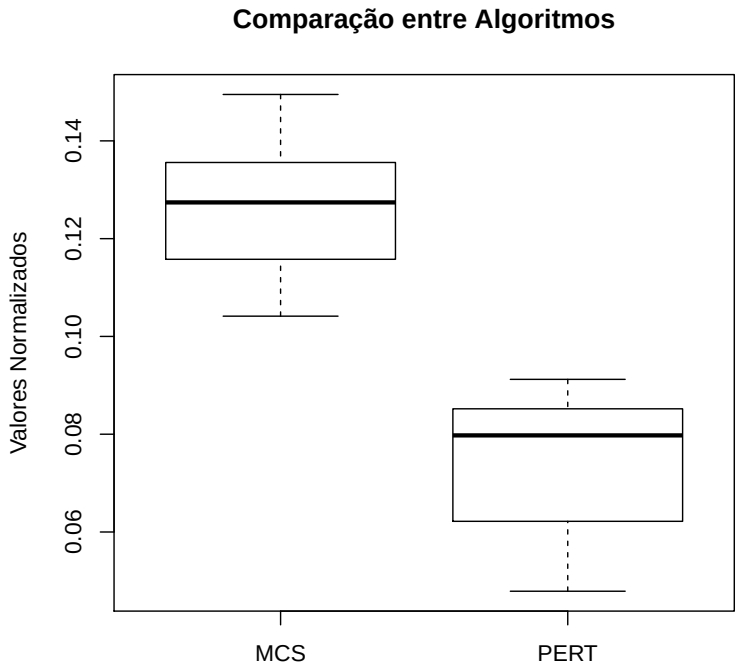

Figura 5.2: *Boxplots* para erros normalizados dos modelos de regressão linear.

e PERT, são apresentados. A partir dessa informação, pode-se afirmar que a Análise PERT é um modelo de estado da arte mais eficiente 40% na média, porém, menos preciso 40% quanto ao desvio padrão quanto ao REMQ comparado com o SMC.

A Análise PERT mostrou ser uma abordagem bastante interessante quando há poucas informações sobre riscos nas lições aprendidas no gerenciamento de projetos anteriores, por meio de uma base de dados de registro de riscos. Uma análise detalhada utilizando uma base de dados revela que essa técnica apresenta resultados aceitáveis pelos gerentes de projetos.

5.3 Perceptron de Múltiplas Camadas e suas variações

Nesse experimento, algumas MLPs foram avaliadas, sendo a principal diferença entre elas a regra de aprendizagem. Os algoritmos *Backpropagation*, Levenberg-Marquardt, Broyden-Fletcher-Goldfarb-Shanno *Backpropagation*, Gradiente Conjugado Escalonado, *Resilient-propagation*, *One-step Secant backpropagation* e Quasi-Newton foram algu-

mas das regras selecionadas.

Na Tabela 5.3, é exibida a estatística descritiva da REMQ normalizada 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(0.0516), mínimo(0.0427) e máximo(0.0603), observa-se que a alternativa ”Reg”é a mais eficiente, mesmo tendo o maior desvio padrão(0.0011) segundo esse experimento.

Tabela 5.3: Estatísticas descritivas para erros normalizados das MLPs.

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

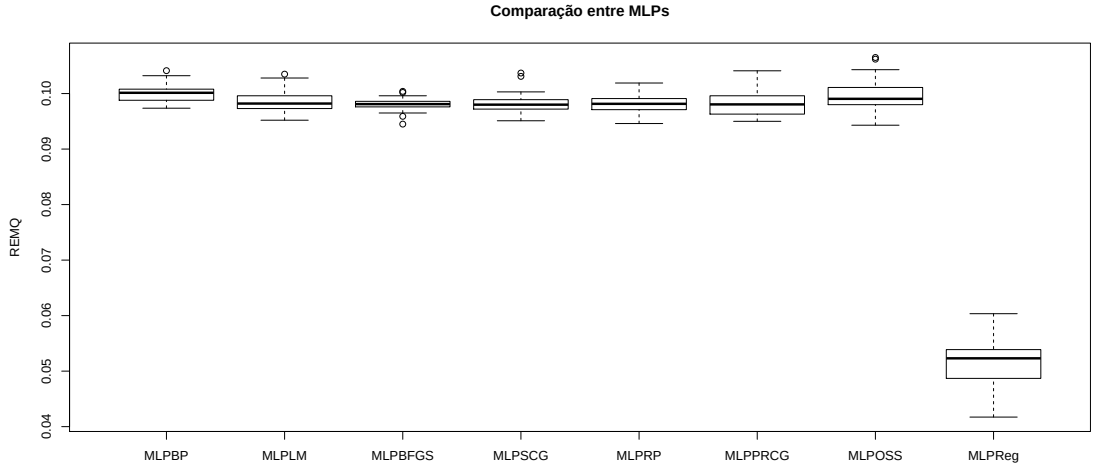

Figura 5.3: *Boxplots* para erros normalizados de várias MLPs.

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

5.4 MLP, SVM, RBF e ANFIS

Após determinar uma MLP 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 da REMQ normalizada 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 MLP ("MLPRegressor") mostram que o último algoritmo apresenta valores bem inferiores de média(0.0516), mínimo(0.0427) e máximo(0.0603). No entanto, o desvio padrão do ANFIS é menor(0.00003). Portanto, "MLPRegressor"provou ser um método mais eficiente 52% na média comparado ao ANFIS porém menos preciso, em comparação com as outras técnicas, para a estimativa do impacto de risco baseada na PERIL.

Tabela 5.4: Estatísticas descritivas para erros normalizados de RNAs.

	ANFIS	SMOReg	RBFReg	MLPReg
Média	0.1079	0.09430	0.09004	**0.0516**
Desv. Padrão	**0.00003**	0.00488	0.00432	0.0042
Mínimo	0.1078	0.08347	0.08024	**0.0427**
Máximo	0.1080	0.10284	0.09790	**0.0603**

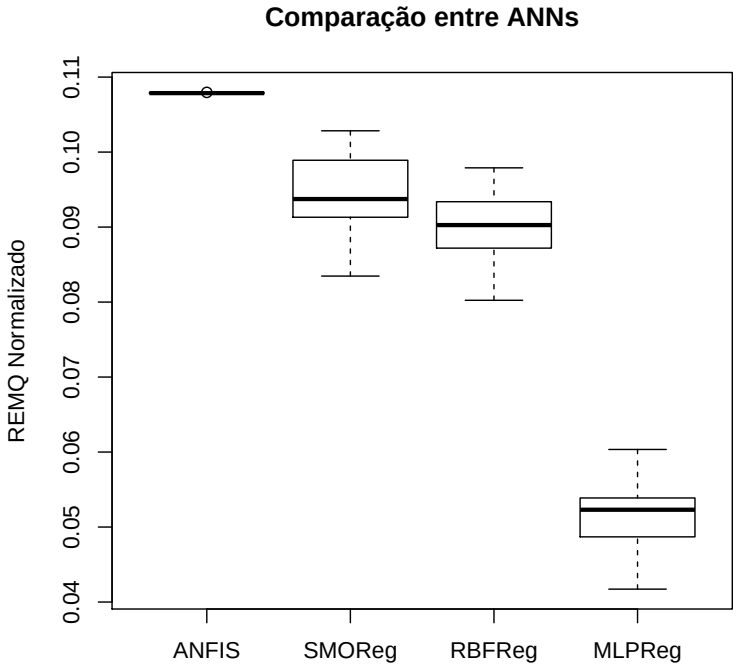

Figura 5.4: *Boxplots* para os erros normalizados das RNAs.

Na Figura 5.4, os *boxplots* da REMQ normalizada, após as estimativas de impacto

das RNAs, são apresentados. A partir dessas informações, pode-se afirmar que o "MLPRegressor"é a RNA mais eficiente, porém ainda um pouco imprecisa, de acordo com essa comparação.

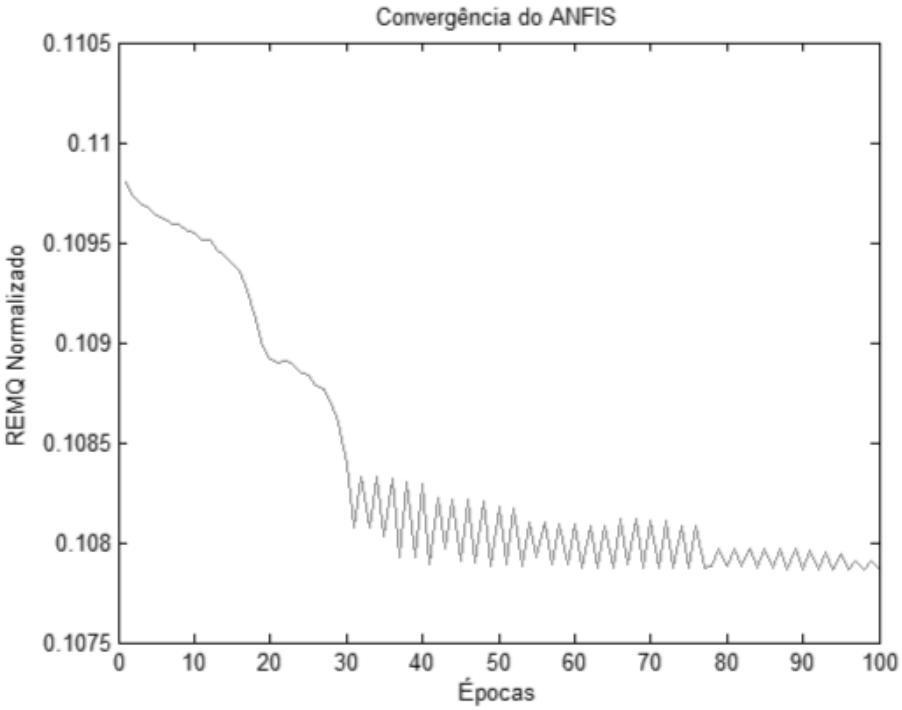

Figura 5.5: Gráfico de convergência do ANFIS.

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

Na Figura 5.6, os *boxplots* dos impactos esperados e calculados pelo algoritmo "SMORegressor"de cinquenta amostras são apresentados. O resultado ideal é que os *boxplots* das duas amostras fossem 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 tentou representar mas distorceu em todas as medidas os impactos esperados.

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"teve 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 cinquenta amostras são apresentados. O resultado ideal é que os *boxplots* das duas amostras fossem o mais semelhantes possível, respeitando a mediana, o intervalo inter-quartil e os valores de mínimo e máximo. A partir da informação apresentada,

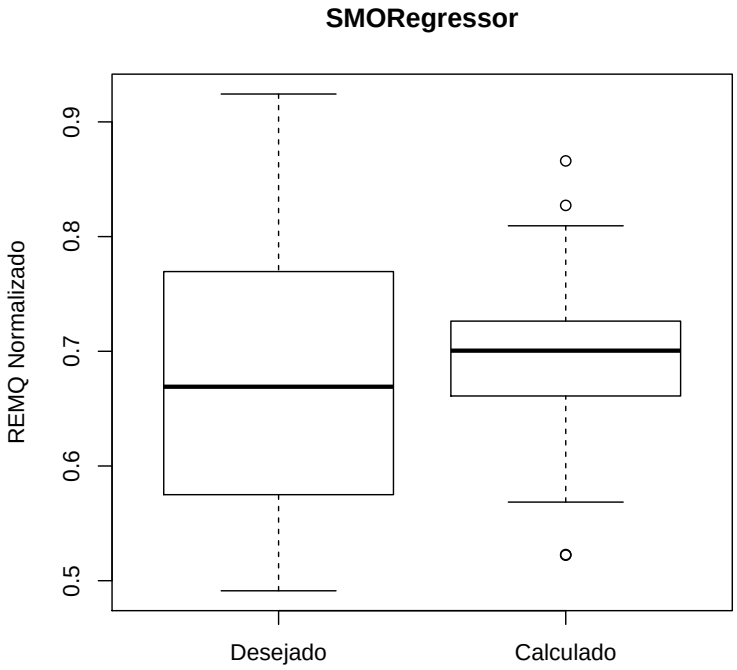

Figura 5.6: *Boxplots* para previsões de impacto esperados e calculados para SVM.

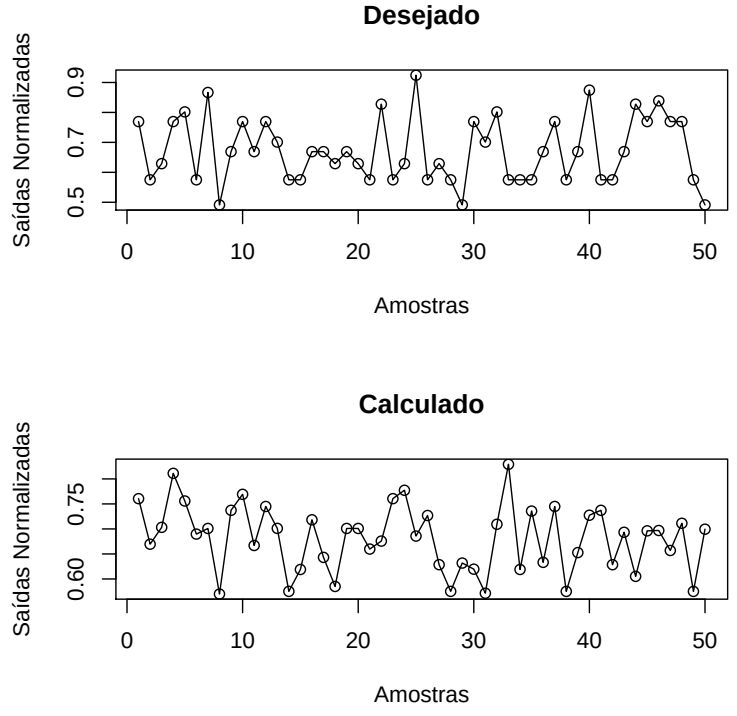

Figura 5.7: Amostras de previsões de impacto esperado e calculado para SVM.

pode-se concluir que o *boxplot* oriundo dos impactos calculados tentou representar mas

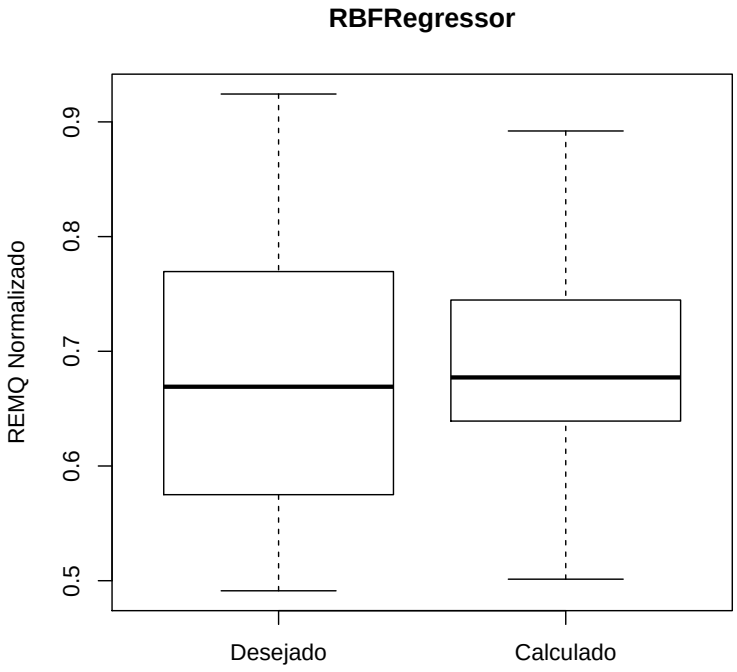

Figura 5.8: *Boxplots* para previsões de impacto esperados e calculados para RBF.

distorceu no quartil inferior e nos valores máximos os impactos esperados.

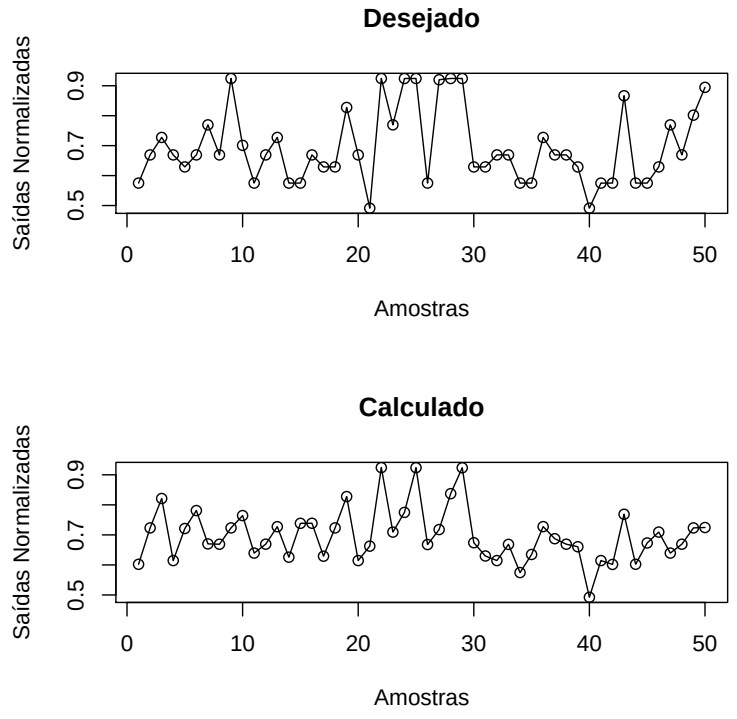

Figura 5.9: Amostras de previsões de impacto esperado e calculado para RBF.

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

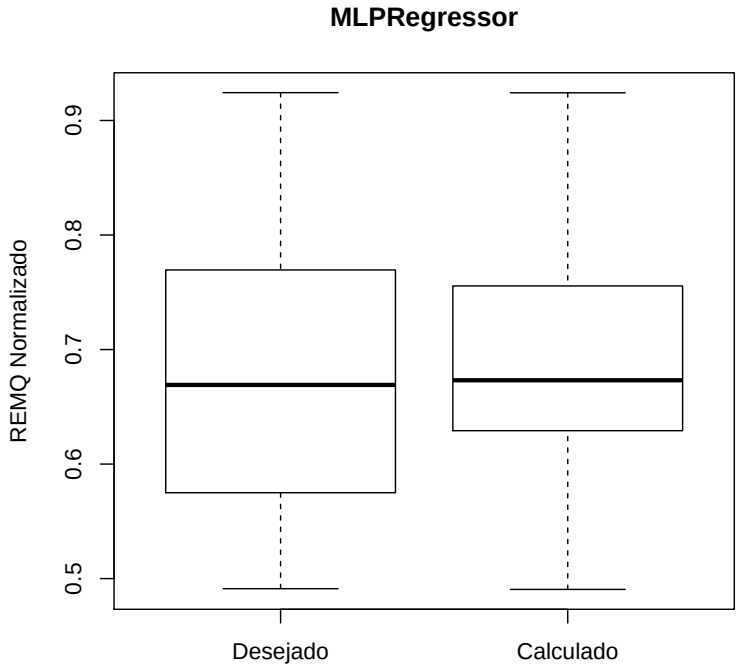

Figura 5.10: *Boxplots* para previsões de impacto esperados e calculados para MLPRegressor.

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 fossem 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 representou, porém com distorções principalmente no quartil inferior, os impactos esperados.

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

5.5 Validação do Melhor Modelo

A Tabela 5.5 mostra a estatística descritiva da REMQ normalizada 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"provou

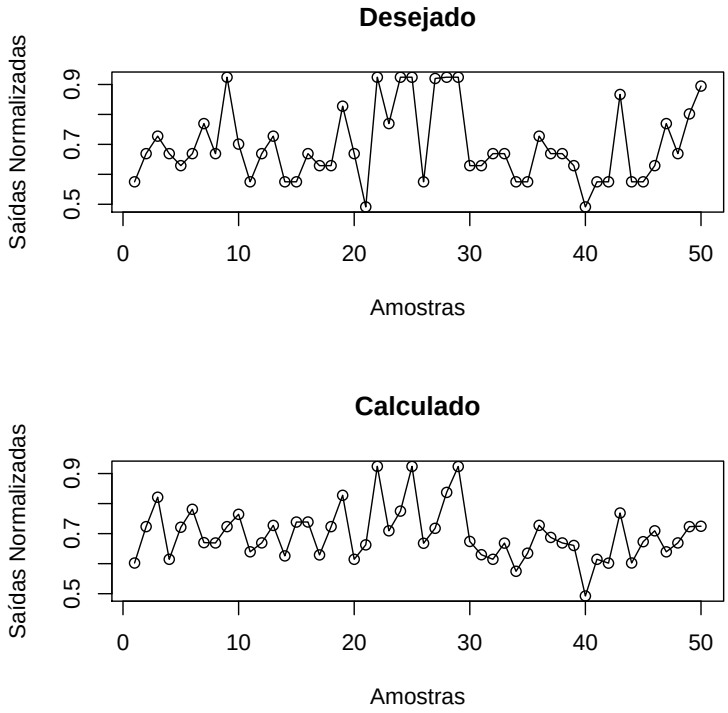

Figura 5.11: Amostras de previsões de impacto esperado e calculado para MLPRegressor.

ser um método mais eficiente e relativamente preciso, em comparação com as outras técnicas, para a estimativa do impacto de risco baseada na PERIL.

Tabela 5.5: Estatísticas descritivas para validação dos modelos selecionados.

	MLR	MCS	PERT	MLPReg
Méd	0.09912	0.12640	0.07466	**0.05168**
DvP	0.00794	0.01250	0.01438	**0.00427**
Mín	0.08956	0.10410	0.04788	**0.04172**
Máx	0.10746	0.14950	0.09122	**0.06035**

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 menores valores de média, desvio padrão, mínimo e máximo de erro. 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

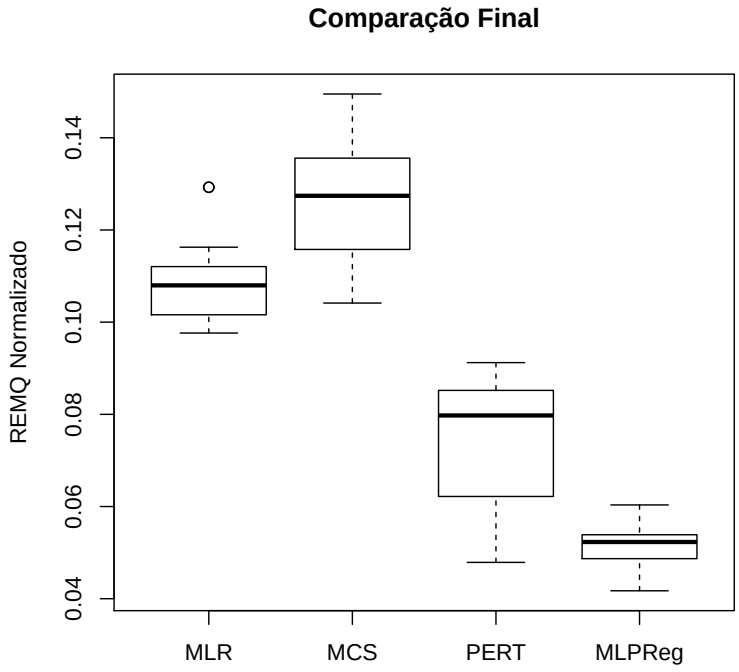

Figura 5.12: *Boxplots* dos erros normalizados para as técnicas de validação.

apresenta melhores resultados que à esquerda. Após analisar a tabela, pode-se afirmar que a MLPRegressor foi melhor que a Análise PERT que, por sua vez, foi melhor que o Modelo de Regressão Linear Múltipla.

Tabela 5.6: Testes de Hipótese para validação do melhor modelo.

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

5.6 Previsão do Impacto e Definição do Intervalo de Confiança

O último experimento consistiu 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 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, foi gerado um intervalo que, para uma determinada previsão, teve 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.

$$\sigma_{total}^2 = \sigma_v^2 + \sigma_w^2 \tag{5.1}$$

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, armazenou-se os erros para todas as entradas utilizadas no treinamento da rede, incluindo dados de treinamento e validação cruzada. Com esses dados devidamente armazenados e normalizados, criou-se um novo modelo, com novos pesos e novas ligações. Esse modelo teve 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 $\sigma_t otal$. Este foi utilizado para calcular o intervalo de confiança a partir da Equação 5.3.

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

onde, x foi o valor calculado pela rede e t foi 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: Tabela T de Student

$P(t_n \leq x)$				
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

Conclusões e Trabalhos Futuros

Este trabalho investigou o uso de redes neurais artificiais, como algoritmos SVM e MLP, para a estimativa do impacto do risco, através da proposta de uma metodologia para a análise de risco em projetos de *software*, a partir de dados históricos de registros de riscos.

Os resultados mostraram que as redes neurais artificiais apresentaram resultados promissores, principalmente porque um modelo MLP chamado "MLPRegressor"obteve os menores erros de previsão. Além disso, alguns resultados apresentaram informações importantes: a SMC apresentou resultados piores comparado com o modelo de linha de base, o MRLM; o modelo ANFIS apresentou resultados piores comparado com a "MLPRegressor", porém é mais preciso que o último modelo; é difícil melhorar ainda mais o desempenho da Análise PERT e da SMC já que todas as medidas foram tomadas para que com as informações disponíveis esses modelos pudessem apresentar os melhores resultados.

Além disso, uma pesquisa rápida realizada com duzentos profissionais, os quais dez responderam às perguntas feitas, informaram que, no geral, entre 0% e 5% é um intervalo de erros ideal de previsão de impacto de risco; 5% e 10% é um intervalo aceitável de erro na estimativa e 10% e 15% é um intervalo indesejado. Portanto, como o "MLPRegressor"apresentou alguns valores no primeiro intervalo, REMQ médio de 0.05168 ou 5,168%, é possível afirmar que esse modelo apresenta resultados satisfatórios.

Algumas dificuldades foram encontradas nesse estudo, principalmente quanto ao pré-processamento dos dados e a seleção das redes neurais e suas variações para se obter os resultados desejados. Porque não há qualquer trabalho científico utilizando a PERIL para se tomar como base.

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

[2] 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.

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

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

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

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

[7] 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.

[8] 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.

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

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

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

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

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

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

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

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

[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] Maurice Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999.

[69] 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.