**Referencial Teórico**

**Related Work**

Após uma revisão sistemática da literatura, foram identificadas inúmeras abordagens para a análise de risco: Modelo de Regressão Logística (MRL), Rede de Crenças Bayesiana (RCB), Redes Neurais Artificiais (RNA), Análise de Discriminantes (AD), Árvore de Decisão (ADE), Algoritmos Genéticos (AG), Otimização por Enxame de Partículas (PSO), Teoria dos Conjuntos Fuzzy (TCF), Sistemas Neuro-Fuzzy (SNF), Mapas Cognitivos Fuzzy Estendidos (E-FCM) \cite{mizuno2001prediction} \cite{huang2004neuro} \cite{hu2007software} \cite{attarzadeh2010novel} \cite{dzega2010classification} \cite{yu2011software} \cite{saxena2012software} \cite{dan2013improving}.

Hu et al. \cite{hu2007software} propuseram um método para análise de riscos de software e previsão do resultado de projetos de software. Algoritmos Genéticos foram utilizados como um método de otimização do desempenho de Redes Neurais por meio da seleção dos pesos, da estrutura da rede e da regra de aprendizado. A RNA padrão, nesse estudo, teve uma melhoria no desempenho após a inclusão de AG. Os resultados dos experimentos mostraram que após a introdução de AG para o processo de treinamento da RNA, o modelo de avaliação de risco de software modificado pôde ser sensivelmente melhorado alcançando uma maior precisão quando comparado com o modelo SVM. Zhang Dan \cite{dan2013improving} 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 \cite{attarzadeh2010novel} utilizaram a RNA para melhorar a precisão da estimativa do esforço comparado com o modelo tradicional COCOMO. Huang et al. \cite{huang2004neuro} apresentaram um \textit{framework} genérico para a estimativa de software baseado em SNF e melhoraram a estimativa de custo para o COCOMO'81.

Yu \cite{yu2011software} 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 \cite{saxena2012software} 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 \cite{lazzerini2011analyzing} sugeriram um \textit{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. \cite{mizuno2001prediction} 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 \cite{dzega2010classification} 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 \textit{boosting} e \textit{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 \cite{yu2011software} \cite{saxena2012software} \cite{lazzerini2011analyzing} \cite{dzega2010classification}.

**Project Risk Management**

Risco pode ser definido como um evento incerto ou condição que, se ocorrer, afeta pelo menos um dos objetivos do projeto. Ele pode ser considerado tanto como uma ameaça (impacto negativo) quanto uma oportunidade (impacto positivo) \cite{PMBOK2008}.

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 \cite{OSUNDAHUNSI2012}.

De acordo com o PMBOK \cite{PMBOK2008}, 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.

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

\item Identificar riscos: o processo de determinação dos riscos que possam afetar o projeto e a documentação de suas características;

\item 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;

\item Realizar a análise quantitativa dos riscos: o processo de análise numérica do efeito de riscos identificados previamente, em termos dos objetivos gerais do projeto;

\item Planejar respostas aos riscos: o processo de desenvolvimento de opções e ações para aumento das oportunidades e diminuição das ameaças aos objetivos do projeto;

\item 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 \textit{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 \cite{HIGUERAHAIMES1996} \cite{williams1999software}.

Boehm \cite{BOEHM1991}, Chapman \cite{chapman1996project}, Fairley \cite{fairley1994risk}, Bandyopadhyay et al. \cite{bandyopadhyay1999framework} 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 \cite{holzmann2011developing}.

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 \cite{kendrick2003identifying}, o gerenciamento de risco de projetos deve focar em riscos no sentido restrito.

O gerenciamento de risco de projetos, no sentido restrito, serve para melhorar as chances de cada projeto individual alcançar o sucesso. Na maioria dos outros campos, o gerenciamento de risco está preocupado principalmente com os valores médios de um grande número de eventos independentes. Para o gerenciamento do risco de projetos, no entanto, o que geralmente mais importa é a previsibilidade - gerenciar a variação esperada no resultado para o projeto específico \cite{kendrick2003identifying}.

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 \cite{kendrick2003identifying}.

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 \cite{kendrick2003identifying}.

O principal benefício do gerenciamento de riscos em projetos é tanto o desenvolvimento de uma base de credibilidade para cada projeto, mostrando que o mesmo é possível, quanto da demonstração da inviabilidade do projeto, mostrando que o mesmo deva ser evitado, abortado ou transformado. A análise de risco pode também revelar oportunidades para melhoria dos projetos que podem resultar em altos retornos no investimento. A análise de riscos adequada reduz tanto o custo total quanto a frustração causada por problemas evitáveis. A quantidade de retrabalho e atraso imprevisto de esforço no projeto é reduzida. Além disso, a análise de risco revela fraquezas num plano de projeto e desencadeia mudanças, novas atividades e realocação de recursos que melhoram o resultado do projeto \cite{kendrick2003identifying}.

**Qualitative Risk Analysis**

O processo de análise qualitativa avalia as características dos riscos de projetos identificados individualmente e prioriza-os baseado nas requisições estabelecidas para o projeto. Em outras palavras, a análise qualitativa avalia a probabilidade de cada evento ocorrer e o efeito individual de cada um deles nos objetivos do projeto. Como tal processo não aborda diretamente o risco global para os objetivos do projeto, que resultam do efeito combinado dos eventos individuais e as interações entre si, então isso pode ser obtido através do uso de técnicas de análise quantitativa \cite{PRACTICESTANDARD2009}.

**Quantitative Risk Analysis**

Análise é a conversão de dados de risco em informação para tomada de decisão. A análise fornece a base para o gerente de projetos trabalhar nos riscos certos e mais críticos. Boehm \cite{BOEHM1991} 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 \cite{PMBOK2008}:

\item 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);

\item 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;

\item 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;

\item 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 \cite{PMBOK2008}.

Alguns trabalhos propõem novas ferramentas de análise quantitativa de riscos. Entre eles, Virine \cite{VIRINE2009} apresenta a metodologia da Cadeia de Eventos. Nesse trabalho, as atividades de um projeto não são um procedimento uniforme e contínuo. Essas tarefas são afetadas por eventos externos, que transformam as atividades de um evento para outro. O momento em que os eventos externos ocorrem são probabilísticos e podem ser definidos utilizando uma distribuição estatística. Além disso, eventos podem causar outros eventos, criando, portanto, a Cadeia de Eventos. A análise dessas combinações é realizada através da Simulação de Monte Carlo.

A análise de risco demonstra a incerteza dos resultados do projeto e é útil para justificar reservas de cronograma e/ou recursos. É mais apropriado para definir uma janela de tempo (ou orçamento) em vez de um objetivo único para projetos arriscados. Por exemplo, o cronograma alvo para um projeto arriscado pode ser doze meses, mas o cronograma empenhado, refletindo problemas potenciais, pode ser fixado em quatorze meses. A conclusão no prazo (ou antes) desse intervalo define um projeto de sucesso; somente se o projeto demorar mais de quatorze meses será considerado um fracasso \cite{kendrick2003identifying}.

**Conventional Techniques for Risk Analysis**

**Monte Carlo Simulation**

A Simulação de Monte Carlo é uma técnica que computa ou itera o custo ou cronograma do projeto muitas vezes usando valores de entrada selecionados aleatoriamente a partir de distribuições de probabilidades de custos ou durações possíveis, para calcular uma distribuição dos custos totais ou datas de finalização do projeto \cite{PMBOK2008}.

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 \cite{kwak2007exploring}.

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 \cite{kwak2007exploring}.

A Simulação de Monte Carlo é uma abordagem detalhada de simulação através de computação intensiva para determinar a probabilidade de resultados possíveis de um objetivo do projeto; por exemplo, a data de conclusão ou o custo total. As entradas para o procedimento são obtidas aleatoriamente a partir de intervalos específicos com funções de distribuição de probabilidade para as durações das atividades do cronograma ou itens da linha de custo. Esses valores de entrada diferentes são usados para construir um histograma de possíveis resultados do projeto e sua probabilidade relativa, como também, a probabilidade cumulativa para calcular as reservas de contingência desejadas de tempo ou custo. Resultados adicionais incluem a importância relativa de cada entrada na determinação do custo geral do projeto e cronograma. Um exemplo de resultados de estimativa de riscos de cronograma e custo são apresentados na Figura \ref{fig:montecarlo} \cite{PRACTICESTANDARD2009}.

Na Figura \ref{fig:montecarlo}, 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.

\begin{figure}[h]

\centering

\includegraphics[width=.6\textwidth]{image/montecarlo.png}

\caption{Exemplo de Resultado da Simulação Monte Carlo}

\label{fig:montecarlo}

\end{figure}

**PERT Analysis**

PERT foi criada pela Marinha dos EUA em 1958 como uma ferramenta para programar o desenvolvimento de um sistema de armamento por completo. A técnica considera o projeto sendo uma rede acíclica de eventos e atividades. A duração de um projeto é determinada por um plano de fluxo do sistema no qual a duração de cada atividade tem um valor esperado e uma variância. O caminho crítico inclui uma sequência de 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 \cite{cottrell1999simplified}.

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 \cite{cottrell1999simplified}.

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 \ref{eq:mediaPERT}, é uma aproximação linear dessa forma:

\begin{equation}

\label{eq:mediaPERT}

\tau\_e = (\alpha + 4m + \beta) / 6

\end{equation}

onde $\tau\_e$ 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:

\begin{equation}

\label{eq:varianciaPERT}

\sigma^2\_100(\tau\_e) = [(b - a) / 6]^2

\end{equation}

onde $\sigma^2$ é a variância da duração da atividade. Uma alternativa, apresentada em \cite{cottrell1999simplified}, é definir $a$ e $b$ como os limiares de 5\% e 95\% do intervalo, respectivamente. Então, a variância é como se segue:

\begin{equation}

\label{eq:variancia2PERT}

\sigma^2\_90(\tau\_e) = [(b - a) / 3.2]^2

\end{equation}

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 \textit{forward} e \textit{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 \cite{cottrell1999simplified}.

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 \cite{cottrell1999simplified}.

**Statistical and Intelligent Computing Techniques for Risk Analysis**

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

**Multiple Linear Regression**

A Regressão Linear Múltipla está entre as técnicas de análise de dados estatísticos mais utilizadas. Esse modelo obtém uma função aditiva relacionando uma variável de saída a um conjunto de variáveis de entrada. Essa função aditiva é a soma de elementos da fórmula $\beta\_i \times X\_i$, onde $X\_i$ é uma matriz estimadora de variáveis e $\beta\_i$ é a matriz de pesos na regressão linear múltipla \cite{torgo2003data}

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 \ref{eq:mlr}:

\begin{equation}

\label{eq:mlr}

Y = \alpha + \beta X,

\end{equation}

onde a variância de $Y$ é assumida ser constante. Os coeficientes $\alpha$ e $\beta$ (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 \ref{eq:betamlr} and \ref{eq:alphamlr}:

\begin{equation}

\label{eq:betamlr}

\beta = \frac{\sum\limits\_{i=1}^{s} (x\_i - x) (y\_i - y) }{\sum\limits\_{i=1}^{s} (x\_i - x)^2},

\end{equation}

\begin{equation}

\label{eq:alphamlr}

\alpha = y - \beta x,

\end{equation}

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 $\alpha$ e $\beta$ 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 \cite{han2006data}.

**Regression Tree Model**

Uma árvore de regressão é uma hierarquia de testes lógicos em algumas das variáveis explanatórias do modelo. Logo, nem todas as variáveis aparecem na árvore. Uma árvore é lida do nó-raiz para os nós-folha. Cada nó da árvore tem dois galhos. Eles estão relacionados ao valor de um teste em cada uma das variáveis de previsão. O teste continua até um nó-folha ser alcançado. Nesses nós-folha tem-se os valores previstos pela árvore. Isso significa que se é necessário usar uma árvore para obter-se uma estimativa para uma amostra particular, é necessário seguir um galho do nó-raiz até um nó-folha, de acordo com os resultados do teste para a amostra. O valor médio da variável alvo encontrado no nó-folha alcançado é a previsão da árvore \cite{torgo2003data} \cite{breiman1984classification}.

\begin{figure}[h]

\centering

\includegraphics[width=\columnwidth]{image/regressiontreemodel.pdf}

\caption{Regression Tree Model for PERIL}

\label{fig:rtm}

\end{figure}

A Figura \ref{fig:rtm} 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 \textit{Date0}. Os próximos nós representam resultados possíveis levando-se em conta somente \textit{Date0}. O segundo nível da árvore mostra testes lógicos associados às variáveis \textit{Category3} e \textit{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 \textit{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 \textit{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 (\textit{overfitting}), e portanto irá ter um desempenho ruim quando exposta a uma nova amostra de dados para a qual previsões são necessárias. O problema de \textit{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 \textit{overfitting}, necessitando, portanto, de uma previsão baseada em estatística para diminuir o impacto desse efeito \cite{torgo2003data}.

**Artificial Neural Networks**

Uma Rede Neural Artificial (RNA) é um processador distribuído maciçamente paralelo constituído por unidades de processamento simples, que tem a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso \cite{linlee1996neuralfuzzy}. 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 \cite{valenca2005aplicando}.

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 \cite{MCCULLOCKPITTS1943}.

Em seguida ao trabalho de McCulloch e Pitts surgiu a regra de aprendizagem proposta por Donald Hebb \cite{hebb1949organization} 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 \cite{widrow1960adaptive} apresentaram uma regra de aprendizagem para uma extensão do Perceptron, desenvolvida previamente por Frank Rosenblatt \cite{rosenblatt1960perceptron}, chamada de ADALINE (Adaptive Linear Neuron). Esta regra baseada no método dos mínimos quadrados ficou conhecida como regra delta. Paul Werbos em 1974 \cite{Werbos74} desenvolveu o algoritmo de \textit{backpropagation}, sendo esse algortimo posteriormente popularizado através da publicação feita por Rumelhart e McClelland em 1985 \cite{rumelhart1985learning}.

O modelo do neurônio de McCulloch e Pitts procura representar o neurônio biológico utilizando uma regra de propagação e uma função de ativação. Considere $x\_1, x\_2, x\_3, ..., x\_n$, como sendo as variáveis de entrada $x\_j$, em que $j = 1,...,n$ do neurônio de saída $i$. A entrada líquida $net\_i$ é dada pela seguinte regra de propagação:

\begin{equation}

\label{eq:net}

net\_i = \sum\_{j=1}^{n} w\_{ij}x\_j - \theta

\end{equation}

onde, $w\_{ij}$ são os pesos sinápticos e $\theta$ é o limiar de ativação do neurônio.

\begin{figure}[h]

\centering

\includegraphics[width=.6\textwidth]{image/neuronio.png}

\caption{Representação gráfica da MLP com três camadas}

\label{fig:neuronio}

\end{figure}

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 \cite{valenca2005aplicando} \cite{engelbrecht2007computational}:

\begin{equation}

\label{eq:linear\_activation}

f(net\_i) = net\_i

\end{equation}

\begin{equation}

\label{eq:step\_activation}

f(net\_i) =

\begin{cases}

\gamma\_1 & \text{if } net\_i \geq \theta \\

\gamma\_2 & \text{if } net\_i < \theta

\end{cases}

\end{equation}

\begin{equation}

\label{eq:ramp\_activation}

f(net\_i) =

\begin{cases}

\gamma & \text{if } net\_i \geq \epsilon \\

net\_i & \text{if } -\epsilon < net\_i < \epsilon \\

-\gamma & \text{if } net\_i \leq -\epsilon

\end{cases}

\end{equation}

\begin{equation}

\label{eq:logisticsigmoid\_activation}

f(net\_i) = \frac{1}{1 + e^{-net\_i}}

\end{equation}

\begin{equation}

\label{eq:hiberbolictangent\_activation}

f(net\_i) = \frac{e^{net\_i} - e^{-net\_i}}{e^{net\_i} + e^{-net\_i}}

\end{equation}

\begin{equation}

\label{eq:gaussian\_activation}

f(net\_i) = e^{-(net\_i - \theta)^2/\sigma^2}

\end{equation}

**MultiLayer Perceptron**

A rede perceptron de múltiplas camadas é uma generalização da rede perceptron simples pela adição de pelo menos uma camada intermediária, conhecida como camada escondida. Em uma rede em camadas, os neurônios estão dispostos em cada uma delas. Na MLP, a primeira delas é a camada de entrada, na qual as variáveis de entrada são conectadas diretamente a um neurônio, exclusivamente. A próxima camada, denominada intermediária, liga completamente os neurônios da camada anterior aos neurônios da camada de saída. Por fim, a camada de saída representa a saída da RNA. Cada entrada em um neurônio tem um peso a ele associado a ser ajustado pelo algoritmo de treinamento. Um modelo comum de MLP contém um neurônio de \textit{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 \textit{forward}". O fluxo de dados na direção oposta é a "fase \textit{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 \cite{haykin1994neural}. O algoritmo \textit{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 \cite{rumelhart1985learning}.

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 \cite{HAYKIN2007}. A Figura \ref{fig:mlp} ilustra a forma gráfica da MLP, apresentando as entradas, saídas e as camadas de entrada, intermediária e de saída.

\begin{figure}[h]

\centering

\includegraphics[width=.6\textwidth]{image/mlp.png}

\caption{Representação gráfica da MLP com três camadas}

\label{fig:mlp}

\end{figure}

\begin{figure}[h]

\centering

\includegraphics[width=.6\textwidth]{image/mlp2.png}

\caption{Representação gráfica da MLP com três camadas}

\label{fig:mlp2}

\end{figure}

O funcionamento da MLP é dividido em duas fases: \textit{forward} e \textit{backward}. Na fase \textit{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 \textit{backward} o erro é retropropagado e os pesos são ajustados, utilizando o algoritmo de ajustes de pesos, inicialmente aleatórios, chamado \textit{Backpropagation} \cite{valenca2005aplicando}.

**Backpropagation Algorithm**

A primeira etapa do Backpropagation consiste na inicialização dos pesos com valores aleatórios. A partir daí, para cada exemplo da base de treinamento é realizado o seguinte processo: a propagação do sinal; a fase \textit{forward} e retropropagação do erro; e a fase \textit{backward}. Na fase \textit{forward} é possível calcular o erro entre a saída desejada para a rede e a saída apresentada pela MLP. Na fase \textit{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:

\begin{equation}

\label{eq:sensibiliy\_function}

\delta^{m-1}\_i = f^{m-1'}(net^{m-1}\_j)\sum\_{i=1}^{N\_{neurons}}w^m\_{ij}\delta^m\_i

\end{equation}

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

O ajuste dos pesos é dado por:

\begin{equation}

\label{eq:backpropagation\_function}

w^m\_{ij}(t+1) = w^m\_{ij}(t) + \alpha \delta^m\_i y^{m-1}\_j + \beta \Delta w^m\_{ij}(t-1)

\end{equation}

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

Durante o treinamento os exemplos vão sendo apresentados à MLP de forma aleatória e os pesos vão sendo ajustados pelo algoritmo \textit{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 \textit{underfitting} (os pesos da MLP não são ajustados de forma satisfatória), e caso o treinamento se torne excessivamente longo, pode ocorrer o \textit{overfitting} (a rede memoriza os exemplos de treinamento e ocorre perda na capacidade de generalização) \cite{valenca1995fundamentos}. Objetivando evitar essas duas condições, e consequentemente uma melhor capacidade de generalização, a base de dados para treinamento da rede pode ser dividida em duas partes: uma parte de fato para treinamento e uma outra parte para validação. A parte para treinamento é apresentada à rede e serve para realizar o ajuste dos pesos pelo algoritmo de treinamento. A parte correspondente ao conjunto de validação serve para realizar uma avaliação do treinamento da rede. Quando o erro médio quadrático para o conjunto de validação começar a aumentar, significa que a MLP está memorizando o conjunto de treinamento e neste momento é adequado interromper o treinamento.

As redes MLP são uma boa ferramenta para a construção de classificadores, já que além de conseguir lidar com o mapeamento entrada-saída não linear, apresentam alto poder de generalização \cite{haykin-1994}.

Em \cite{HUHUANG2007}, 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.

**Support Vector Machine**

A Máquina de Vetor de Suporte, do inglês \textit{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 \cite{HAYKIN2007} \cite{valenca2005aplicando}. 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 \ref{fig:svm}. Na Figura \ref{fig:svm}, 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.

\begin{figure}[h]

\centering

\includegraphics[width=.6\textwidth]{image/svm.png}

\caption{Vetores de Suporte e Hiperplano de Separação ótimo}

\label{fig:svm}

\end{figure}

As SVMs foram desenvolvidas tendo por base a teoria do aprendizado estatístico, detalhada por Vapnik em 1995 e 1998 \cite{vapnikSnature} \cite{vapnik1998statistical} e inicialmente proposta por Vapnik e Chervonenkis em 1971 \cite{vapnik1971uniform}.

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 \cite{vapnik1998statistical}. Entretanto, para problemas de aprendizado supervisionado, em geral não se dispõe da distribuição de probabilidade da população,uma vez que em problemas reais o que se utiliza é uma amostra da população sendo possível apenas o cálculo de uma função erro, como por exemplo o erro médio quadrático. Esta função erro calculada da amostra de dados é chamada de Risco Empírico. Logo, a depender do tamanho da amostra utilizada para treinamento, a minimização do erro de treinamento, não significa necessariamente uma minimização do erro de generalização ou Risco Funcional.

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

Desta forma, de acordo com a teoria do aprendizado estatístico proposta por Vapnik \cite{vapnik1998statistical}, 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 \ref{eq:svm\_functional\_risk}.

\begin{equation}

\label{eq:svm\_functional\_risk}

R\_{funcional} \leq R\_{empirico} + R\_{estrutural}

\end{equation}

O Risco Estrutural por sua vez pode ser calculado, conforme mostrado por Vapnik \cite{keylist} como:

\begin{equation}

\label{eq:svm\_structural\_risk}

R\_{estrutural} = \sqrt{\frac{h \left( \log \left( \frac{2N}{h} \right) + 1 \right) - \log \left( \frac{\delta}{4} \right)}{N}}

\end{equation}

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 \textit{perceptron} está na forma de seleção do hiperplano de separação das classes. Enquanto numa rede \textit{perceptron} se busca qualquer hiperplano que satisfaça o problema, numa SVM de margem rígida se procura o hiperplano ótimo, ou seja, aquele cuja margem de separação é máxima.

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

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

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

\begin{eqnarray}

\label{eq:svm\_example}

w^T.x\_i + b \geq +1 ,& se& y\_i= +1 \\

w^T.x\_i + b \leq -1 ,& se& y\_i= -1

\end{eqnarray}

ou de forma resumida

\begin{eqnarray}

\label{eq:svm\_example\_summary}

y\_i (w^T.x\_i + b) \geq 1 ,&& i=1,..,n

\end{eqnarray}

A margem (d) é a distância entre os dois planos paralelos e pode ser expressa como uma distância euclidiana, na Equação \ref{eq:svm\_euclian\_distance}.

\begin{equation}

\label{eq:svm\_euclian\_distance}

d = \frac{| w^T.x\_i + b |}{|| w ||} + \frac{| w^T.x\_i + b |}{|| w ||} = \frac{2}{|| w ||}

\end{equation}

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 \ref{eq:svm\_example\_summary}.

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.

**Radial Basis Function Network**

As Redes de Função de Base Radial \textit{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:

\item 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;

\item 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;

\item A camada de saída é composta por unidades de processamento lineares.

As funções de ativação mais utilizadas são \cite{valenca2005aplicando} \cite{engelbrecht2007computational}:

\item Função Linear

\begin{equation}

\label{eq:rbf\_linear\_function}

\phi\_i(x) = || x\_j - \mu\_i ||

\end{equation}

\item Função Cúbica

\begin{equation}

\label{eq:rbf\_cubic\_function}

\phi\_i(x) = || x\_j - \mu\_i ||^3

\end{equation}

\item Função de base Gaussiana

\begin{equation}

\label{eq:rbf\_gaussian\_function}

\phi\_i(x) = exp \left( - \frac{|| x\_j - \mu\_i ||^2}{2\sigma\_{i}^{2}} \right)

\end{equation}

\item Função Logística

\begin{equation}

\label{eq:rbf\_logistic\_function}

\phi\_i(x) = \frac{1}{1 + exp \left( - \frac{|| x\_j - \mu\_i ||^2}{\sigma\_{i}^{2} - \theta} \right)}

\end{equation}

\item Função de base Multi-quadrática

\begin{equation}

\label{eq:rbf\_multiquadratic\_function}

\phi\_i(x) = \sqrt{|| x\_j - \mu\_i ||^2 + \sigma\_i^2}

\end{equation}

\item Função de base Multi-quadrática inversa

\begin{equation}

\label{eq:rbf\_inverse\_multiquadratic\_function}

\phi\_i(x) = \frac{1}{\sqrt{|| x\_j - \mu\_i ||^2 + \sigma\_i^2}}

\end{equation}

\item Função de base lâmina Spline fina

\begin{equation}

\label{eq:rbf\_spline}

\phi\_i(x) = \frac{|| x\_j - \mu\_i ||}{\sigma\_i^2} \log \left( \frac{|| x\_j - \mu\_i ||}{\sigma\_i} \right)

\end{equation}

onde, $x\_j$ são exemplos de entrada, $\mu\_i$ e $\sigma\_i$ representam respectivamente o centro e a largura (dispersão) da i-ésima função de base radial e $\theta$ é um bias ajustado.

O cálculo da resposta da rede RBF correspondente aos neurônios da camada de saída é realizado pela Equação \ref{eq:rbf\_output\_formula}.

\begin{equation}

\label{eq:rbf\_output\_formula}

y\_j = \sum\_{j=1}^{s} w\_{Kj} \phi\_K(x)

\end{equation}

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 \cite{musavi1992training} e mapas auto-organizáveis de Kohonen \cite{valenca1995fundamentos}.

O método k-média tem por objetivo encontrar um conjunto de $K$ centros $\mu\_i$ (i=1,2,...,$K$) representativos das funções de base radial em função do conjunto de exemplos de entrada $x\_j$ com $j=1,2,...,N$. O algoritmo divide o conjunto de exemplos de entrada em K subconjuntos $S\_i$cada um deles contendo $N\_i$ exemplos. O objetivo do método é minimizar a Equação \ref{eq:rbf\_minimization\_function}.

\begin{equation}

\label{eq:rbf\_minimization\_function}

F = \sum\_{i=1}^{K} \sum\_{j \in S\_i} || x\_j-\mu\_i ||^2

\end{equation}

onde, $\mu\_i$ é a média dos pontos pertencentes ao conjunto $S\_i$ calculada por

\begin{equation}

\label{eq:rbf\_mi\_i}

\mu\_i = \frac{1}{N\_i} \sum\_{j \in S\_i} x\_j

\end{equation}

Inicialmente, os exemplos de entrada são localizados aleatoriamente dentro de cada subconjunto $K$ e então os centros $\mu\_i$ 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) $\sigma\_i$ 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 (\textit{Orthogonal Least Squares}) \cite{chen1991orthogonal}.

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

As redes RBF por serem redes de aprendizado local, necessitam de uma quantidade maior de exemplos para o seu treinamento, para que se obter uma precisão similar a das redes MLP, que são redes de ajuste global. Isto implica que as redes MLP em geral são melhores aproximadores de funções, pois o ajuste global tende a fornecer uma maior capacidade de generalização. Entretanto, em problemas de classificação, as redes MLP tendem a cometer maiores erros de classificação "falso positivo" do que as redes RBF \cite{valenca1995fundamentos}.

**Learning Rules**

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

**Gradient Descent Learning**

O Gradiente Descendente (GD) requer a definição de uma função erro (ou objetivo) para medir o erro do neurônio na aproximação do alvo, a soma quadrática dos erros é normalmente usada \ref{eq:sse}. 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 \ref{fig:gd} \cite{engelbrecht2007computational}.

\begin{equation}

\label{eq:sse}

\epsilon = \sum\limits\_{i=1}^{P\_T}(e\_i - c\_i)^2

\end{equation}

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.

\begin{figure}[h]

\centering

\includegraphics[width=.6\textwidth]{image/gd.png}

\caption{Vetores de Suporte e Hiperplano de Separação ótimo}

\label{fig:gd}

\end{figure}

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

\begin{equation}

\label{eq:gd1}

v\_i(t) = v\_i(t-1) + \Delta v\_i(t)

\end{equation}

onde

\begin{equation}

\label{eq:gd2}

\Delta v\_i(t) = \eta (- \frac{\partial \epsilon}{\partial v\_i})

\end{equation}

e $\eta$ 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 \cite{engelbrecht2007computational}.

**Widrow-Hoff Learning**

A regra de aprendizagem Widrow-Hoff assume que se a função objetivo $f = net\_i$, então $\frac{\partial f}{\partial net\_i}$. Essa regra de aprendizagem também é conhecida como o algoritmo \textit{least-mean-square} (LMS), foi um dos primeiros algoritmos usados para treinar redes neurais em camadas com múltiplos neurônios lineares adaptativos (Madaline) \cite{engelbrecht2007computational}.

**Error Correction Learning**

A regra delta é uma generalização da regra de Widrow-Hoff \cite{widrow1960adaptive} 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:

\begin{equation}

\label{eq:deltarule}

\Delta w\_{ij} = \alpha(d\_i - y\_i)x\_jf'(net\_i)

\end{equation}

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 \cite{haykin-1994}. Assim, o ajuste a ser aplicado aos pesos é:

\begin{equation}

\label{eq:peso\_novo}

w\_{ij}(new) = w\_{ij}(old) + \alpha(d\_i - y\_i)x\_jf'(net\_i)

\end{equation}

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 \cite{beale2010neural}.

**Memory based Learning**

Na aprendizagem baseada em memória, todas as (ou a maioria das) experiências passadas são armazenadas explicitamente em uma grande memória de exemplos de entrada-saída classificados corretamente: $ \{ ( x\_i, d\_i ) \}^{N}\_{i=1}$, 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 $\zeta\_1$ e $\zeta\_2$. Neste exemplo, a resposta desejada $d\_i$ assume o valor 0(ou -1) para a classe $\zeta\_1$ e o valor 1 para a classe $\zeta\_2$. 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:

\item O critério utilizado para definir a vizinhança local do vetor de teste $x\_teste$;

\item 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

\begin{equation}

\label{eq:nearestneighborhood\_vector}

x'\_N \in \{x\_1, x\_2, ..., x\_N\},

\end{equation}

é o vizinho mais próximo de $x\_teste$ se

\begin{equation}

\label{eq:nearestneighborhood}

\min\_i d(x\_i, x\_teste) = d(x'\_N,x\_teste),

\end{equation}

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.

**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 \ref{eq:scg1} \cite{engelbrecht2007computational}:

\begin{equation}

\label{eq:scg1}

\varepsilon\_T(D\_T,w) = \frac{1}{2}w^THw - \theta^Tw

\end{equation}

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, $p^T(t\_1)wp(t\_2) = 0$ 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 \cite{engelbrecht2007computational}.

Os fatores de escala também pode ser calculado através de métodos Polak-Ribiere e Hestenes-Stiefer. Moller \cite{moller1993scaled} propôs o algoritmo de gradiente conjugado escalonado (GCS) como um algoritmo de aprendizado \textit{batch}. Os tamanhos do passo são determinados automaticamente e o algoritmo é reiniciado após $n\_w$ iterações se uma boa solução não foi encontrada.

**Quasi-Newton Learning**

As variações em algoritmos de minimização são o gradiente e a matriz Hessiana. O gradiente deve ser conhecido com precisão assim como as direções de descidas devem ser calculadas a partir delas mesmas e aproximações ao gradiente não oferecem a precisão necessária. Por outro lado, a matriz Hessiana pode ser aproximada pela técnica das secantes. Uma vez que a matriz Hessiana é a matriz Jacobiana de um sistema de equações não-lineares $\nabla F(x) = 0$, ela pode ser aproximada. porém, a matriz Hessiana tem duas propriedades importantes: ela é sempre simétrica e, frequentemente, positiva. A incorporação dessas duas propriedades na aproximação secante é um aspecto importante dos métodos \textit{Broyden-Fletcher-Goldfarb-Shanno back-propagation} (BFGS-BP) e \textit{One Step Secant back-propagation} (OSS-BP) discutidos adiante. Eles são mais frequentemente chamados de atualizações secantes positivas definidas \cite{saini2002artificial}.

**Broyden-Fletcher-Goldfarb-Shanno back-propagation method**

No método de Newton uma aproximação quadrática é utilizada ao invés de uma aproximação linear da função $F(x)$. A próxima solução aproximada é obtida num ponto que minimize a função quadrática na Equação \ref{eq:bfgs1}.

\begin{equation}

\label{eq:bfgs1}

F(x\_{k+1}) = F(x\_k + \Delta x\_k) = F(x\_{k})+ g^T\_k\Delta x\_k + \frac{1}{2}\Delta x^T\_k A\_k \Delta x\_k

\end{equation}

Assim, a sequência obtida é a Equação \ref{eq:bfgs2}.

\begin{equation}

\label{eq:bfgs2}

x\_{k+1} = x\_k - A^{-1}\_k g\_k

\end{equation}

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 \ref{eq:bfgs3} \cite{saini2002artificial}.

\begin{equation}

\label{eq:bfgs3}

F(x\_{k+1}) = F(x\_{k})+ g^T\_k\Delta x\_k + \frac{1}{2}\Delta x^T\_k H\_k \Delta x\_k

\end{equation}

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 \cite{saini2002artificial}.

**One-step Secant backpropagation method**

Uma desvantagem da atualização do BFGS-BP na Equação \ref{eq:bfgs3} 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 \ref{eq:oss1} \cite{saini2002artificial},

\begin{equation}

\label{eq:oss1}

p\_{k+1} = - p\_{k+1} + B\_k y\_k +C\_k s\_k

\end{equation}

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)

\begin{equation}

\label{eq:oss2}

B\_k = \frac{s^T\_k g\_{k+1}}{s^T\_k y\_k}

\end{equation}

e

\begin{equation}

\label{eq:oss3}

C\_k = 1 \left(1 + \frac{y^T\_k y\_k}{s^T\_k y\_k} \right) \frac{s^T\_k g\_{k+1}}{s^T\_k y\_k} + \frac{y^T\_k g\_{k+1}}{s^T\_k y\_k} .

\end{equation}

A linha de busca de \textit{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 $\left(1 + \frac{y^T\_k y\_k}{s^T\_k y\_k} \right)$ 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 \cite{saini2002artificial}.

**Levenberg-Marquardt Algorithm**

O Algoritmo de Levenberg-Marquardt adaptativamente varia as atualizações de parâmetros entre a atualização do gradiente descendente e a atualização de Gauss-Newton \cite{marquardt1963algorithm},

\begin{equation}

\label{eq:scg}

\varepsilon\_T(D\_T,w) = \frac{1}{2}w^THw - \theta^Tw

\end{equation}

onde pequenos valores de parâmetros do algoritmo $\lambda$ resulta em uma atualização Gauss-Newton e grandes valores de $\lambda$ resultam numa atualização do gradiente descendente. O parâmetro $\lambda$ é inicializado para ser grande. Se uma iteração acontece de resultar numa aproximação pior, $\lambda$ é aumentado. À medida que a solução se aproxima do mínimo, $\lambda$ 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 \cite{marquardt1963algorithm}.

O algoritmo sugeriu uma relação de atualização como na Equação \ref{eq:LMupdate}.

\begin{equation}

\label{eq:LMupdate}

\varepsilon\_T(D\_T,w) = \frac{1}{2}w^THw - \theta^Tw

\end{equation}

**Neuro-Fuzzy Systems**

**ANFIS: Adaptive Neuro-Fuzzy Inference Systems**

Nesta seção, apresentamos uma classe de redes adaptativas que são funcionalmente equivalentes aos sistemas de inferência Fuzzy, analisados durante esse estudo. A arquitetura é denominada como ANFIS, que significa \textit{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 \cite{jang1997neuro}.

\begin{figure}[h]

\centering

\includegraphics[width=.7\textwidth]{image/anfis.png}

\caption{(a) Um modelo fuzzy de Sugeno de primeira ordem com duas entradas e duas regras; (b) arquitetura equivalente ANFIS}

\label{fig:anfis}

\end{figure}

**ANFIS Architecture**

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

\begin{enumerate}

\item Regra 1: Se $x$ é $A\_1$ e $y$ é $B\_1$, então $f\_1 = p\_1x + q\_1y + r\_1,$

\item Regra 2: Se $x$ é $A\_2$ e $y$ é $B\_2$, então $f\_2 = p\_2x + q\_2y + r\_2.$

\end{enumerate}

%\begin{enumerate}

%\item Rule 1: If $x$ is $A\_1$ and $y$ is $B\_1$, then $f\_1 = p\_1x + q\_1y + r\_1,$

%\item Rule 2: If $x$ is $A\_2$ and $y$ is $B\_2$, then $f\_2 = p\_2x + q\_2y + r\_2.$

%\end{enumerate}

A Figura \ref{fig:anfis}(a) ilustra o mecanismo de raciocínio para esse modelo de Sugeno; a arquitetura ANFIS equivalente correspondente é como mostrada na Figura \ref{fig:anfis}(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 $O\_{1,i}$).

\textbf{Camada 1}: Cada nó $i$ nessa camada é um nó adaptativo com um função:

\begin{equation}

\label{eq:layer1anfis}

O\_{1,i} = \mu\_{A\_i}(x), for i = 1,2 or \\

O\_{1,i} = \mu\_{B\_{i-2}}(y), for i = 3,4

\end{equation}

em que $x$ (ou $y$) é a entrada para o nó $i$ e $A\_i$ (ou $B\_{i-2}$) é um rótulo linguístico (tal como "pequeno" ou "grande") associado a esse nó. Em outras palavras, $O\_{1,i}$ é o grau de pertinência a um conjunto fuzzy $A (A\_1, A\_2, B\_1 or B\_2)$ e ele especifica o grau em que a entrada de dados $x$ (ou $y$) satisfaz o quantificador $A$. Aqui, a função de pertinência para $ A $ pode ser qualquer função de pertinência parametrizada apropriada, como a função sino generalizada \cite{jang1997neuro}:

\begin{equation}

\label{eq:bell\_anfis}

\mu\_{A}(x) = \frac{1}{1 + \left\| \frac{x - c\_i}{a\_i} \right\|^{2b}},

\end{equation}

\textbf{Camada 2}: Cada nó nessa camada é um nó fixo rotulado $\Pi$, cuja saída é o produto de todos os sinais de entrada:

\begin{equation}

\label{eq:layer2anfis}

O\_{2,i} = w\_i = \mu\_{A\_i}(x)\mu\_{B\_i}(y) for i = 1,2.

\end{equation}

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.

\textbf{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:

\begin{equation}

\label{eq:layer3anfis}

O\_{3,i} = \overline{w\_i} = \frac{w\_i}{w\_1 + w\_2} for i = 1,2.

\end{equation}

Por conveniência, as saídas dessa camada são chamadas \textbf{forças de disparo normalizadas}.

\textbf{Camada 4}: Every node $i$ in this layer is an adaptive node with a node function

\begin{equation}

\label{eq:layer4anfis}

O\_{4,i} = \overline{w\_i}f\_i = \overline{w\_i}(p\_ix + q\_iy + r\_i),

\end{equation}

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".

\textbf{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:

\begin{equation}

\label{eq:layer5anfis}

overall output = O\_{5,i} = \sum\limits\_{i}\overline{w\_i}f\_i = \frac{\sum\_{i} w\_if\_i}{\sum\_{i}w\_i},

\end{equation}

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 \cite{jang1997neuro}.

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 \cite{jang1997neuro}.