LEONARDO NASCIMENTO FERREIRA

UM TRADING SYSTEM AUTÔNOMO BASEADO EM REDES NEURAIS ARTIFICIAIS

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Orientador

Prof. Cristiano Leite de Castro

LAVRAS
MINAS GERAIS – BRASIL
2009

LEONARDO NASCIMENTO FERREIRA

UM TRADING SYSTEM AUTÔNOMO BASEADO EM REDES NEURAIS ARTIFICIAIS

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

APROVADA em _____de _____de _____.

_____
Prof. Thiago de Souza Rodrigues

_____
Prof. Wilian Soares Lacerda

_____
Prof. Cristiano Leite de Castro
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2009

Aos meus pais, Silvana e Ludgero

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

UM TRADING SYSTEM AUTÔNOMO BASEADO EM REDES NEURAIS ARTIFICIAIS

Resumo

Existem hoje no mercado inúmeros sistemas eletrônicos que, a partir de gráficos históricos e dados atuais, conseguem prever a tendência dos preços das ações. Esses trading systems fazem a análise técnicas dos preços observando vários índices e procurando por padrões já conhecidos em séries históricas. O grande problema com esses sistemas é que são limitados ao conhecimento do especialista o qual determinou os padrões. Assim, utilizando Redes Neurais Artificiais e gráficos de *Candlesticks*, propõem-se a criação de um trading system que faça a analise gráfica, aprenda de forma autônoma a reconhecer padrões em séries e determine automaticamente gatilhos de compra e venda de uma determinada ação.

Palavras chave: Redes Neurais Artificiais; Análise Técnica; Candlesticks.

AN AUTONOMOUS TRADING SYSTEM BASED ON ARTIFICIAL NEURAL NETWORKS

Abstract

Recently, several computational systems have used historical prices series and economic indexes to predict share prices in stock market. In general, these trading systems are based on technical analysis tools which preprocess data in order to find buying and selling patterns. The problem of this approach is that these patterns should be indicated by specialists in stock market. To overcome this problem, we propose an autonomous trading system based on Artificial Neural Networks, candlesticks graphs and stochastic index. Our trading system is able to learn patterns by itself and automatically to predict buying and selling triggers of a specific share.

Keywords: Artificial Neural Network; Technical Analysis; Candlesticks;

1.INTRODUÇÃO

Com o surgimento das bolsas de valores, inúmeros estudos foram realizados afim de tentar encontrar explicações para a variação dos preços das ações, mas especificamente, tentar obter lucro com a compra e venda das ações. A princípio, acreditava-se que esse comportamento era apenas reflexo de acontecimentos referentes a empresa como por exemplo, a notícia de descoberta de ouro por uma mineradora provavelmente elevaria o preço do papel. Porém, a menos que o investidor tenha acesso a informações privilegiadas, é muito difícil acompanhar e prever notícias como essas.

No século XVIII, acredita-se que um japonês negociante de arroz chamado Homma Munehisa, inventou um gráfico para representar o preço das ações conhecido como Gráfico de *Candlesticks*. Essa representação possibilitou inúmeros estudos matemáticos, mas ainda se acreditava que não era possível, a partir de gráficos, prever o comportamento das ações. Pensava-se que os eventos eram aleatórios.

No final do século XIX, o jornalista americano Charles Henry Dow criou a teoria, que mais tarde sustentou a Análise Técnica. Essa teoria, afirma que o mercado desconta tudo, ou seja, em algum momento, a descoberta de ouro pela mineradora será refletida nos gráficos de preço. Assim, passou-se a acreditar que toda a Análise Fundamentalista, aquela baseada em notícias, está englobada na Análise Técnica.

Com o passar do tempo e o avanço nos estudos relacionados a engenharia financeira, descobriu-se que os gráficos de *Candlesticks* facilitavam a observação de padrões que antecipavam pontos de reversão ou confirmavam a

tendência dos preços. Além desses padrões, vários indicadores, baseados também em gráficos e fórmulas matemáticas, foram criados para funcionarem como gatilhos, ou seja, para tentarem responder a pergunta que todos os investidores gostariam de saber: Qual o melhor momento para comprar ou vender um determinado papel?

A resposta para essa pergunta ainda é muito difícil de se descobrir, porém com o surgimento dos computadores e a internet, os sistemas eletrônicos aumentaram, de forma significativo, a taxa de acertos. Esses trading Systems viabilizaram o investimento além de popularizar o mercado entre pequenos investidores tornando-se poderosas ferramentas de auxilio e suporte a decisão.

Sistemas de apoio à tomada de decisão incorporam ferramentas computacionais para mineração de dados, previsão e reconhecimento de padrões comportamentais do mercado. Dentre as ferramentas, destacam-se, no ramo da Inteligência Artificial, as Redes Neurais Artificiais (RNAs). RNAs são modelos inspirados no aprendizado dos neurônios biológicos e têm sido extensivamente utilizadas em aplicações na área de previsão (regressão) e reconhecimento de padrões. Especialmente, para o mercado de ações, inúmeros trabalhos têm revelado a eficácia de tal ferramenta como auxílio à decisão do investidor.

No contexto de gráficos de *Candlesticks*, Análise Técnica e RNAs, propõem-se o desenvolvimento de um *tranding system* que faça a análise gráfica, aprenda de forma autônoma a reconhecer padrões em séries financeiras e determine automaticamente os pontos de reversão (gatilhos de compra e venda) para uma determinada ação.

Para alcançar esse objetivo, será necessário o desenvolvimento de um parser que irá percorrer as séries temporais e armazenar os padrões de reversões em um arquivo que será utilizado como conjunto de treinamento para uma RNA.

A construção desse sistema é interessante pois ainda não se conhece um sistema independente que utilize *candlesticks*. Apenas sistemas tutores e de apoio a decisão foram implementados.

2. REFERENCIAL TEÓRICO

Tendo em vista toda a problemática, se vê necessário uma revisão bibliográfica abordando os principais tópicos citados.

2.1 ANÁLISE TÉCNICA

A Análise Técnica, é o estudo do comportamento dos preços das ações utilizando gráficos, por isso também chamada de Análise Gráfica [1]. A partir desse estudo, muitas teorias surgiram para tentar explicar comportamentos como a reversão de tendência nos preços. Vários índices, expressos em fórmulas matemáticas, foram criados com o intuito de tentar prever e explicar comportamentos notados nos gráficos.

Antes do surgimento da Análise Técnica, acreditava-se que somente era possível prever comportamento dos preços analisando os fatos que ocorriam envolvendo a empresa. Se a empresa está crescendo financeiramente provavelmente o preço das ações deve aumentar. Se uma empresa petrolífera encontrar um poço de petróleo, o preço do ativo deve subir. Assim, uma análise interna da empresa é feita estudando gastos, lucro, greves, fatores políticos, notícias envolvendo a empresa em geral, a fim de tentar observar fatores que pudessem levar ao crescimento e conseqüentemente, ao aumento do preço do papel. Esse estudo, também conhecido como Análise Fundamentalista, foi usado durante muito tempo e ainda é usado. Entretanto, trata-se de estudo específico de cada empresa, sendo muitas vezes difícil de prever ou encontrar notícias como essas.

2.1.1 A teoria de Dow

O principal motivo que deu força a Análise Técnica foram as teorias do norte americano Charles H. Dow que após realizar estudo sobre comportamento dos preços das ações da bolsa de Nova Iorque, levantou três pontos principais [1]. Vejamos suas conclusões.

2.1.2 O mercado desconta tudo

Isso significa que os índices refletem o conjunto de negociação de todos os investidores, seja este um investidor leigo ou um detentor de uma informação privilegiada. Conseqüentemente já estão descontados nos índices informações desconhecidas pelo grande público. Voltando ao exemplo da mineradora que encontrou ouro, provavelmente os diretores da empresa serão os primeiros a comprar ações da empresa, logo que possuem informação privilegiada em relação aos outros investidores. Isso começa a elevar os preços antes mesmos dos outros investidores saberem da notícia.

Além de informações como essas, existem outros tipos de acontecimentos que são imprevisíveis. São exemplos as catástrofes naturais, guerras, etc, que podem gerar fortes oscilações iniciais porém que com o passar do tempo, são absorvidas pelo mercado.

2.1.3 As tendências de mercado

As tendências são movimentos nos gráficos onde predominou uma dada

direção, sendo essa de alta ou baixa nos preços. Elas podem ser observadas em diferentes espaços de tempos e são classificadas como:

- Primárias: São tendências observadas em anos, por exemplo, 1 ou 2 anos.
- Secundárias: Estão contidas nas tendências primárias e são contadas em meses podendo ser notadas de 2 a 5 meses.
- Terciárias: Possuem tempo menor e estão contidas nas tendências secundárias sendo notadas entre semanas e dias.

Não existe um número exato para definir o período correspondente às tendências sendo a única regra o fato das tendências terciárias estarem contidas nas secundária e essas contidas nas primárias.

2.1.4 Tendência dividida em três fases

Considerando uma tendência de alta, desde o início até o fim do movimento, ela pode ser divida em três fases:

- 1ª fase: leve alta nos preços decorrente da compra de ações por investidores que provavelmente detêm informações privilegiadas;
- 2ª fase: alta mais rápida refletindo a entradas dos investidores experientes;
- 3ª fase: período onde o investidor leigo é levado pela euforia do mercado e compra ações. É caracterizado por grandes altas do preço mas que provavelmente é seguida de reversão de tendência

que é percebida pelo investidor experiente que vende suas ações.

Existem também 3 fases para a tendência de baixa que normalmente ocorre depois de uma tendência de alta. São elas:

- 1ª fase: inicia quando o investidor experiente vende suas ações;
- 2ª fase: a incerteza do mercado faz com que os investidores comecem a vender suas ações com receio de haver prejuízo maior;
- 3ª fase: onde os prejuízos são altos sendo seguida de reversão de tendências. No fim dessa fase o investidor experiente compra a ações.

2.2 GRÁFICOS DE CANDLESTICKS

Os gráficos de *candlesticks* surgiram no Japão durante o século XVIII e atribui-se sua criação ao negociante de arroz, Munehisa Honma que ganhou fama rapidamente devido a facilidade de observar o movimento dos preços das ações [2].

O Gráfico é formado por *candles*, que representam os preços de abertura, máximo, mínimo e fechamento de um dado papel durante o dia [2]. O *candle* é formado por uma barra indicando os preços de abertura e fechamento, podendo ter também uma linha vertical, chamada de sombra, que representa os preços de máximo e mínimo. O corpo do *candle* pode ser formado por duas cores, uma determina dia de alta e outra baixa nos preços. Na figura 1, pode-se observar dois *candles* e a figura 3 ilustra um exemplo de gráfico de *candlesticks.*

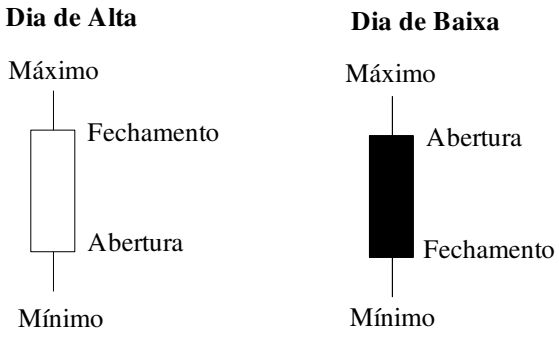

Figura 1 – *Candles*.

Figura 2 – Exemplo de gráfico de *Candlesticks*.
Fonte: http://www.fxribafree.com/

Com o passar do tempo, os acionistas começaram a notar padrões nos gráficos e começaram a usá-los para tentar prever o comportamento das tendências, seja ela de queda nos preços, alta ou confirmação de que a tendência atual ainda continuará. Começou-se então a tentar extrair informações contidas no gráfico e interpretá-las para tentar entender a dinâmica e motivos que levavam à queda ou alta dos preços.

Assim, vários relações e explicações foram retiradas a partir das observações e estudos. Uma delas foi a diferença entre o tamanho dos *candles*, observada na figura 4:

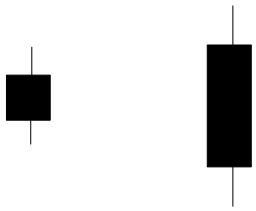

Figura 3 – *Candles* de diferentes tamanhos.

Segundo [1], Os menores *candles* significam pouca negociação e exibem uma possível estagnação do preço. Por outro lado, os *candles* maiores significavam alto número de negociações em um único dia, mostrando a instabilidade do papel e possível queda ou alta do preço.

Outra conclusão de [1] foi realizada quando observado os gráficos sem sombras (linhas verticais). Notou-se que quando o valor de fechamento correspondia também ao valor de máximo, significava que os compradores sobressaíram, indicando altas dos preços. A figura 5 exibe dois *candles* sem sombra.

Figura 4 - *Candles* sem sombra.

Vários outros padrões foram encontrados, inclusive padrões formados por mais de um *candle* e deram-se a eles nomes sugestivos, em sua maioria de origem japonesa [1]. Alguns padrões podem ser observados nas figuras 6 e 7.

Hoje, os gráficos de *candlesticks* ainda são muito utilizados e sua principal finalidade é de prever tendências dos preços.

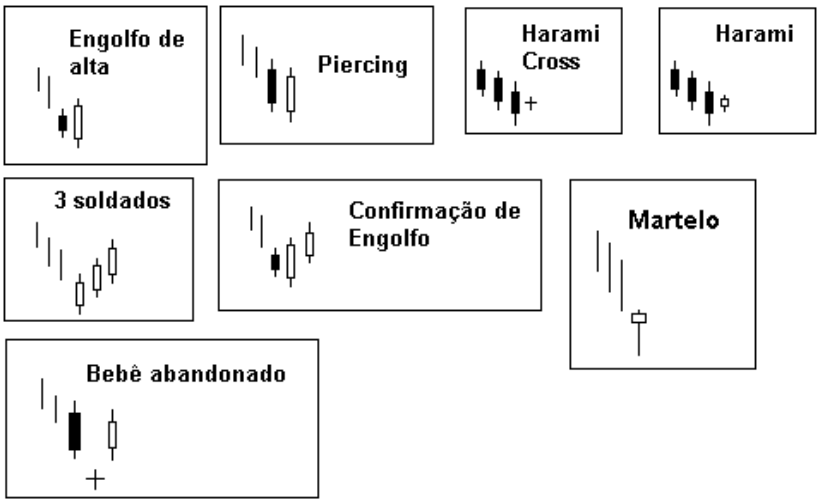

Figura 5 - Padrões que prevêem tendência de alta.
Fonte: http://www.grafbolsa.com

Figura 6 - Padrões que prevêem tendência de baixa.
Fonte: http://www.grafbolsa.com

2.3 INDICADORES TÉCNICOS

Além de observar a forma ou aparência do gráfico a fim de encontrar padrões, diversos pessoas criaram fórmulas matemáticas que utilizavam diversos valores de entrada para também tentar prever comportamentos. Esses valores geralmente são preços de abertura, fechamento, máximo, mínimo, volume de negociações, etc. Valores estes, observados em uma dada janela de tempo, que serviam de entrada para uma equação matemática e resultavam em um indicador que principalmente tenta prever comportamento dos preços [3].

Esses indicadores geralmente vão sendo calculados na mesma freqüência em que o gráfico de preço é atualizado, ou seja, tomando o gráfico de *candlestick* diário como exemplo, a cada novo dia um novo *candle* é adicionado ao gráfico, da mesma forma, o índice é recalculado, agora incluindo esse novo *candle*, obtendo assim o valor do indicador para aquele dado momento. Assim, é possível plotar um gráfico com todos esse valores, observar o comportamento do

indicador e utilizá-lo como suporte para tomada de decisões. Analisando esse gráfico é possível observar que alguns índices oscilam em uma dada faixa numérica e por isso são também chamados de osciladores [1].

O tamanho da janela de tempo utilizada para este cálculo tem importante influência. Janelas de tempos mais curtas são mais sensíveis e prevêem mais rapidamente um acontecimento, porém estão mais sujeitas a erros logo que percebem oscilações a curto prazo. Por outro lado, janelas de tempo mais longas são menos sensíveis e mais confiáveis, porém, às vezes, são perdidas oportunidades de compra ou venda de um ativo deixando de lucrar.

2.3.1 Índice estocástico

Dentre os indicadores, será utilizado nesse trabalho o oscilador estocástico. Segundo [1], esse índice mede a força dos compradores em levar o preço de fechamento de um papel mais próximo do preço máximo alcançado no dia e da força dos vendedores de levar o preço de fechamento mais próximo do preço mínimo.

Existem algumas variações para o cálculo desse indicador. Neste trabalho, optou-se pelo cálculo da linha rápida. Essa linha, denotada por $\%K$, mede a relação entre as forças de compra e venda. Essa linha é mais sensível e indica mais rapidamente um possível aumento das forças compradoras e, devido a sua rapidez, pode-se obter falsos resultados.

A fórmula matemática para essa linha é:

$$\%K = \frac{\mathrm{Fech}_{\mathrm{hoje}} - \mathrm{Min}_{\mathrm{n}}}{\mathrm{Max}_{\mathrm{n}} - \mathrm{Min}_{\mathrm{n}}} \times 100 \qquad (1)$$

Onde $Fech_{hoje}$ é preço do fechamento de hoje, Min_n e Max_n são os preços máximo e mínimo alcançados na janela de tempo que geralmente é de cinco dias.

Como mencionado acima, a principal finalidade do índice estocástico é definir se o papel está sobrevendido ou sobrecomprado. Para isso, são utilizadas margens superior e inferior para determinar situações de compra ou venda de um ativo. Quando o índice estocástico ultrapassar a margem superior, significa que o papel está sobrecomprado, indicando uma boa fase para a venda do papel, logo que uma reversão de tendência para queda está próxima. O mesmo ocorre quando o índice ultrapassa a margem inferior. Quando isso ocorre, o papel esta sobrevendido, ou seja, a maior força foi dos vendedores. Isso implica em um baixo preço do papel e uma boa oportunidade de compra, logo que uma reversão de tendência de alta nos preços deve ocorrer em breve.

Costuma-se utilizar 85% para definir a margem superior e 15% para definir a margem inferior [4]. Índices inferiores a 15% indicam reversão de tendência para alta, logo deve se comprar o papel. Índices superiores a 85% indicam reversão de tendência para baixa e sugerem a venda do papel. Valores de índices dentro dessa faixa indicam a permanência do papel.

Os valores para essas margens podem ser alteradas segundo o perfil do investidor. Investidores mais ousados podem aumentar o intervalo de permanência e utilizar valores como 10% para limite inferior e 90% para limite inferior. Isso faz com que pontos que eram de reversão para margens de 15% e 85% deixem de ser e conseqüentemente o investidor permanecerá mais tempo com um papel, dado que o indicador irá demorar mais para apontar reversão de tendência.

Aumentar a faixa de permanência pode trazer mais lucro. Tomando como

exemplo uma semana onde só houve alta nos preços. O estocástico com margens de 15% e 85% poderia informar na quinta-feira dessa semana que tendência de baixa está prevista, fazendo com que o investidor venda seu papel naquele dia. Porém, se este tivesse permanecido com o papel ele teria lucrado mais, logo que na sexta-feira ainda houve valorização. Portanto, se este investidor tivesse utilizado como margens 10% e 90%, ele teria lucrado mais. Isso pode parecer uma boa prática, porém deve se ter precaução, pois quando se aumenta essa faixa de permanência, diminui-se a faixas de sobrecomprado e sobrevendido. Assim uma situação inversa pode ocorrer. Supondo agora que nesta semana todos os dias foram marcados por baixa nos preços, o estocástico com margens de 15% e 85% indicaria na quinta-feira para vender o papel. O investidor que utilizou margens de 10% e 90% venderia o papel na sexta-feira e teria prejuízo maior. Assim, cada investidor deve ajustar esses limites segundo seu perfil de investimento, seja ele precavido ou ousado.

2.4 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados nas redes de neurônios biológicos existentes no cérebro humano. São caracterizadas por processarem os dados de forma paralela da mesma forma que o cérebro faz e utiliza o neurônio como unidade básica de processamento que é capaz de aprender e generalizar [5].

Sua principal característica, a generalização, permite que depois de treinado e aprendido a resolução de um problema, seja possível se obter uma resposta boa para um problema semelhante ainda desconhecido. Outra característica muito importante é capacidade de aproximar qualquer função

matemática seja ela contínua ou não, sendo o aprendizado da RNA nada mais que a tentativa de encontrar a melhor função que se aproxima a função desejada [5].

Existem inúmeras aplicações para essa ferramenta que resolve principalmente problemas de classificação, reconhecimento de padrões, previsão e extração de informações desconhecidas.

Tomando como exemplo o problema de previsão de tempo, podemos facilmente criar uma RNA que prevê se o próximo dia irá chover ou fazer sol. Para simplificar o problema, serão utilizadas apenas valores de temperatura e pressão para definir se o dia é chuvoso ou ensolarado (com certeza o problema não se resume à estas duas entradas), podemos plotar um gráfico com dados históricos já classificados e obter um gráfico de duas dimensões semelhante a figura 8.

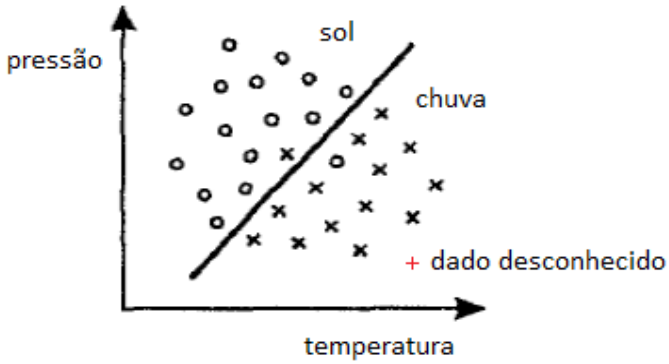

Figura 7 – Gráfico de temperatura e pressão para dias históricos, contendo a reta de separação de classes.

Assim, é possível treinar uma RNA com os dados históricos, e depois de treinada, é possível apresentar um par de valores de temperatura e pressão desconhecido, que a RNA irá prever se o dia será ensolarado ou chuvoso. Esse é

uma aplicação simples e didático de RNA podendo ser aplicadas a problemas com números maiores de variáveis de entrada e classes.

2.4.1 O neurônio artificial

A estrutura do neurônio artificial é composta de três partes que formam o neurônio biológico. São elas: dendritos, corpo (também chamado de soma) e axônio [5]. Os dendritos recebem os estímulos elétricos de outros neurônios e são levados ao corpo que "soma" os estímulos, excitando ou não o neurônio. Se este receber estímulo suficiente, o corpo irá deixar com que o estímulo seja transmitido pelo axônio para que os outros neurônios ligados a ele recebam o pulso elétrico. Assim, o neurônio é semelhante a uma função matemática que recebe valores de pulso elétrico e define uma saída. A figura 9 mostra a estrutura do neurônio artificial, também chamado de perceptron.

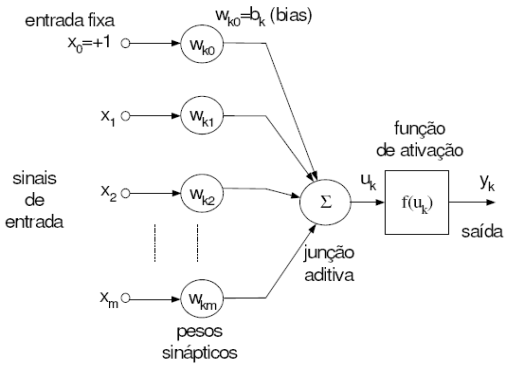

Figura 8 - Funcionamento do neurônio artificial.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 – [6]

Os dendritos são os sinais de entrada, representados por x_1, x_2, ... x_m. Estes sinais são multiplicados por seus respectivos pesos, representados por w_{k1},

w_{k2}, ... w_{km}. Os pesos determinam o quão influente é a entrada no processo de excitação do neurônio. Além das variáveis de entrada, existe uma entrada fixa que determina o grau de influência das entradas, também chamado de *bias*. Depois de multiplicar todos os valores de entrada por seus respectivos pesos, o corpo do neurônio (junção aditiva) realiza a soma ponderada das variáveis de entrada (u_k) e utiliza uma função de ativação para definir o valor de saída do neurônio (y_k).

As funções de ativação mais utilizadas são a tangente hiperbólica, sigmoidal e limiar, conforme ilustrado pelas equações 2, 3 e 4.

$$f = \tanh(\text{soma}) \qquad (2)$$

$$f = \frac{1}{1+\exp(-\text{soma})} \qquad (3)$$

$$f = 0 \text{ se soma} < 0$$
$$f = 1 \text{ se soma} > 0 \qquad (4)$$

Assim, o funcionamento do perceptron pode ser expresso através da seguinte expressão matemática:

$$y_k = f(u_k) = f\left(\sum_{j=1}^{m} w_{kj} x_j + b_k\right) \qquad (5)$$

Onde y_k é a saída do neurônio, u_k é a soma ponderada, f é a função de ativação, m o número de entradas, w o vetor de pesos da entrada, x o vetor de

entrada e *b* o bias.

Voltando ao exemplo de classificação de dias de sol e chuva, pode-se observar na figura 8 que é possível traçar uma reta de separação entre as duas classes. Pares de temperatura e pressão que se encontram a cima da reta de separação são da classe sol e pares inferiores a reta são da classe chuva. A fórmula matemática para essa reta é ilustrada pela equação 6.

$$(w_1 x_1)+(w_2 w_2)+b=0 \qquad (6)$$

2.5 TREINAMENTO DO PERCEPTRON

2.5.1 Treinamento

O treinamento do perceptron se resume à tarefa de tentar encontrar os melhores pesos para seja possível traçar melhor hiperplano de separação entre classes.

Essa tarefa pode ser trivial para problemas linearmente separáveis, como o exemplo de classificação de dias de sol e chuva (introduzido na seção 2.4), onde é possível traçar uma reta que separa as classes. Porém, essa tarefa pode ser muito difícil ou até mesmo impossível quando o problema é não linearmente separável, conforme ilustrado na figura 10. Supondo que ao plotar os valores de temperatura e pressão dos dados históricos, fosse obtido o gráfico da figura 10. É impossível encontrar uma reta de separação.

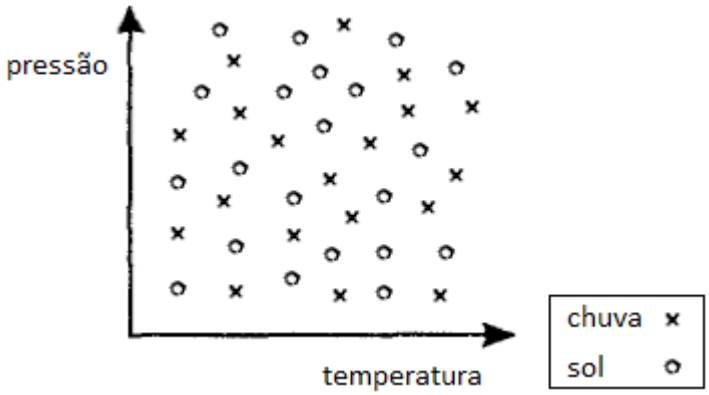

Figura 9 - Problema não linearmente separável.

Para realizar o treinamento é necessário que se tenha dados históricos previamente rotulados. Para o problema de classificação de dias, pode-se analisar os dias do ano passado e classificar cada dia como sendo chuvoso ou ensolarado. Esse conjunto de dias, também chamado de conjunto de treinamento, são apresentados ao neurônio que irá utilizá-los para ajustar seus parâmetros.

O conjunto de treinamento tem papel fundamental no aprendizado. Assim, quanto maior o número de amostras analisadas, melhor será a generalização do modelo. Outro cuidado é com disposição dos dados dentro do espaço de entradas possíveis. Se utilizarmos apenas dias com temperatura e pressão altas como conjunto de treinamento, o perceptron provavelmente não irá aprender a classificar dias com temperatura baixa. Para que o aprendizado seja eficiente, deve-se tentar apresentar ao perceptron dias com o maior número de combinações de temperatura e pressão possíveis.

Segundo [7], o algoritmo de treinamento do perceptron sempre encontra um bom hiperplano de separação para problemas linearmente separáveis em

tempo finito. É interessante ressaltar que um bom hiperplano de separação não precisa necessariamente dividir exatamente as classes, sendo que pequenos erros de classificação podem ocorrer.

2.5.2 O Algoritmo

O algoritmo de treinamento do perceptron segue os seguinte passos [6]:

1. Apresentar todos os dados do conjunto de treinamento para o perceptron. Esse período é chamado de época;
2. Para cada dado da época, ajustar os pesos utilizando a regra delta;
3. O conjunto de treinamento é apresentado novamente até que se tenha alcançado o número de épocas desejadas ou até que o erro seja menor ou igual ao erro de tolerância.

A regra delta é uma expressão matemática que atualiza cada peso de entrada com base no erro atual da época e a taxa de aprendizado, sendo calculada pela seguinte equação:

$$w(n+1)=w(n)+\eta\, e\, x(n) \qquad (7)$$

Onde:

- w(n+1) é o peso atualizado;
- w(n), o peso atual;
- η, a taxa de aprendizado, que indica a velocidade com que o vetor de peso será atualizado.

- *e*, o erro atual, que é a diferença entre a saída atual do perceptron com a saída desejada, conforme equação 8;
- x(n), o valor de entrada.

$$e = y_{esperado} - y_{obtido} \qquad (8)$$

A princípio, o vetor de pesos e o bias são iniciados com valores aleatórios e costuma-se utilizar valores entre 0,2 e 0,8 para a taxa de aprendizado. Não existe uma fórmula para calcular os melhores parâmetros para o algoritmo de treinamento[1]. Utilizar valor muito alto de taxa de aprendizado pode fazer com que o passo seja tão alto, que o algoritmo nunca encontrará uma boa solução. Assim, testes devem ser realizados para tentar encontrar a melhor combinação para tentar diminuir o tempo de aprendizado.

2.5.3 Momento

Uma técnica que acelera o tempo de aprendizado é chamada de momento e consiste em adicionar um fator ao cálculo de atualização dos pesos da regra delta [6]. Fazendo uma analogia ao sistemas mecânicos, o momento funciona como a inércia que faz com que uma esfera que deslize sobre uma superfície inclinada e continue seu caminho até que o atrito faça com que a esfera pare. De forma análoga, o circuito onde a esfera caminha pode ser o gráfico de erro para um perceptron de única entrada e a posição da esfera, o erro quadrático da época atual.

1 No Anexo I encontra-se a implementação do algoritmo de treinamento em forma de script para ser executado no Ambiente Scilab.

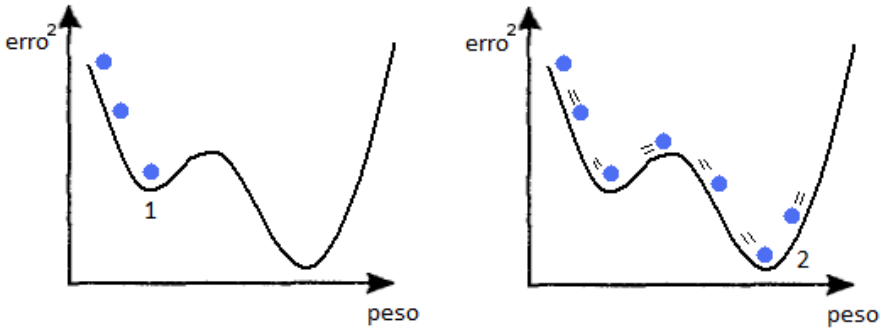

Figura 10 - Tempo de aprendizado sem momento (esquerda) e com momento (direita).

Na figura 11, podemos notar que o aprendizado com momento tende a levar menos épocas para encontrar o ponto de menor erro.

O momento também é útil para resolver o problema dos mínimos locais. Esse problema ocorre quando a função de custo possui mínimos locais, tornando-se "armadilhas" para o algoritmo de treinamento. Assim, tomando como exemplo a figura 12 a seguir, pode ser que a regra delta ficará presa no mínimo local, onde o gradiente da função de custo é nulo. Porém se fosse utilizado momento, a regra delta alcançaria o mínimo global.

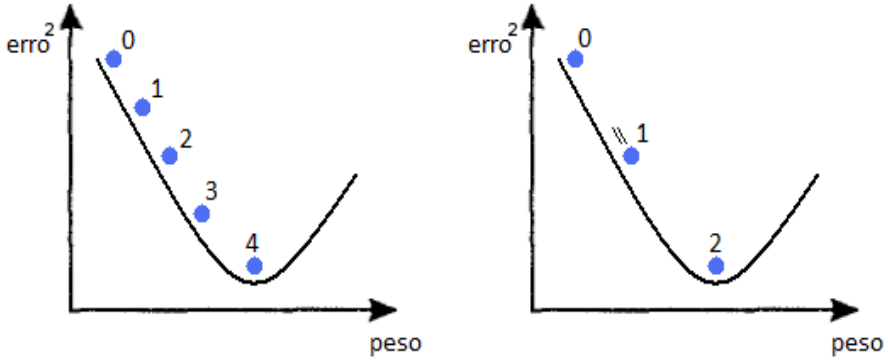

Figura 11 - Aprendizado sem momento (direita) e com momento (esquerda).

O uso de momento é aconselhado para diminuir o tempo de treinamento e solucionar o problema dos mínimos locais e costuma-se utilizar valores entre 0,1 e 0,5. Altas taxas de momento podem fazer com que o algoritmo de treinamento passe pelo mínimo global, prejudicando o treinamento, conforme mostrado pela figura 13.

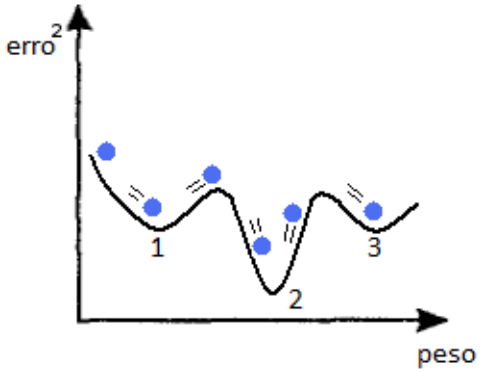

Figura 12 - Alta taxa de momento pode prejudicar treinamento.

2.6 PERCEPTRON DE MÚLTIPLAS CAMADAS

2.6.1 Histórico

Em 1969, Minsky e Papert provaram que o perceptron simples (de uma camada) só é capaz de resolver problemas linearmente separáveis, logo que só consegue traçar hiperplanos. Afirmaram também que por mais que já fosse provado que o treinamento de perceptron simples converge, o mesmo não ocorre para redes perceptron de mais de uma camada. Esse trabalho teve repercussão negativa e desanimou os pesquisadores a continuar nessa área. Contudo, alguns pesquisadores continuaram com seus trabalhos e mostraram mais tarde que

Minsky e Papert foram pessimistas [5].

Em 1974, Paul Werbos propôs o algoritmo backpropagation, porém somente em 1986 o algoritmo ganhou reconhecimento através do trabalho de David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams. A criação desse algoritmo marcou a retomada nos estudos em RNAs [5].

2.6.2 Redes de Neurônios

As Redes de neurônios são arquiteturas onde vários neurônios estão conectados entre si. As ligações entre os neurônios são chamadas de sinapses que transferem a saída de um neurônio à entrada do neurônio seguinte. Possuem as mesmas qualidades de um nerônio simples, porém são mais poderosas e mais eficientes podendo aproximar funções complexas [6].

2.6.3 Topologias

Existem três tipos principais de arquitetura, sendo elas:

- Feedforward de única camada;
- Feedforward de múltiplas camadas;
- Redes recorrentes.

As redes Feedforward, são caracterizadas pelo seu funcionamento de alimentação para frente, ou seja, nenhuma saída de um neurônio volta para sua entrada ou para a entrada de neurônios anteriores. Podem possuir uma ou mais camadas sendo a primeira chamada de camada de entrada, as seguintes são as intermediárias (também chamadas de camadas escondidas) e última a camada de

saída. É interessante ressaltar que, diferente do perceptron simples, a rede pode ter um vetor de valores de saída, logo que é formada de vários neurônios.

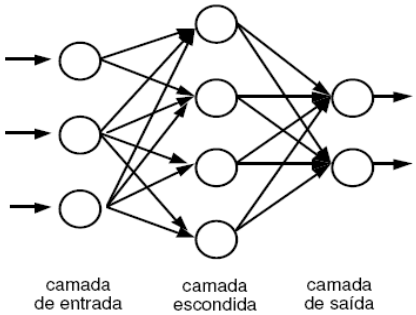

Figura 13 - Nomenclatura para as camadas das Redes de Neurônios.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 - [6]

O funcionamento é simples. Inicialmente, são apresentadas os valores para os neurônios da camada de entrada (neurônios sensoriais). Esses simplesmente repassam a informação para os neurônios da camada escondida, que irão de fato processar a entrada. Assim, para cada camada escondida, os valores são reprocessados e enviados para próxima camada. Se esta for a última camada, então será realizado o último processamento e que resultará em um vetor de saídas de tamanho igual ao número de neurônios da última camada. Para as redes feedforward de única camada, os dados são processados somente uma vez resultam no vetor de respostas.

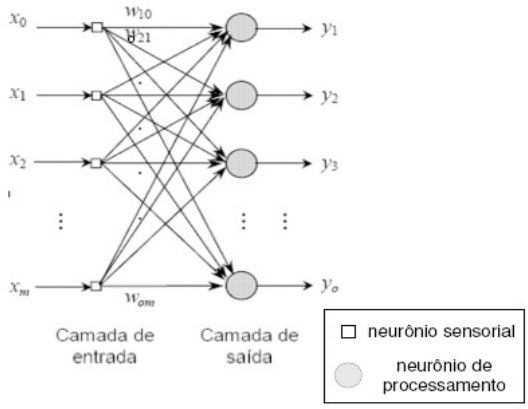

Figura 14 - Rede feedforward de única camada.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 - [6]

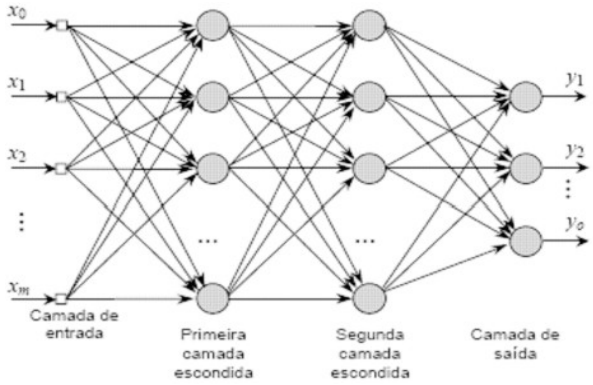

Figura 15 - Rede feedforward de múltiplas camadas.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 - [6]

As redes feedforward de múltiplas camadas, também conhecidas como perceptron de múltiplas camadas (do inglês Multi Layer Perceptron – MLP), são as mais conhecidas e mais utilizadas. Podem ter qualquer número de camadas escondidas. Entretanto, segundo Cybenkon [8], uma rede MLP que contenha uma camada escondida pode aproximar qualquer função matemática contínua e se houverem duas camadas escondidas, a rede pode aproximar qualquer função

contínua ou descontínua. Assim, não são necessárias mais que duas camadas escondidas para resolver um problema.

Redes recorrentes [5], ao contrário das redes feedforward, utilizam a saída do neurônio como entrada de neurônios da mesma camada ou camadas anteriores. Existem várias topologias para esse tipo de rede, porém não é objeto de estudo deste trabalho.

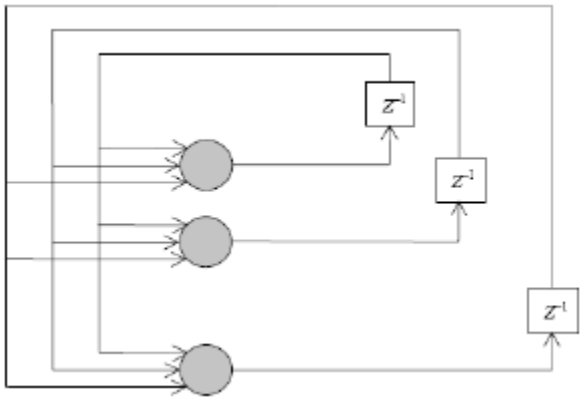

Figura 16 - Exemplo de rede recorrente.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 - [6]

2.7 TREINAMENTO DE REDES PERCEPTRON

2.7.1 Introdução

O treinamento de uma rede neural consiste em treinar todos os neurônio da mesma para tentar aproximar a uma dada função, sendo considerado sua principal funcionalidade [6]. Esse processo é caracterizado por tentar encontrar

os melhores pesos para todos os neurônios podendo ser finalizado quando a rede provê uma boa generalização.

Existem vários algoritmos de treinamento para vários tipos de redes neurais diferenciando apenas em como serão atualizados os pesos. Sendo assim, de forma genérica, o algoritmo de treinamento nada mais é do que a busca pelos pesos corretos.

2.7.2 Tipos de Treinamento

Existem três tipos principais de treinamento [6], sendo eles:

- supervisionado;
- não-supervisionado;
- com reforço.

No aprendizado supervisionado existe o papel do professor, que conhece as respostas. Para o problema de classificação de dias de sol e chuva proposto na introdução da seção 2.4, o professor conhece todos os pares de entrada (temperatura e pressão) e suas respectivas saídas. Ele é responsável em guiar o aprendizado, corrigindo os pesos se estes estiverem errados.

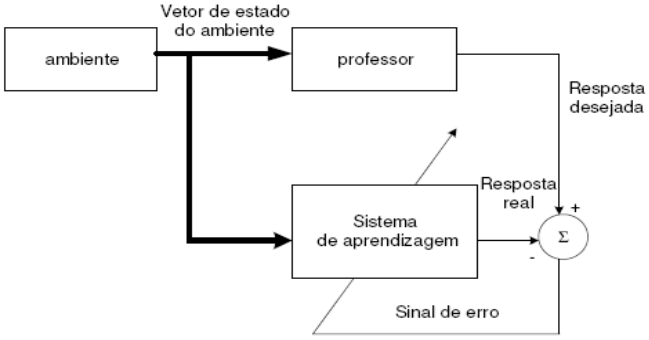

Figura 17 - Aprendizado supervisionado.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 - [6]

Na figura 18, pode-se observar que o ambiente fornece os dados conhecidos (conjunto de treinamento). Cada amostra desse conjunto é apresentada ao professor, que conhece a resposta, e para o sistema que deseja aprender. A resposta do sistema de aprendizado é comparada com a saída do professor e se houver divergência, é lançado um sinal de erro para o sistema de aprendizado que atualiza seus pesos. Isto ocorre até que o erro seja considerado tolerável ou até atingir número máximo de épocas de treinamento [6].

Esse tipo de treinamento é utilizado quando se tem muitos dados de treinamento e se conhece seus valores de saída, porém, existem situações onde não se conhece esses valores, logo não há o papel do professor. Para esse caso é utilizado o aprendizado não supervisionado. O funcionamento deste tipo de aprendizado é realizado através da apresentação dos dados do ambiente e a RNA aprende a disposição dos dados.

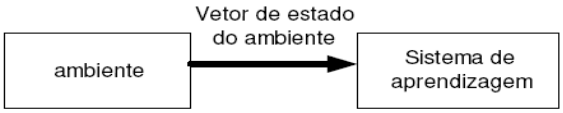

Figura 18 - Aprendizado não supervisionado.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 - [6]

O aprendizado não supervisionado [5] é aplicado principalmente a problemas onde se deseja extrair informações estatística, ainda desconhecidas, relacionado aos dados, sendo útil para problemas de classificação.

Existe um terceiro tipo de aprendizado chamado de aprendizado por reforço que é muitas vezes considerado na literatura como um subtipo do aprendizado supervisionado. Neste esquema não existe o papel do professor, mas sim o papel de um crítico que valida a saída do sistema de aprendizado informando se está boa ou ruim [5].

Segundo [9], se a ação tomada pelo sistema de aprendizado for boa, então o sistema tende a fortalecer está ação, caso contrário, o sistema tende a enfraquecer e buscar novas soluções. Esse tipo de aprendizado é utilizado principalmente em sistemas onde é permitido que hajam erros.

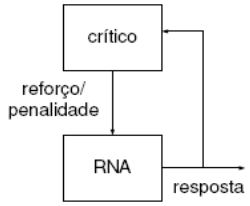

Figura 19 - Aprendizado com reforço.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 - [6]

2.7.3 Treinamento para perceptron de múltiplas camadas

O funcionamento do algoritmo de treinamento de MLPs, Backpropagation, é composto por duas etapas. A primeira, chamada de propagação (forward), é caracterizada pela apresentação dos dados à rede que processa os dados e emite uma saída. A segunda etapa, chamada de

retropropagação (backward), realiza a comparação da saída obtida com a desejada [6]. Se houver erro, ele é retropropagado da camada de saída até a camada de entrada, fazendo com que cada neurônio atualize seus pesos. Esse processo é repetido até que o erro seja menor ou igual ao valor de tolerância ou quando o número de épocas máximo desejado for atingido.

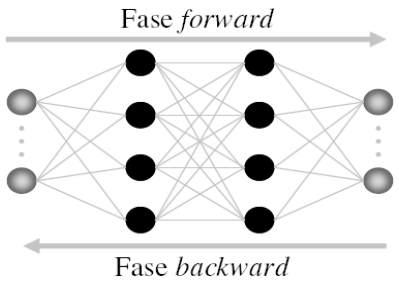

Figura 20 - Fases do algoritmos Backpropagation.
Fonte: Slides de aula da disciplina de RNA – UFLA, 2009 - [6]

Assim, o algoritmo Backpropagation, pode ser descrito resumidamente através do pseudocódigo contido na figura 22.

```
Inicializa pesos (geralmente de forma randômica)
        faça
                Para cada amostra "a" do conjunto de treinamento
                s = saída_da_rede(RNA, a);
                t = saída_do_professos(a);
                erro = t - s;
                Calcula, através da regra delta, os pesos dos neurônios
                das camadas escondidas;
                Calcula, através da regra delta, os pesos dos neurônios
                da camada de entrada;
                Atualiza os pesos da RNA;
        fim da época
até que todos os exemplos estejam sendo classificados corretamente ou
atinja critério de parada;
```

Figura 21 – Algoritmo Backpropagation

O algoritmo Backpropagation utiliza os mesmo parâmetros utilizados no treinamento do perceptron simples. São eles: taxa de aprendizado, bias, momento, número máximo de épocas e erro de tolerância, sendo os dois últimos utilizados como critério de parada do algoritmo.

2.7.4 Definições de arquitetura e parâmetros

Uma das principais questões que surgem durante a implementação de uma RNA é com relação ao número de neurônios nas camadas escondidas. Esse problema geralmente não ocorre para a camada de entrada pois sabemos o número de características que compõem as amostras do conjunto de treinamento. O mesmo não ocorre para a camada de saída pois conhecemos o resultado esperado para o tipo de problema em questão. Assim, o problema está na definição da quantidade de neurônios que irá melhor aproximar a função desejada.

O número de neurônios das camadas escondidas é definido durante a construção da rede, porém, não existe uma forma de saber exatamente quantos neurônios são necessários. A solução para esse problema é empírica, ou seja, testes devem ser feitos treinando redes de várias topologias e comparando suas saídas a fim de obter melhor generalização**.**

O mesmo problema ocorre para os parâmetros de treinamento. Costuma-se utilizar faixas de valores para a taxa de aprendizado e para o momento. Porém não se sabe uma forma de obter os parâmetros que treinaram a RNA mais rapidamente. Da mesma forma, são necessários testes com diferentes combinações de parâmetros a fim de minimizar o tempo de aprendizado.

2.8 TRABALHOS CORRELATOS

Um dos primeiros trabalhos foi de Kimoto et al [11] que utilizou RNA para tentar prever o índice da bolsa de valores de Tóquio. Mais tarde, Mizuno et al [12] utilizou novamente RNA para tentar prever sinais de compra e venda de um papel. Phua et al [13] utilizou RNA juntamente com *algoritmo Genético*[1] para tentar prever a tendência do mercado. Lee [14] utilizou sistemas especialistas para treinar o modelo a reconhecer padrões de *candlesticks*.

O trabalho de Lee [14] apresentou um grande problema no momento de treinamento. O aprendizado foi limitado ao especialista que determinou os padrões de *Candlesticks*, ou seja, nenhum novo padrão é notado.

1 Modelo computacional baseado no comportamento evolutivo biológico.

3. METODOLOGIA

3.1 INTRODUÇÃO

Neste Capítulo são apresentados os métodos e as ferramentas usados para o desenvolvimento do sistema proposto. A figura 22 a seguir, corresponde a um fluxograma que mostra todas as etapas do sistema e permite ter uma visão geral do funcionamento do mesmo.

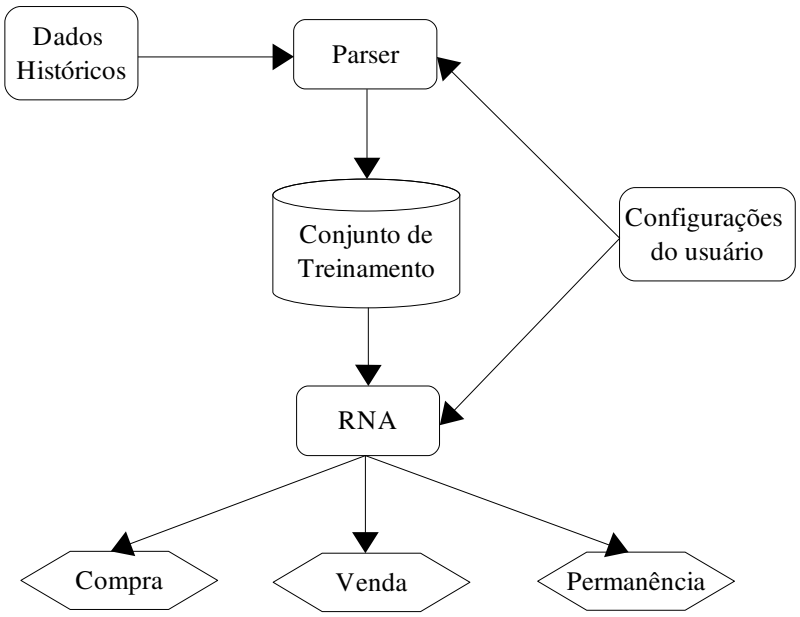

Figura 22 - Fluxograma de funcionamento do sistema

Inicialmente, o parser realiza o processamento das séries históricas de preços, utilizando as configurações do usuário para classificar cada janela de

tempo como padrões de compra, venda ou permanência de uma papel. Ao mesmo tempo, o parser grava em um arquivo texto cada janela analisada e seu respectivo rótulo. Esse arquivo é então usado como conjunto de treinamento para a RNA. Após a etapa de treinamento, a RNA pode generalizar o conhecimento obtido classificando novos padrões que não faziam parte do conjunto de treinamento.

3.2 O PARSER

3.2.1 Processamento

Como citado na introdução, o parser tem a função de processar os arquivos de preços, classificando as janelas de tempo. Para isso, estabeleceu-se que cada janela de tempo deveria conter cinco dias. Essa escolha foi feita devido ao fato de que os padrões conhecidos em gráficos de *candlesticks* possuem tamanho máximo de cinco dias.

O funcionamento do parser consiste um extrair os valores de abertura, máximo, mínimo e fechamento dos arquivos históricos obtidos no site da BOVESPA [17]. Este arquivo possui várias informações, e nesse trabalho somente as informações necessárias para montar os *candles* foram extraídas.

O arquivos históricos disponibilizados gratuitamente pela BOVESPA possuem formato “txt” e contém todas as transações ocorridas ao longo de um ano. A interpretação deste arquivo é realizada segundo as determinações da BOVESPA e as informações úteis para este trabalho são extraídas de acordo com a tabela 1.

Tabela 1 – Interpretação do arquivo de histórico de preços

Informação	Seqüência de Caracteres	Casas decimais
Código da Ação	12 a 23	
Abertura	56 a 66	67 a 68
Máximo	69 a 79	80 a 81
Mínimo	82 a 92	93 a 94
Fechamento	108 a 118	119 a 120

Cada linha do arquivo de histórico de preços corresponde às informações relativas a um dado ativo durante um dia. A coluna "Seqüencia de Caracteres", da tabela 1, informa o intervalo de caracteres da linha, que corresponde a informação desejada. Se, este valor é referente um dado preço, a coluna "Casas decimais" informa a seqüência de caracteres que correspondem as casas decimais para compor o valor do preço.

Assim, cada linha é lida e verifica-se se o seu código corresponde ao desejado pelo usuário. Caso isso ocorra, esse *candle* é então armazenado em memória, segundo a ordem que foi lido. O período desejado para esse processamento é informado pelo usuário.

3.2.2 Rotulação

Após o processamento dos arquivos históricos e armazenamento em memória dos *candles* desejados, é realizada a etapa de rotulação. Essa etapa consiste em processar janelas de tempo de cinco dias a partir dos *candles*

obtidos.

Para cada janela de tempo, calcula-se o índice estocástico utilizando como faixa de permanência os valores entre 15% e 85%, ou seja, valores menores que 15% serão rotulados como compra, valores superiores a 85% serão rotulados como venda e dentro dessa faixa serão rotulados como permanência de estado do papel.

3.2.3 Armazenamento

Após a da fase de rotulamento, cada janela de tempo é gravada em arquivo texto (formato ".txt") seguindo a ordem de disposição dos *candles*. São salvos todos os quatro valores de cada *candle*. Cada padrão contém portanto vinte valores pois, cada janela é formada por cinco dias. Além do *candle*, o rótulo também é gravado. A ordem em que os *candles* e o rótulo são salvos é ilustrada na figura 23.

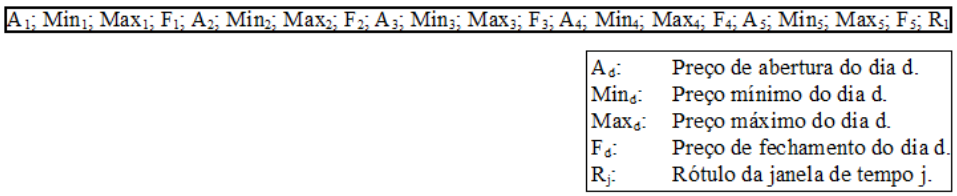

Figura 23 – Representação da janela de tempo rotulada salva em arquivo.

O rótulo é composto por três valores e são representados seguindo as possíveis saídas da RNA[2] sendo ilustradas na tabela 2.

2 As possíveis saídas da RNA serão detalhadas na seção 3.3.1

Tabela 2 – Rótulo para possíveis classificação de janela de tempo.

Classificação	Rótulo
Compra	1;0;0
Venda	0;1;0
Permanência	0;0;1

3.2.4 Normalização

Antes de realizar o processo de armazenamento dos padrões no arquivo texto, é feita a normalização de cada janela de tempo. Isso consiste em realizar uma operação matemática sobre os dados a fim de representá-los utilizando a mesma faixa de valores.

Optou-se por realizar a divisão de todos os preços que compõe os *candles* da janela pelo maior valor. Isso possibilitou transformar os preços em valores reais entre 0 e 1. A normalização dos preços foi realizada para padronizar os valores de entrada da RNA facilitando o aprendizado da mesma.

3.2.5 Funcionamento

O processo de funcionamento completo do parser pode ser descrito pelos seguintes passos:

1 Usuário informa período histórico para que o conjunto de treinamento seja gerado;

2 Usuário informa o conjunto de códigos de papéis que serão processados;

3 O parser, lê todos arquivos de preço e armazena em memória os

candles relevantes;

4 Rotula a janela;

4.1 Obtém a janela de tempo;

4.2 Obtém maior valor de preço da janela;

4.3 Normaliza

4.4 Calcula-se o índice estocástico para a janela;

4.5 Classifica segundo o estocástico;

5 Salva em arquivo a janela normalizada juntamente com seu rótulo segundo a classificação da tabela 2;

6 Repete os passos 4 e 5 até que todas as janelas tenham sido rotuladas.

7 O conjunto de treinamento é gerado.

3.3 REDE MLP

3.3.1 Topologia

A topologia da rede perceptron de múltiplas camadas utilizada neste trabalho possui vinte unidades de entrada, que correspondem aos valores de preços contido na janela de tempo, uma camada escondida de neurônios e uma camada de saída.

As sinapses que ligam as camadas são completas, ou seja, todos os neurônios de uma camada estão ligados com todos os neurônios da próxima camada. A estrutura da RNA pode ser observada na figura 24.

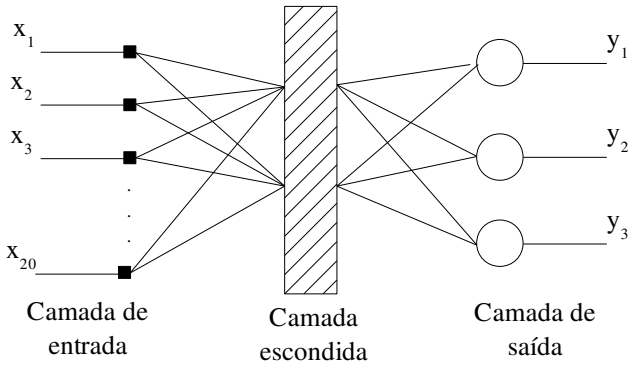

Figura 24 – Estrutura da RNA utilizada.

A camada de saída da RNA possui três neurônios que correspondem aos possíveis rótulos da janela de tempo. O primeiro neurônio (saída y_1), é responsável por detectar padrões de compra, o segundo (saída y_2) detecta padrões de venda e o terceiro (saída y_3), sinaliza permanência do papel.

O número de neurônios da camada escondida é informado pelo usuário antes de realizar o treinamento.

3.3.2 Função de ativação

Foi utilizada a função de ativação sigmoidal para os neurônios das camadas escondida e de saída. A saída linear dos neurônios da camada de saída é portanto pós-processada por uma função para que seja transformada em valores binários. Essa função[3] transforma a saída do neurônio em 1 caso seu valor linear seja maior ou igual a 0.5 ou em 0, caso contrário.

3 Função que determina os rótulos de classificação para a tabela 2

3.3.3 Definição de parâmetros

Como visto anteriormente, alguns parâmetros de treinamento devem ser definidos. Para a maioria deles, o usuário deve informar, antes de realizar o treinamento, as configurações desejadas. São elas: taxa de aprendizado, momento e número máximo de épocas.

3.3.4 Algoritmo de treinamento

O algoritmo utilizado para o treinamento da RNA é o Backpropagation sendo implementando pelo *framework* JOONE[4]. Para cada linha do arquivo do conjunto de treinamento, utiliza-se os vinte primeiros valores (separados por ";") como entrada da RNA. A saída é comparada com a classificação contida no arquivo (os três últimos valores da linha atual) e, em caso de erro, atualiza-se os pesos segundo o algoritmo backpropagation.

Uma época de treinamento é finalizada quando o algoritmo processa todos os padrões. O critério de parada adotado foi o número máximo de épocas, que deve ser passado pelo usuário.

3.3.5 Validação

Utilizando a mesma metodologia para a geração do conjunto de treinamento, foi gerado um conjunto para realizar a validação da RNA. O procedimento de validação segue os seguintes passos:

1 Cria-se um conjunto de validação;

2 Apresenta-se esse conjunto à uma RNA já treinada;

4 Será introduzido na seção 3.3.2.

3 Para cada valor desse conjunto;

3.1 Compara saída da RNA com a contida no conjunto de validação;

3.2 Se são iguais, incrementa contador de acertos;

3.3 Caso contrário, incrementa contador de erros;

4 Calcula porcentagem de acerto;

3.4 TECNOLOGIA

3.4.1 Implementação

Para a implementação do parser e da RNA, utilizou-se a linguagem de programação Java com o paradigma de orientação a objetos. A versão do kit de desenvolvimento java da Sun Mycrosystems [18] (em inglês, Sun Java Developer Kit – Sun JDK) é a 1.6. Utilizou-se também como ambiente de desenvolvimento integrado (em inglês, Integrated Development Environment – IDE) o Netbeans versão 6.5.1 [19]. Utilizou-se também um framework de código aberto para manipulação de RNAs, o JOONE [20].

3.4.2 Framework JOONE

Em inglês, *Java Object Oriented Neural Engine* – JOONE, é um framework para construir, treinar e validar RNA utilizando a tecnologia Java. Escolheu-se sua utilização pela facilidade de implementação, por ser robusto, de código aberto e principalmente por atender todas necessidades provenientes

deste trabalho.

Entre suas principais funcionalidades, podemos citar:

- •Aprendizado supervisionado:
  - •Redes *Feed Forward*;
  - •Redes recursivas;
  - •Algoritmo Backpropagation clássico;
  - •Algoritmo Backpropagation com variações;
- •Aprendizado não supervisionado;
- •Possui ambiente gráfico onde se pode criar, treinar e testar RNAs de forma visual;

Um exemplo[5] interessante retirado da documentação do *framework* é o código a seguir que cria, treina e valida uma RNA utilizando apenas três linhas de código:

```
// Cria uma rede MLP com 3 camadas possuindo dois neurônios nas
// camadas de entrada e escondida, e um neurônio de saída utilizando
// função logística  como ativação
NeuralNet nnet = JooneTools.create_standard(new int[]{2,2,1},
        JooneTools.LOGISTIC);
// Treina a RNA por 5000 épocas ou até que o erro seja menor que
// 0.01. Basta apenas informar o vetor de entrada e o vetor desejado.
double rmse = JooneTools.train(nnet, inputArray, desiredArray, 5000,
        0.01, 0,  null, false);
// Realiza validação da RNA passando um vetor de teste
double[] output = JooneTools.interrogate(nnet, testArray);
```

Figura 25 – Criação, treinamento e validação de uma RNA utilizando *framework* JOONE

5 No anexo B encontra-se o exemplo de implementação uma RNA para resolver o problema da porta lógica OU exclusivo, utilizando o JOONE.

RESULTADOS

4.1 Trading System

O *Trading System* desenvolvido contém o parser para processamento dos dados históricos e toda a parte que envolve RNA. A tela principal deste software pode ser observada na figura 25.

Os passos para gerar o conjunto de treinamento são:

1. Configurar a pasta onde se encontram os arquivos de dados históricos. Para isto, basta clicar no primeiro botão "..." que se encontra na caixa de grupo "Caminhos de Arquivo", ao lado da caixa de edição da configuração de "Histórico de preços";
2. Usuário deve informar suas configurações para o ano inicial e final, sendo estes o período o qual será processado os dados.
3. Usuário também deve informar os nomes dos papéis desejados, separados por vírgula, na caixa de edição "Nomes" que se encontra no grupo "Papel";
4. Clicar no botão "Gerar Conjunto".

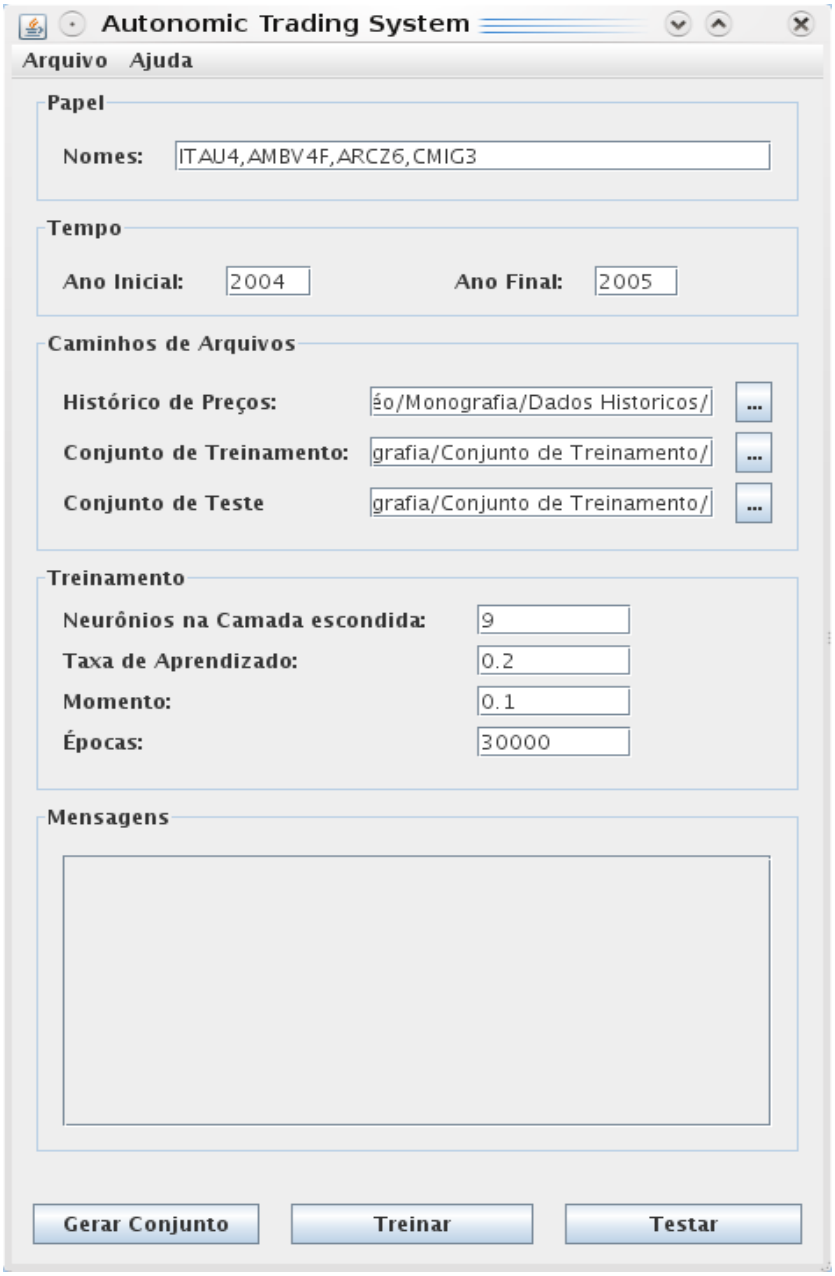

Figura 26 – Tela principal do software desenvolvido.

Os passos para treinar uma RNA são:

1.Configurar a pasta onde será salvo o arquivo de treinamento. Para isto, basta clicar no segundo botão "..." que se encontra na caixa de grupo "Caminhos de Arquivo", ao lado da caixa de edição da configuração de "Conjunto de Treinamento";

2.O usuário deve informar todos os parâmetros contidos no grupo "Treinamento";

3.Clicar no botão "Treinar";

Para se validar a RNA treinada, basta realizar os seguintes passos:

1.Gerar conjunto de validação qualquer;

2.Configurar caminho do arquivo gerado através do botão "..." que se encontra no grupo "Caminhos de Arquivos", ao lado da caixa de edição da configuração de "Conjunto de Teste";

3.Clicar no botão "Testar" e observar a taxa de erro.

4.2 Testes

Para gerar o conjunto de treinamento, foram utilizados os papéis que obtiveram maior volume de negociação no período de 2000 até 2006. Não foram utilizadas negociações muito antigas, pois ao longo do tempo, alguns papéis mudaram de nome ou foram pouco representativos durante algum período. Os ativos utilizados são apresentados na tabela 3.

Tabela 3 – Ações processadas

Ação	Código
AMBEV	AMBV4
ARACRUZ	ARCZ6
CEMIG	CMIG3
ITAUBANCO	ITAU4

Utilizou-se o conjunto de treinamento gerado para treinar diferentes RNAs com números diferentes de neurônios na camada escondida. Além disso, para a mesma topologia de RNA, foram realizados testes para diferentes parâmetros de treinamento. Os resultados obtidos são ilustrados na tabela 4.

Tabela 4 – Resultados para diferentes topologias de RNA e diferentes parâmetros

Neurônios na camada escondida	Taxa de Aprendizado	Momento	Épocas de Treinamento	Erro de Treinamento	Acurácia para conjunto de teste
1	0,2	0,1	100	43,05%	56,00%
1	0,5	0,3	100	45,43%	56,00%
1	0,8	0,5	100	43,72%	56,00%
5	0,2	0,1	100	24,74%	97,00%
5	0,5	0,3	100	24,71%	93,00%
5	0,8	0,5	100	25,96%	93,00%
10	0,2	0,1	100	23,22%	93,00%
10	0,5	0,3	100	23,44%	97,00%
10	0,8	0,5	100	25,18%	93,00%
15	0,2	0,1	100	22,94%	94,00%
15	0,5	0,3	100	23,29%	93,00%

15	0,8	0,5	100	24,03%	92,00%
20	0,2	0,1	100	22,88%	92,00%
20	0,5	0,3	100	22,71%	94,00%
20	0,8	0,5	100	24,21%	94,00%

Utilizou-se o período de 2006 até 2007 para se gerar o conjunto de validação e testar a capacidade de generalização da RNA.

5. CONCLUSÃO E TRABALHOS FUTUROS

Observando os objetivos propostos no inicio do trabalho e os resultados alcançados, conclui-se que o *trading system* desenvolvido atendeu às espectativas, podendo ser utilizado como ferramenta de investimento em bolsas de valores.

Os melhores resultados foram observados quando utilizou-se de dez a vinte neurônios na camada escondida da RNA, taxa de aprendizado igual a 0,2 e 0,1 para o momento. Esta combinação de parâmetro mostrou-se eficiente mesmo para o treinamento com poucas épocas.

Alguns pontos positivos deste trading system são:

- Bons resultados para RNA bem treinadas;
- Análise Técnica é realizada de forma automática, sendo desnecessário realizar cálculos de índices todos os dias, eliminando o trabalho repetitivo;
- A generalização permite que uma RNA, depois de treinada, aponte reversões de tendências para qualquer ativo;
- O investidor não precisa conhecer como funciona a análise técnica para operar o sistema;

Os pontos negativos:

- O usuário deve possuir algum conhecimento de RNAs. Seria interessante se o funcionamento do software fosse transparente ao usuário, ou seja, o próprio software determinasse os melhores parâmetros de treinamento da RNA;

- Muitos testes devem ser realizados para se descobrir melhores parâmetros de treinamento. Assim, se perde muito tempo tentando encontrá-los, logo que o treinamento é um processo de alto custo computacional.

Muitas melhorias podem ser adicionadas a este trabalho a fim de se construir um *trading system* com mais recursos. As principais são:

- Utilizar outros índices da analise técnica. Seria interessante se o investidor pudesse escolher o índice que deseja escolher até mesmo para comparar qual deles classifica melhor os padrões de reversão e permanência de ativo;
- Utilizar outros algoritmos de treinamento e previsão, sendo possível descobrir qual melhor técnica de aprendizado de máquina ou técnica de previsão;
- Realizar periodicamente *download* das cotações de preços diário e informar ao investidor quais decisões devem ser tomadas;
- As configurações dos parâmetros de treinamento da RNA poderiam ser camuflada de tal forma que um usuário leigo em RNA possa operar o sistema;
- Poderiam existir configurações quanto ao perfil dos investidores, fazendo com que o *trading system* seja mais ousado ou mais cuidadoso em suas decisões;
- Desenvolver diferentes interfaces, permitindo que os investidores acessem o sistema via internet ou sistemas móveis;
- Prover informações em tempo real de ativos na bolsa de valores, onde usuário poderia acompanhar comportamento da bolsas de

valores.

- Seria interessante utilizar RNA para catalogar novos padrões nos gráficos de *candlesticks*, contribuindo com a análise técnica.

De forma geral, foi possível concluir que RNA é uma poderosa ferramenta para resolução de problemas de previsão onde estão disponíveis dados históricos, sendo possível obter bons resultados, se utilizada de forma correta.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] NELOGICA, novembro 2008. http://www.nelogica.com

[2] C. A. Debastiani. *Candlestick*. Novatec, 2007.

[3] C. A. Debastiani. *Análise Técnica de Ações – Identificando Oportunidades de compra e venda*. Novatec, 2008.

[4] CONEXÃO DINHEIRO, fevereiro 2009. http://www.conexaodinheiro.com

[5] A. P. Braga, T. B. Lurdemir, and A. C. P. L. F. Carvalho. *Redes Neurais Artificiais: teorias e aplicações*. LTC, 2000.

[6] Lacerda, W. S. *Notas de aula da disciplina de Redes Neurais Artificiais.* 2009

[7] F. Rosenblatt. The perceptron: A probabilistc model for information storage and organization in the brain. *Psychol. Rev.*, 65:386-408, 1958.

[8] G. Cybenkon. Approximation by superpositions of a sigmoid function. *Mathematics of Control, Signals and Systems,* 2:303-314, 1989

[10] R. S. Sutton, A. G. Barto and R. J. Willians. Reinforcement learning is direct adaptive optimal control. In. *Proc. of the American Control Conference*, pp. 2143-2146, 1991.

[11] Kimoto, T., Asakawa, K., Yoda, M., and Takeoka, M. (1990), Stock market prediction system with modular neural network, in Proceedings of the International Joint Conference on Neural Networks, 1-6.

[12] Mizuno, H., Kosaka, M., Yajima, H. and Komoda N. (1998), Application of Neural Network to Technical Analysis of Stock Market Prediction, Studies in Informatic and Control, Vol.7, No.3, pp.111-120.

[13] Phua, P.K.H. Ming, D., Lin, W. (2000), Neural Network With Genetic Algorithms For Stocks Prediction, Fifth Conference of the Association of Asian-

Pacific Operations Research Societies, 5th - 7th July, Singapore.

[14] Chiung-Hon Leon Lee, WenSung Chen, and Alan Liu. An implementation of knowledge based pattern recognition for finicial prediction. *Cybernetics and Intelligent Systems - IEEE Conference*, 1:218–223, 2004.

[15] Simon Haykin. Redes Neurais*: Princípio e prática*. Bookman, 2001.

[16] GRAFBOLSA, maio 2009. http://www.grafbolsa.com/

[17] BOVESPA, outubro 2008. http://www.bovespa.com.br.

[18] SUN MICROSYSTEMS, maio 2009. http://www.sun.com/

[19] NETBEANS, maio 2009, http://www.netbeans.org/

[20] JOONE, maio 2009, www.jooneworld.com/

ANEXO A

SCRIPT DO ALGORÍTMO DE TREINAMENTO DO PERCETRON

A seguir, encontra-se a implementação do algoritmo de treinamento do perceptron utilizando a linguagem do Scilab. A função treina_perceptron recebe o vetor de pesos inicial (w), o bias inicial (b), os dados de treinamento (xin), as saídas esperadas para a entrada (yin), o número máximo de épocas (maxepocas) e erro de tolerância. Retorna o vetor de pesos, o bias e o erro da última época.

Logo após a implementação da função, apresenta-se ao perceptron como dados de treinamento a tabela verdade da porta lógica AND e utiliza a função de treinamento para treinar neurônio a classificar pares de valores binários.

```
function [w,b,erroepoca] = treina_perceptron(w,b,xin,yin,alfa,maxepocas,tol)
        // contador de épocas
        t = 1;
        // Numero de entradas
        Nsize = size(xin);
        // incializa erro com infinito
        E = 9999;
        // enquanto não percorrer todas as epocas e o Erro foi maior que a
        tolerância
        while ((t <= maxepocas) & (E > tol))
                // para cada padrão de entrada
                for i = 1 : Nsize(2)
                        // guarda saída
                        y(1 , i) = yperceptron(xin(:, i), w, b);
                        // guarda erro da saida = desejado - encontrado
                        erroepoca(1, i) = yin(1 , i) - y(1 , i);
                        // atualiza vetor de pesos
                        w = w +(alfa * erroepoca(1, i) * xin(:, i)');
```

```
                    // atualiza o bias
                    b = b + alfa * erroepoca(1, i);
                end;
                // eleva erros ao quadrado para ficarem positivos e soma
                E = sum(erroepoca ^ 2);
                // passa pra próxima época
                t = t + 1;
        end;
endfunction

// padrões de entrada
xin = [0 1 0 1 ;
       0  0 1 1];
// saidas esperadas
yin = [0 0 0 1];
// taxa de aprendizado
alfa = 1.2;
// número máximo de épocas
maxepocas = 10;
// tolerância para o erro
tol = 0.001;
// inicia pesos (costuma-se utilizar valores randômicos)
w = [1.2 0.8];
// inicia bias  (costuma-se utilizar valor randômico)
b = 0.3;
// chama função de treinamento
[w, b, e] = treina_perceptron(w,b,xin,yin,alfa,maxepocas,tol)
```

ANEXO B

EXEMPLO JOONE PARAO PROBLEMA DA PORTA LÓGICA XOR

A classe a seguir realiza o treinamento de RNA para resolver o problema da porta lógica XOR, utilizando o framework de RNA JOONE. Primeira, é necessário fazer o download do engine no JOONE no site [20] do framework e adicionar ao projeto. Para executar o programa basta alterar os dois atributos, inputData e outputFile, com o caminho dos arquivos do conjunto de treinamento e erro, respectivamente e compilar.

```
/*
 * XOR.java
 * Sample class to demostrate the use of the Joone's core engine
 * see the Developer Guide for more details
 *
 * Versão adaptada ao português por Leonardo Nascimento Ferreira
 */

/*
 * JOONE - Java Object Oriented Neural Engine
 * http://joone.sourceforge.net
 */

import java.io.File;
import org.joone.engine.*;
import org.joone.engine.learning.*;
import org.joone.io.*;
import org.joone.net.NeuralNet;

public class XOR implements NeuralNetListener {
   // path do arquivo contendo o conjunto de treinamento
   private static String inputData = "/media/sda1/Instaladores/" +
         "Framework Java/JOONE/joone-engine-2.0.0RC1/samples/engine/xor/xor.txt";
   // path do arquivo de saida contendo evolução do erro de treinamento
   private static String outputFile = "/media/sda1/Instaladores/" +
        "Framework Java/JOONE/joone-engine-2.0.0RC1/samples/engine/xor/error.txt";
```

```
/**
 * Construtor da classe
 */
public XOR() {
}

/**
 * Método principal
 * @param args os argumentos de linha de comando
 */

public static void main(String args[]) {
   // cria objeto do tipo XOR
   XOR xor = new XOR();
   // cria RNA passando path do conjunto de treinamento e path do arquivo
   // de saída, que irá conter a evolução do erro
   xor.Go(inputData, outputFile);
}

/**
 * Metodo que cria e treina RNA
 * @param inputFile
 * @param outputFile
 */
public void Go(String inputFile, String outputFile) {
   // Inicialmente, cria três camadas
   LinearLayer input = new LinearLayer();
   SigmoidLayer hidden = new SigmoidLayer();
   SigmoidLayer output = new SigmoidLayer();
   // da nome às camadas
   input.setLayerName("input");
   hidden.setLayerName("hidden");
   output.setLayerName("output");
   // configura o número de neurônios que cada camada irá possuir
   input.setRows(2);
   hidden.setRows(3);
   output.setRows(1);
   // Cria sinapse que irá ligar camada de entrada com camada escondida
   FullSynapse synapse_IH = new FullSynapse();
   // Cria sinapse que irá ligar camada escondida com camada de saída
   FullSynapse synapse_HO = new FullSynapse();
```

```
// da nome às sinapses
synapse_IH.setName("IH");
synapse_HO.setName("HO");
// conecta as camadas de entrada e escondida
input.addOutputSynapse(synapse_IH);
hidden.addInputSynapse(synapse_IH);
// conecta a camada escondida com a camada de saída
hidden.addOutputSynapse(synapse_HO);
output.addInputSynapse(synapse_HO);
// Cria objeto de leitura de arquivo
FileInputSynapse inputStream = new FileInputSynapse();
// as duas primeiras colunas do arquivo irão conter os padrões
inputStream.setAdvancedColumnSelector("1,2");
// associa ao path do arquivo
inputStream.setInputFile(new File(inputFile));
input.addInputSynapse(inputStream);
// cria objeto que que representa o professor
TeachingSynapse trainer = new TeachingSynapse();
// Cria objeto de leitura de arquivo
FileInputSynapse samples = new FileInputSynapse();
samples.setInputFile(new File(inputFile));
// a saída desejada dos padrões estão na coluna 3
samples.setAdvancedColumnSelector("3");
// informa o professor que as saídas desejadas estão na terceira coluna
trainer.setDesired(samples);
// Cria objeto que irá conter o erro
FileOutputSynapse error = new FileOutputSynapse();
error.setFileName(outputFile);
// informa ao professor qual será o arquivo de erro
trainer.addResultSynapse(error);
// conecta o professor à camada de saída da RNA
output.addOutputSynapse(trainer);
// Cria RNA
NeuralNet nnet = new NeuralNet();
// insere camada de entrada
nnet.addLayer(input, NeuralNet.INPUT_LAYER);
// insere camada escondida
nnet.addLayer(hidden, NeuralNet.HIDDEN_LAYER);
// insere camada de saída
nnet.addLayer(output, NeuralNet.OUTPUT_LAYER);
// indica à RNA seu professor
nnet.setTeacher(trainer);
```

```
    // cria monitor que irá configurar parâmetros de aprendizado
    Monitor monitor = nnet.getMonitor();
    // configura taxa de aprendizado
    monitor.setLearningRate(0.8);
    // configura momento
    monitor.setMomentum(0.3);
    // A aplicação registra ela mesma como listener para receber
    // notificações da RNA
    monitor.addNeuralNetListener(this);
    // define o número de padrões (linhas) que se encontram no conjunto de
    // treinamento
    monitor.setTrainingPatterns(4);
    // defini número de épocas
    monitor.setTotCicles(2000);
    // sinaliza que a RNA irá ser treinada
    monitor.setLearning(true);
    // inicia treinamento
    nnet.go();
}

/**
 * Método executado quando treinamento é finalizado
 * @param e
 */
public void netStopped(NeuralNetEvent e) {
    // informa que a rede finalizou treinamento
    System.out.println("Treinamento finalizado!");
}

/**
 * Método executado quando uma época termina
 * @param e
 */
public void cicleTerminated(NeuralNetEvent e) {

}

/**
 * Método executado quando inicia treinamento
 * @param e
 */
 public void netStarted(NeuralNetEvent e) {
```

```
        // informa usuário sobre inicio do treinamento
        System.out.println("Iniciando treinamento...");
    }

    /**
     * Método executado quando o erro de treinamento é alterado
     * @param e
     */
    public void errorChanged(NeuralNetEvent e) {
        // obtém monitor da RNA
        Monitor mon = (Monitor)e.getSource();
        // a cada 200 épocas
        if (mon.getCurrentCicle() % 200 == 0)
            // informa ao usuário o número de épocas restantes para o fim do
            // treinamento e erro
            System.out.println("Epocas para o fim: " + mon.getCurrentCicle() +
                    " - Erro de treinamento = " + mon.getGlobalError());
    }

    /**
     * @param e
     * @param error
     */
    public void netStoppedError(NeuralNetEvent e,String error) {

    }
}
```

O arquivo do conjunto de treinamento possui extensão “.txt” e deve conter o seguinte texto:

```
0;0;0
1;0;1
0;1;1
1;1;0
```