Departamento de Estatística, Matemática Aplicada e Computação - Unesp Rio Claro

Apoio à tomada de decisões no Mercado Financeiro utilizando Redes Neurais

Daniel Carlos Guimarães Pedronette
Autor

Prof. Dr. Ivan Rizzo Guilherme
Orientador

Ciências da Computação Noturno

Índice

Índice de Figuras

Capítulo I: Introdução

A capacidade de Redes Neurais Artificiais em mapear sistemas complexos, sem se ter a necessidade de conhecer eventuais modelos matemáticos que descrevem o seu comportamento, têm as tornado uma ferramenta atrativa que pode ser aplicada em vários processos relacionados ao comportamento de mercados financeiros.

Diversas pesquisas vêm sendo realizadas visando a predição da movimentação do mercado de ativos financeiros, sendo boa parte deles utilizando redes neurais para efetuar tal predição. Modelos baseados na teoria do caos partem do princípio que existe uma componente determinística no movimento dos preços dos ativos financeiros e que não seguem um processo estocástico (aleatório), segundo a hipótese de que o comportamento dos preços de uma ação pode ser originado a partir de um processo determinístico complexo[2].

Foram atingidos vários sucessos través de sistemas híbridos composto de redes neurais e sistema baseado em regras, apesar do grande número de horas gasto para a construção do modelo. Todavia, a escolha das ferramentas, modelos de redes neurais, definição dos dados de entrada, abordagem adequadas, testes empíricos e sucessivos fazem parte do processo de desenvolvimento de tais modelos de solução.

O presente trabalho discorre sobre a utilização de Redes Neurais, utilizando o pacote Joone, na previsão de tendências de um determinado ativo financeiro na bolsa de valores de São Paulo - Bovespa.

No decorrer da pesquisa bibliográfica realizada foi possível reconhecer diversos tipos de aplicações das Redes Neurais no apoio a tomada de decisões no mercado de ações. Embora todas elas convirjam em utilizar séries históricas para auxiliar decisões futuras elas diferem-se bastante em como atacar essa problemática. Certas aplicações, como em a apresentada em [4], focam-se em decidir qual a melhor ação para investir-se dado um determinado momento. Outras, como a abordada em [3], propõe-se a identificar formas nos gráficos formadas pelas cotações para distinguir bons momentos de compra ou venda. As abordagens apresentadas em [1,2] objetivam prever valores futuros de cotações das ações.

Para o presente trabalho, como será mais amplamente abordado no Capítulo III, a problemática proposta foi essencialmente identificar no momento atual tendências de alta ou queda, e baseados em algumas regras emitir indicativos de compra ou venda para o investidor.

Os dados de entrada escolhidos assemelham-se parcialmente às entradas utilizadas em [1], já que ambos baseiam-se em dados relacionados às cotações atuais, anteriores e volume de negociações. Todavia, a abordagem presentemente desenvolvida inclui indicadores estatísticos de Análise Técnica visando fornecer entradas mais elaboradas e conclusivas para a Rede Neural.

Numa análise focada no tempo de previsão, verifica-se que o modelo apresentado em [1] possui processo realimentação da rede neural com seus próprios dados de saída, procurando assim produzir previsões para um período

mais longo, como dez unidades de tempo (nesse caso dias). Já a abordagem apresentada em [2], assim como na solução proposta nesse trabalho, apresenta previsões para apenas a próxima unidade de tempo.

Verifica-se, entretanto, no trabalho proposto algumas características inovadoras em relação à bibliografia. A grande maioria dos artigos produzidos nessa área dedicam-se à pesquisa de aplicações em previsões para dias, semanas ou meses enquanto esse trabalho propões a fazer uma análise intra-day. Pode-se verificar também que os trabalhos analisados geralmente propõe-se a fazer previsões do valor da ação, como em [1,2], enquanto o trabalho proposto foca-se em identificar tendências, já que isso é o bastante para orientar e auxiliar o investidor.

Capítulo II: Mercado de Capitais

§ Introdução: Ações e Abertura de Capital

Dado um determinado um grupo de investidores, quando é decidido por estes formar uma nova empresa é necessário decidir o valor do capital social da empresa – valor monetário investido. Também é necessário estabelecer o número de ações em que este valor será dividido. Assim cada ação representa uma parte do capital social da empresa em questão. O número de ações recebido por cada investidor - ou acionista - depende do valor que ele investiu. Ou seja, ao comprar ações de uma empresa, os acionistas passam a possuir "fatias" desta empresa. Os acionistas - em geral, milhares de indivíduos e instituições - possuem parte do patrimônio desta companhia, ou seja, uma fração do todo. As empresas emitem dois tipos básicos de ações: ordinárias e preferenciais - as primeiras, com direito a voto e as demais, com prioridade no recebimento de dividendos.

Quando as empresas abrem capital, elas transferem aos investidores parte do seu controle acionário (caso o aplicador tenha adquirido ações ordinárias, do contrário, ele só terá preferência na distribuição de proventos, não participando da tomada de decisões). Em troca, as companhias abertas recebem dinheiro para seus investimentos e se financiam, podendo, então, expandir seus negócios.

Ao comprar ações, o investidor espera receber dividendos ou juros sobre capital próprio, ou seja, parte dos lucros da empresa. Ele espera também que o preço das ações se valorize, fazendo sua aplicação auferir lucros. Porém, como qualquer investimento, ao comprar ações, o investidor não tem garantia de performance. Isso significa que, da mesma forma que o papel pode valorizar-se, ele também pode depreciar-se.

O risco que os investidores assumem quando eles compram ações são os de que a empresa na qual estão investindo não tenha bons resultados financeiros ou que os preços das ações sofram desvalorização. No pior dos casos, é possível perder todo o investimento - mas não mais do que isto. Os acionistas não são responsáveis pelas dívidas da empresa. No longo prazo a aplicação no mercado de renda variável tem produzido bons resultados históricos, se comparado com os de outras alternativas de investimento.

§ Bovespa

A Bolsa de Valores de São Paulo - BOVESPA foi fundada em 23 de agosto de 1890, atuando no Mercado de Capitais da economia brasileira. Até meados da década de 60, a BOVESPA e as demais bolsas brasileiras eram entidades oficiais corporativas, vinculadas às secretarias de finanças dos governos estaduais e compostas por corretores nomeados pelo poder público.

Com as reformas do sistema financeiro nacional e do mercado de capitais implementadas em 1965/66, as bolsas assumiram a característica institucional que

mantêm até hoje, transformando-se em associações civis sem fins lucrativos, com autonomia administrativa, financeira e patrimonial. A antiga figura individual do corretor de fundos públicos foi substituída pela da sociedade corretora, empresa constituída sob a forma de sociedade por ações nominativas ou por cotas de responsabilidade limitada. A Bolsa de Valores de São Paulo é uma entidade auto-reguladora que opera sob a supervisão da Comissão de Valores Mobiliários (CVM).

Desde a década de 60, tem sido constante o desenvolvimento da BOVESPA, seja no campo tecnológico, seja no plano da qualidade dos serviços prestados aos investidores, aos intermediários do mercado e às companhias abertas.

Na BOVESPA, são regularmente negociadas ações de companhias abertas, opções sobre ações, direitos e recibos de subscrição, bônus de subscrição e quotas de fundos, debêntures e notas promissórias. Além disso, também são negociados na BOVESPA os BDRs ('Brazilian Depository Receipts'), que são certificados representativos de valores mobiliários de emissão de companhia aberta ou assemelhada com sede no exterior, emitidos por instituição depositária no Brasil. Podem ainda ser negociados na BOVESPA certificados de depósitos de ações lançados por empresas sediadas nos países que integram o Acordo do MERCOSUL.

§ Análises de Mercado

Para quem investe no mercado de ações, há duas técnicas distintas de análise de papéis visando a identificação de bons momentos de Compra e Venda: a Análise Técnica e a Análise Fundamentalista. Enquanto a Análise Técnica utiliza recursos estatísticos e parte do princípio de que se pode prever a tendência de uma ação a partir de seu comportamento passado, a Análise Fundamentalista se baseia em estudo de informações fornecidas pela empresa, pelo setor em que ela atua e por indicadores macroeconômicos.

Essas duas técnicas são tão distintas que existe no mercado de capitais verdadeira rivalidade entre grafistas (os partidários da Análise Técnicas) e fundamentalistas (os partidários da Análise Fundamentalista).

§ Análise Técnica

A Análise Técnica defende que todos os fatos (econômicos, políticos, psicológicos e mesmo fundamentalistas) condicionam os preços das ações e que o preço de uma ação no Mercado em um dado momento é ditado antes pelas exigências da oferta e procura, que pelo seu valor intrínseco. Preços desta forma,

seriam apenas o reflexo das mudanças do ritmo da oferta e procura. Se a procura é maior que oferta, o preço sobe; se o inverso ocorre, o preço desce.

O segundo princípio em que a Análise Técnica baseia-se é que preços movem-se em tendências e tendências persistem. O ritmo da oferta e procura coloca uma tendência em movimento. Uma vez em movimento a tendência persiste, até que acabe. Desta forma, preços de Mercado movem-se em tendências. Primeiro, os preços movem-se em uma direção, criando uma tendência. A tendência persiste até que o movimento dos preços diminui, e emite avisos antes de finalmente reverter e começar o movimento na direção oposta. Neste ponto tem início uma nova tendência.

O terceiro e último dos princípios básicos é baseado no fato de que os movimentos do Mercado são repetitivos. Certos padrões repetem-se ciclicamente nos gráficos. Estes padrões têm significados e podem ser interpretados em termos de prováveis movimentos futuros de preços. A natureza humana é tal que tende a reagir similarmente á situações padrões. Como regra, pessoas agem da mesma forma que já agiram no passado. Desde que o Mercado de Ações é um reflexo das ações das pessoas, o Analista Técnico analisa situações recorrentes com finalidade de antecipar-se a altas e baixas do Mercado.

Considerando os princípios mostrados acima, a Análise Técnica pode ser definida como o estudo de ações individuais e do Mercado com base na oferta e procura. Analista Técnicos registram em Gráficos as atividades de preços e volumes e deduzem de sua história gráfica as prováveis futuras tendências dos preços.

A combinação dos indicadores utilizados na Análise Técnica é uma poderosa ferramenta que, com otimizações e ponderações apropriadas, complementadas com um estudo estatístico, permitem calcular a probabilidade de um movimento futuro dos preços num dada direção.

§ Alguns Indicadores Estatísticos

Momento

Este indicador mede a aceleração ou desaceleração dos preços. Obtém-se o momentum simplesmente subtraindo-se os preços de fechamento entre dois períodos determinados, por exemplo, cinco dias. Supõe-se que o momentum antecipa o comportamento dos preços em fases de mudança de mercado. Quando o mercado atinge um pico, o momentum sobe repentinamente e então cai. Similarmente, quando o mercado atinge um fundo, o momentum cai repentinamente e então sobe.

O valor do Momento varia entre valores positivos e negativos, tentando definir se o ativo se encontra num estado "overbought" ou "oversold". Este indicador é calculado numa base absoluta em pontos e é designado por Momento simples.

Volume

Existem várias maneiras de interpretar as mudanças dos volumes das ações. Uma das mais correntes refere-se ao preço de uma ação em alta, relacionado com o volume também em alta, mostrando claramente que o mercado está em ascensão. Pelo contrário, os poucos volumes de uma ação indicam um mercado em queda. Os volumes das ações permite-nos analisar o interesse do mercado por um determinado título num determinado momento. Deste modo, um fraco volume de um determinado título significará um período de indecisão pelo mercado. Encontramos geralmente estes períodos durante as consolidações, período este, durante o qual as cotações flutuam à preços pouco habituais.

IFR

Índice de Força Relativa (IFR) ou Relative Strength Index (RSI). Ele foi introduzido por J. Welles Wilder num artigo publicado em junho de 1978 e a descrição detalhada dos métodos de cálculo e interpretação do RSI podem ser encontrados no seu livro "New Concepts in Technical Systems" (1978).

O nome Índice de Força Relativa na verdade não compara a força relativa entre dois ativos diferentes mas sim a força interna relativa entre dois momentos de único ativo. O nome mais apropriado para o indicador seria "Índice de Força Interna". O IFR procura medir a evolução da relação de forças entre compradores e vendedores ao longo do tempo, tendo sido desenvolvido com a idéia de aperfeiçoar o "Momento"; seus valores oscilam entre zero (dominação total dos vendedores) e cem (dominação total dos compradores). Seu acompanhamento muitas vezes possibilita observar o enfraquecimento de uma tendência, rompimentos, suporte e resistência antes de se tornarem aparentes no gráfico de barras.

A metodologia de cálculo utilizada é bastante simples:

RSI = 100 - [100 / (1 + U/D)]

onde:

U: média das variações de preço para cima (altas)

D: média das variações de preço para baixo (baixas)

SMI

Stochastic Momentum Index. Consiste numa variável estatística que se baseia na posição de uma determinada cotação de fecho em comparação com os máximos e mínimos valores de compra e venda dessa mesma cotação num período de tempo determinado (normalmente 5 sessões). Baseia-se na teoria de que a medida que os preços sobem, os fechamentos tem a tendencia de posicionarem-se mais próximos das altas do período.

Similarmente, se os preços descem, os fechamentos tendem a se aproximarem das baixas. O cálculo do indicador é bastante simples:

SMI = 100 x (Cotação - Mínimo) / (Máximo - Mínimo).

Capítulo III: Apresentação do Problema

Pode-se dizer que o desejo de predizer o futuro é quase que inerente à condição humana, sendo por muitas vezes o motor que impulsiona e motiva o desenvolvimento do conhecimento nas mais distintas áreas. Não poderia deixar de ser assim na área financeira e principalmente no mercado de títulos e ações que movimenta cifras tão grandiosas.

A possibilidade de prever ou sequer aproximar o comportamento do mercado de ações vem motivando matemáticos e estudiosos já há bastante tempo. Qualquer investidor, em posse de um mecanismo desses poderia reduzir erros de investimentos e maximizar imensamente seus lucros. Para isso não seria necessário fazer previsões exatas do preço de cada ação, bastaria que se pudesse predizer os movimentos de subida e descida do preço das ações. Assim bastaria adquirir as ações antes de tendências de alta e dispor das mesas antes de movimentos de queda. Qualquer mecanismo ou algoritmo capaz de realizar ou mesmo aproximar tais resultados seria de extremo valor para os investidores.

Tendo em vista que o valor de negociação das ações é um dado vinculado a diversas variáveis, como patrimônio (monetário, humano, etc), lucro, receita líquida da empresa e ainda influenciado por macro variáveis econômicas e não há um modelo matemático direto que relacione esses aspectos seria impossível construir um algoritmo/função que fosse capaz de identificar as tendências de preços acima citadas.

Somente métodos dinâmicos e adaptativos, capazes de determinar padrões e relacionamentos nos dados globais do mercado, sem contudo que haja uma relação direta entre elas poderiam ser úteis nessas situações. Métodos assim podem mostrar-se importantes na identificação de oportunidades pois, num mercado rápido e extremamente interligado, os preços de um ativo financeiro refletem instantaneamente todas as informações relevantes do mercado. Essas informações por sua vez implicam em decisões por parte dos agentes econômicos no que diz respeito à compra ou venda de títulos. Assim a agilidade da tomada de decisão é um fator decisivo para que os investidores obtenham competitividade no mercado.

De forma geral, a problemática giraria em torno do desenvolvimento de métodos e algoritmos capazes de aproximar com relativa precisão os futuros movimentos das cotações de uma ação. Esses métodos ainda poderiam ter como base de dados o comportamento anterior dessas ações. Isso possibilitaria ao investidor realizar suas transações alicerçado numa ferramenta computacional e minimizando seus erros. Além disso a ferramenta deverá oferecer seus resultados em tempo hábil para que o investidor esteja sempre atualizado e competitivo. No presente trabalho, estuda-se a utilização de Redes Neurais para construção da proposta ferramenta computacional.

Capítulo IV: Redes Neurais

§ Redes Neurais: Definição

Redes neurais artificiais são um conceito da computação que visa trabalhar no processamento de dados de maneira semelhante ao cérebro humano. O cérebro é tido como um processador altamente complexo e que realiza processamentos de maneira paralela. Para isso, ele organiza sua estrutura, ou seja, os neurônios, de forma que eles realizem o processamento necessário. Isso é feito numa velocidade extremamente alta e não existe qualquer computador no mundo capaz de realizar o que o cérebro humano faz.

Nas redes neurais artificiais, a idéia é realizar o processamento de informações tendo como princípio a organização de neurônios do cérebro. Como o cérebro humano é capaz de aprender e tomar decisões baseadas na aprendizagem, as redes neurais artificiais devem fazer o mesmo. Assim, uma rede neural pode ser interpretada como um esquema de processamento capaz de armazenar conhecimento baseado em aprendizagem (experiência) e disponibilizar este conhecimento para a aplicação em questão.

§ Funcionamento das redes neurais artificiais

As redes neurais artificiais são criadas a partir de algoritmos projetados para uma determinada finalidade. É impossível criar um algoritmo desse sem ter conhecimento de modelos matemáticos que simulem o processo de aprendizado do cérebro humano. Por este ser um artigo de introdução a este assunto, abordaremos uma explicação conceitual eliminando ao máximo os princípios matemáticos naturalmente relacionados.

Basicamente, uma rede neural se assemelha ao cérebro em dois pontos: o conhecimento é obtido através de etapas de aprendizagem e pesos sinápticos são usados para armazenar o conhecimento. Uma sinapse é o nome dado à conexão existente entre neurônios. Nas conexões são atribuídos valores, que são chamados de pesos sinápticos. Isso deixa claro que as redes neurais artificiais têm em sua constituição uma série de neurônios artificiais (ou virtuais) que serão conectados entre si, formando uma rede de elementos de processamento.

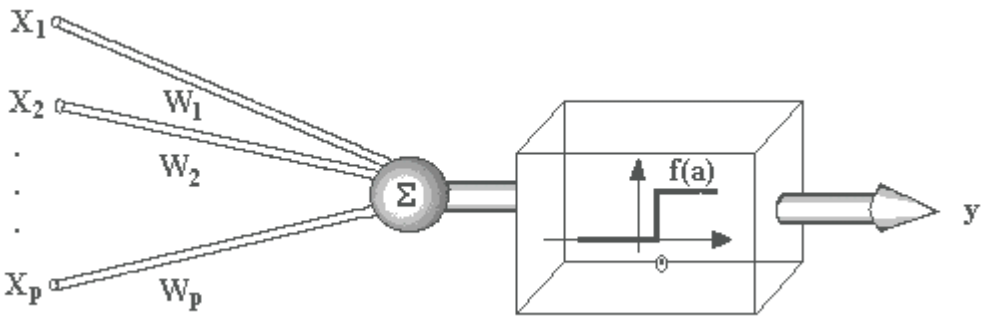

Figura 1. Representação de Neurônio e suas Sinapses

Tendo uma rede neural montada, uma série de valores podem ser aplicados sobre um neurônio, sendo que este está conectado a outros pela rede. Estes valores (ou entradas) são multiplicados no neurônio pelo valor do peso de sua sinapse. Então, esses valores são somados. Se esta soma ultrapassar um valor limite estabelecido, um sinal é propagado pela saída (axônio) deste neurônio. Em seguida, essa mesma etapa se realiza com os demais neurônios da rede. Isso quer dizer que os neurônios vão enfrentar algum tipo de ativação, dependendo das entradas e dos pesos sinápticos.

Existem várias formas de se desenvolver uma rede neural. Ela deve ser montada de acordo com o(s) problema(s) a ser(em) resolvido(s). Em sua arquitetura são determinados o número de camadas usadas (as camadas são formadas por neurônios), a quantidade de neurônios em cada camada, o tipo de sinapse utilizado, etc.

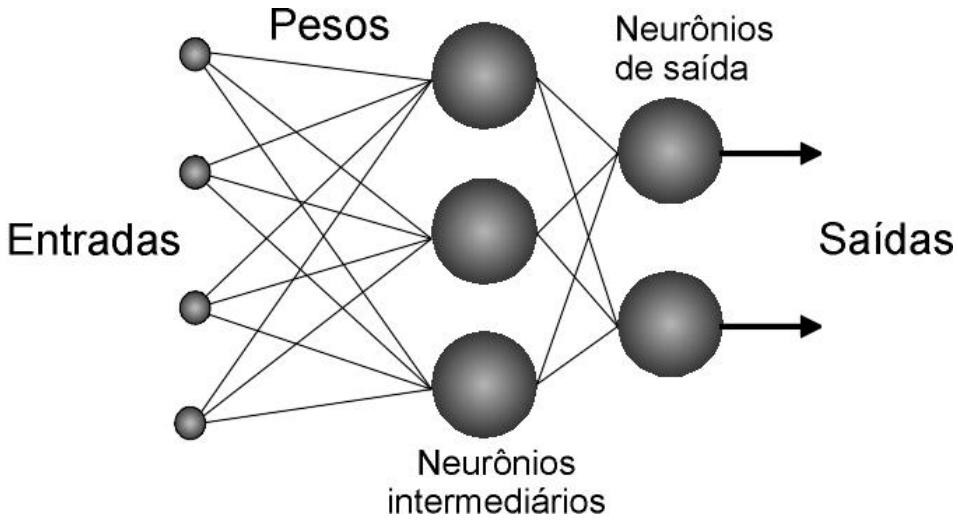

Figura 2. Rede Neural de 3 camadas

§ O aprendizado

O processo de aprendizagem das redes neurais é realizado quando ocorrem várias modificações significantes nas sinapses dos neurônios. Essas mudanças ocorrem de acordo com a ativação dos neurônios. Se determinadas conexões são mais usadas, estas são reforçadas enquanto que as demais são enfraquecidas. É por isso que quando uma rede neural artificial é implantada para uma determinada aplicação, é necessário um tempo para que esta seja treinada.

Existem, basicamente, 3 tipos de aprendizado nas redes neurais artificiais:

Supervisionado: neste tipo, a rede neural recebe um conjunto de entradas padronizados e seus correspondentes padrões de saída, onde ocorrem ajustes nos pesos sinápticos até que o erro entre os padrões de saída gerados pela rede tenham um valor desejado;

Não-supervisionado: neste tipo, a rede neural trabalha os dados de forma a determinar algumas propriedades dos conjunto de dados. A partir destas propriedades é que o aprendizado é constituído;

Híbrido: neste tipo ocorre uma "mistura" dos tipos supervisionado e não-supervisionado. Assim, uma camada pode trabalhar com um tipo enquanto outra camada trabalha com o outro tipo.

§ Redes Neurais Multiníveis

As redes multiníveis, também conhecidas como Multilayer Perceptron (MLP) permitem o aprendizado das RNA baseadas no modelo do Perceptron com qualquer número de níveis. O primeiro algoritmo deste categoria é o algoritmo back propagation.

Uma rede neural artificial MLP usando uma topologia feedforward, pode ser treinada pelo algoritmo back propagation, onde o aprendizado é supervisionado. O Back Propagation é um algoritmo de propósito geral. Este algoritmo é muito poderoso mas é caro em termos de requisitos computacionais para treinamento.

Uma rede back propagation utiliza uma única camada escondida de elementos de processamento que pode modelar qualquer função contínua sem qualquer grau de precisão. Existem diversas variações deste algoritmo. O back propagation é baseado em uma forma relativamente simples de otimização conhecida por gradiente descendente.

As RNA back propagation podem ser usadas para classificação, modelagem e previsão de séries temporais. Para problemas de classificação, os atributos de entrada são mapeados para as categorias de classificação desejadas.

Para construir modelos ou funções de aproximação, os atributos de entrada são mapeados para a função de saída.

Alguns parâmetros são utilizados para controlar o processo de treinamento de uma rede back propagation, dentre os parâmetros existentes, cita-se: taxa de aprendizado e quantidade de movimento. A taxa de aprendizado é usada para especificar se a RNA fará ajustes após cada tentativa de aprendizado. A quantidade de movimento é utilizado para controlar possíveis oscilações nos pesos, as quais poderiam ser causadas por assinalamentos alternados de erros de sinais.

O algoritmo back propagation recebe como parâmetros de entrada para a realização do treinamento da RNA um conjunto de valores de entrada e um conjunto de valores representando a(s) saída(s) desejada(s) por utilizar o tipo supervisionado de aprendizado. O funcionamento básico do algoritmo do back propagation consiste em propagar o conjunto de valores de entrada através da rede até que alcance a unidade de saída. As saídas calculadas pela RNA são subtraídas da(s) saída(s) desejada(s) e um sinal de erro é propagado. Este sinal de erro é a base do algoritmo back propagation porque os erros são propagados através da RNA. Com a propagação dos erros, é calculado a contribuição de cada unidade de processamento escondida no erro e são ajustados os valores para produzir a saída correta. As conexões de peso são ajustadas e a RNA obteve apenas um "aprendizado" como experiência. O erro, para as unidades de saída, é calculado como a diferença entre a saída correta e a atual. Este processo de cálculo dos erros e ajustes das conexões de peso é realizado até que a RNA consiga calcular o valor de saída próximo ao valor da saída desejada.

§ Aplicações para redes neurais

As redes neurais artificiais podem ser aplicadas para resolver uma grande quantidade de problemas. Um bom exemplo de aplicação são softwares de reconhecimento de voz, que precisam aprender a conhecer a voz de determinadas pessoas. Redes neurais também são usados em rôbos que desarmam bombas. Se você já usou um scanner para retirar um texto de um jornal, por exemplo, saiba que o software de OCR, que é responsável por isso, precisa aprender a reconhecer caracteres da imagem. Logo, ele certamente possui algoritmos de rede neural. Existem inclusive softwares que aprendem a identificar SPAMs em e-mails e apagá-los (e conseguem uma margem aceitável de acertos). Mas no geral as redes neurais são usadas principalmente em aplicações mais complexas. Finalmente, a aplicação mais difundida das redes neurais é na previsão de comportamentos no mercado financeiro, que será a aplicação utilizada neste trabalho.

Capítulo V: Joone

§ Visão Geral

O Joone (Java Oriented Object Neural Engine) consiste num framework escrito em Java que tem por objetivo disponibilizar ferramentas que possibilitem a construção de aplicações de Inteligência Artificial baseada em Redes Neurais e modelo de aprendizado backpropagation.

Trata-se de um projeto open source licenciado pela LGPL (Lesser GNU Public License), possibilitando a livre utilização do código qualquer espécie de ônus. O site oficial do projeto é http://www.joone.org.

O Joone foi construído segundo uma arquitetura modular baseada em componentes adaptáveis que podem inclusive ser estendidos para criar novos algoritmos de aprendizado e Redes Neurais. Todos os componentes apresentam características como persistência, multithreading, serialização permitindo alta escabilidade e extensibilidade para o pacote como um todo. Além disso os componentes podem ser acoplados num outra aplicação segundo se apresentarem as necessidades.

§ Principais Componentes

No pacote Joone, toda Rede Neural tem como principais componentes os objetos denominados Layers. Os Layers representam um conjunto de neurônios de comportamento semelhante. A interconexão entre os Layers é realizada através de objetos denominados Synapses. Os tipos e características dos Layers e Synapses determinam as principais características da Rede Neural.

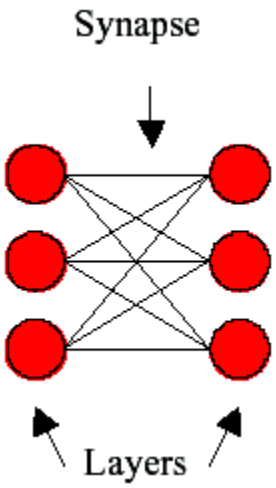

Figura 3. Componentes do Pacote Joone

Os Layers trabalham de maneira independente, recebendo dados da Synapse de entrada, processando via função de transferência e enviando para a Synapse de saída. Características importantes são definidas nos Layers, como o número de neurônios e a função de transferência, determinada pelo tipo: Linear Layer, Sigmoid Layer, Tahn Layer, etc. As Synapses por sua vez, diferem-se principalmente quanto a geometria de conexão entre os neurônios.

Além desses dois tipos principais de objetos o Joone permite acoplar um componente Monitor capaz de controlar toda a Rede Neural através de um ponto central. Assim, acoplando-se o Monitor a todas as camadas da rede é possível definir características, como o Momento e Taxa de Aprendizado, para a Rede toda simplesmente através do Monitor.

A entrada e saída de dados da Rede Neural é realizada definindo Layers específicos para leitura ou escrita nas extremidades da rede, como os InputStream e OutputStream. Os métodos mais comuns são arquivos mas pode-se realizar o intercâmbio de dados também via interface web.

Como foi comentado anteriormente, o pacote Joone é altamente adaptável e extensível dispondo ainda de diversos plugins, ampliando ainda mais suas funcionalidades. Estudos mais aprofundados podem ser encontrados em [7].

Capítulo VI: Modelagem do Problema

§ Introdução e Dados Utilizados

De maneira geral as Redes Neurais propõem o uso de um enorme volume de dados disponíveis que, muitas vezes, são pouco ou mal utilizados, transformando-os em informação útil à tomada de decisões. A capacidade de Redes Neurais Artificiais em mapear sistemas complexos, sem se a necessidade de conhecer eventuais modelos matemáticos que descrevem o seu comportamento torna-as um mecanismo muito interessante na previsão de ativos financeiros.

Entretanto, a utilização de redes neurais exige que se faça uma série de escolhas não triviais na busca de um modelo para o problema. As escolhas envolvem, por exemplo, o modo de implementação, programação de uma rede ou adoção de um software voltado para a criação de redes, o número de camadas e neurônios, o tipo de sinapses e método de aprendizado entre outros.

A idéia central da solução proposta consiste em utilizar o histórico de dados de uma ação, contendo dados como cotação, oscilação, volume de negócios,etc aliados aos indicadores estatísticos apresentados como Momento, SMI, IFR para compor o conjunto de informações que servirá de suporte para futuras tomadas de decisões.

Assim todo o histórico de informações disponíveis a respeito de uma determinada Ação será o conjunto de dados para o treinamento da Rede Neural. Os dados atuais de cotação e demais indicadores estatísticos serão os dados de entrada. E a saída esperada da Rede serão tendências de compra e venda do papel em questão. Já as "Ordens" de Compra e Venda serão dadas através do processamento de algumas regras sobre os valores de saída da Rede Neural.

Todas as informações utilizadas no projeto consistem em dados reais de Ações negociadas na Bovespa com atualizações a cada 1 minuto. O conjunto de dados disponíveis para o treinamento e, depois para entrada da Rede Neural é composto pelas seguintes informações:

- Cotação da Ação
- Valor de Compra
- Valor de Venda
- Última Oscilação
- Volume de Negociações

Indicadores Estatísticos:

- PP (Valor instantâneo do SMI)
- SMI (Média Móvel de PP)
- Momento
- IFR
- Fôlego

§ Arquitetura e Aspectos Tecnológicos

O modelo tecnológico adotado consiste num projeto de uma Rede Neural totalmente orientada a objeto construído com base no pacote JOONE (Java Object Oriented Neural Engine). O JOONE é um pacote Java que disponibiliza os objetos necessários (camadas, sinapses) para a criação de Redes Neurais baseadas no método de aprendizagem Backpropagation como foi discorrido no Capítulo V. Dessa forma, utilizando os conceitos de composição, a Rede foi completamente montada utilizando os objetos disponíveis no pacote, desde as camadas e sinapses aos monitores de aprendizado.

A tarefa de instanciar os objetos e compor a Rede será realizada por uma ferramenta construída em Linguagem Java e formulários Swing. Estarão contidas nessa ferramenta o resultado das difíceis escolhas a respeito da Rede como número de camadas, neurônios, tipos de camadas e sinapses assim como um número adequado para a quantidade de ciclos de treinamento.

Quanto a topologia, em virtude da complexidade do problema em questão, a solução escolhida foi uma Rede constituída de 3 camadas: Entrada, Intermediária e Saída. A primeira camada será composta por 10 neurônios, onde cada um receberá um dos Dados de Entrada citados acima. A segunda camada ou Camada Intermediária será composta por 20 neurônios e a Camada de Saída por apenas um neurônio responsável pela saída. Todas as camadas – e também as sinapses - são integradas por objetos monitores.

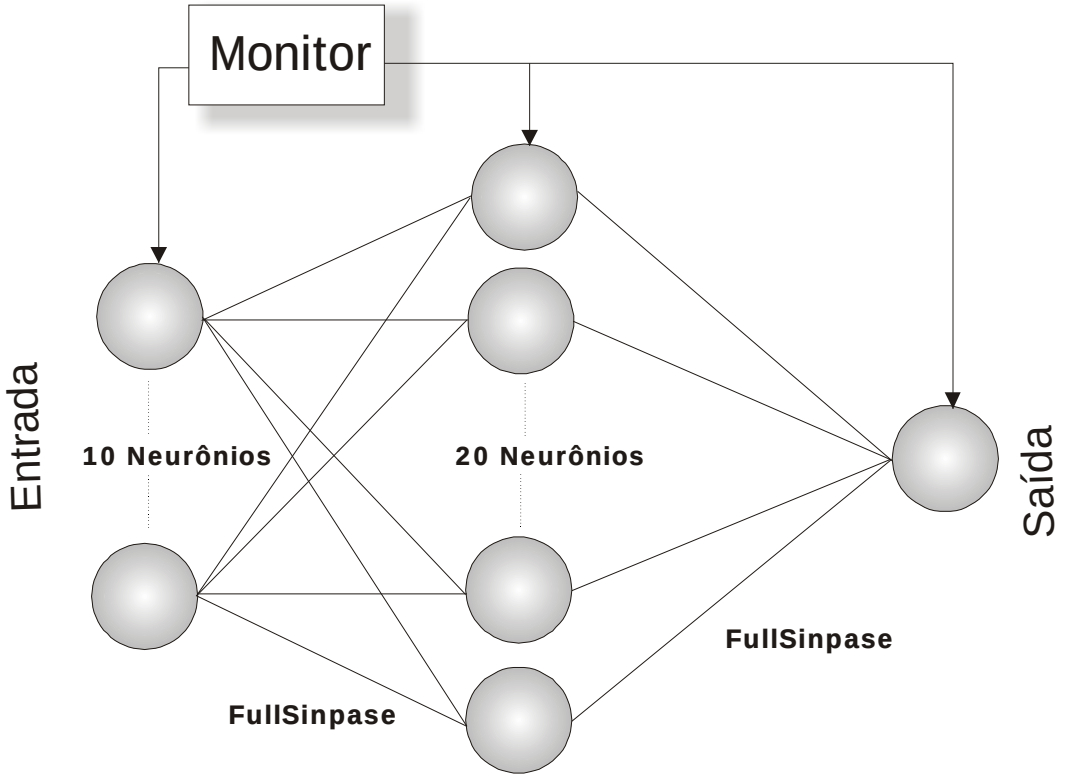

Figura 4. Ilustração da Rede Neural construída

O modelo de camadas utilizadas baseou-se na classe SigmoidLayer, cuja função de ativação é uma senóide. Testes empíricos mostraram que camadas com outros tipos de função de ativação disponíveis no pacote não seriam adequadas para o problema, pois não foram capazes de aproximar os resultados esperados. As sinapses - conexões entre as camadas - utilizaram a classe FullSinapse que representam interconexões entre todos os neurônios de uma camada e outra, aumentando a complexidade e capacidade da Rede.

Baseado em teste empíricos realizados com os dados disponíveis levaram aos valores adotados para o Momento e a Taxa de Aprendizado que ficaram respectivamente em 0.4 e 0.8.

§ Treinamento e Processamento dos Dados

O meio de intercâmbio de dados entre a Rede Neural e os meios externos será realizado por pseudo-camadas que manipulam arquivos textos. Assim, tanto a entrada de dados para treinamento ou processamento, quanto a saída da Rede Neural serão realizadas utilizando arquivos textos.

A entrada de dados será realizada através de um arquivo texto composto por 11 colunas, contendo os dados citados anteriormente mais o valor esperado da Rede, separados por ponto-e-vírgula e utilizando pontos como separadores decimais .

A saída esperada da Rede Neural será composta por valores entre 0 e 1. Os valores próximos de 0 representam boas oportunidades de Venda e próximos de 1 bons momentos de Compra da Ação em questão. A metodologia utilizada para o treinamento dos valores esperadas da Rede foi a seguinte:

- Todo momento que antecedia uma queda era marcada com valor 0
- Todo momento que antecedia uma alta era marcada com valor 1
- Os momentos restantes foram marcados com 0.5

Assim, todos os registros de dados disponíveis foram marcados, de acordo com os critérios acima, em 0, 0.5 ou 1. Utilizou-se o Microsoft Excel para tal tarefa.

Experimentalmente com dados reais de empresas negociadas na Bovespa, a Rede passou a obter resultados de aceitáveis a bons numa média de treinamentos de 15.000 a 30.000 ciclos. A quantidade de registros de treinamento variou de 150 a 200 registros.

Assim que esteja concluída a etapa de treinamento, a Rede estará apta para realizar o processamento dos dados de entrada que será realizado logo na seqüência. Entretanto a saída da rede são apenas valores numéricos variando entre 0 e 1 e como foi colocado anteriormente, como resultado final deveriam ser identificadas determinadas ordens de compra ou venda. Assim, a partir dos dados de saída da Rede Neural, foi necessário realizar um processamento baseado em algumas regras simples. Optou-se por utilizar o software Microsoft Excel para essa tarefa por facilitar a comparação dos dados de entrada versus saída e pela facilidade de representação gráfica dos dados.

As saídas obtidas da Rede foram processadas da seguinte maneira:

- § Determina-se um valor para o **Indicador de Tendências;**
- § A cada saída, calcula-se a diferença/variação entre a saída atual e a anterior. O valor obtido pode ser positivo ou negativo;
- § Compara-se a diferença calculada e o Indicador de Tendências:
  Se a Diferença > Indicador, **Ordem de Compra**
  Se a Diferença < -Indicador, **Ordem de Venda**

As saídas cujas Diferenças ficarem no intervalo [-Indicador,+Indicador] serão considerados movimentos nulos ou simplesmente manutenção da posição atual.

O valor estipulado para o Indicador de Tendências é de suma importância para que emissão correta de Ordens de Compra ou Venda. Assim, quanto menor o valor, maior a quantidade movimentos indicados e maiores os riscos de erro. De modo inverso, quanto maior o valor, menor a quantidade movimentos, mas maior a segurança de acerto.

O ciclo total de processamento dos dados pode ser visualmente analisado de acordo com a figura abaixo.

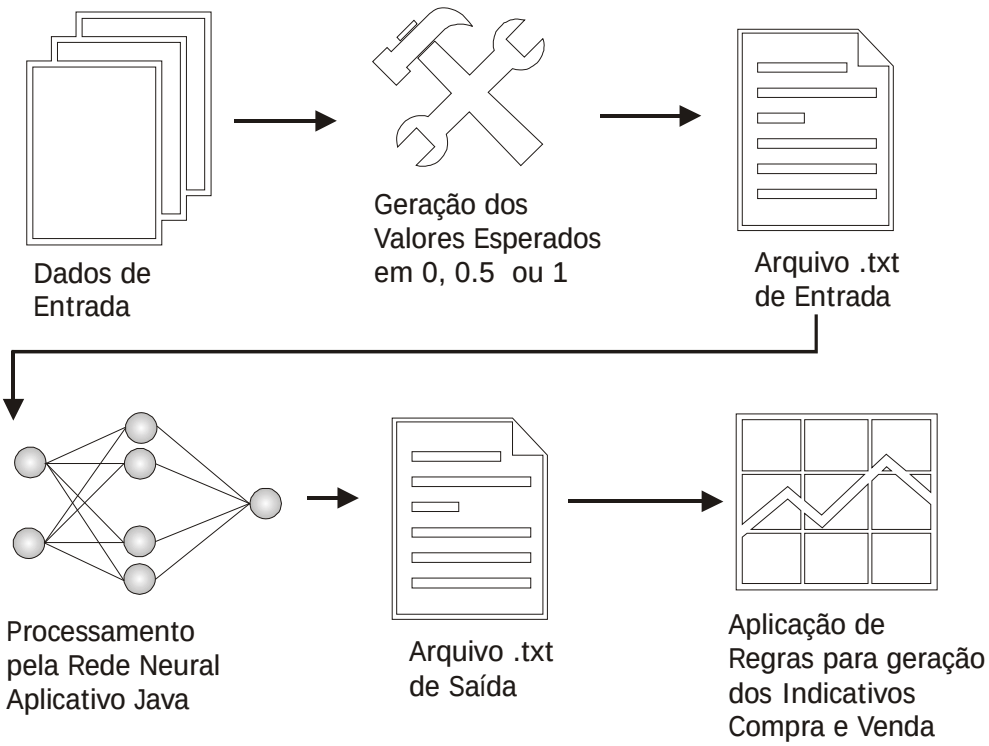

Figura 5. Representação do Processamento dos Dados

Capítulo VII: Resultados Gerados

Os principais resultados obtidos durante o trabalho de pesquisa foram o aplicativo desenvolvido em linguagem Java, responsável por instanciar os objetos do pacote Joone, treinar e processar os dados da Rede Neural e as Simulações realizadas utilizando o Microsoft Excel para avaliar e validar a eficiência da Rede.

§ Aplicativo Java

O aplicativo Java foi construído utilizando a IDE Forte para a construção dos formulários, permitindo a configuração dos parâmetros da Rede Neural de forma gráfica e simplificada. A figura abaixo ilustra a interface do aplicativo criado.

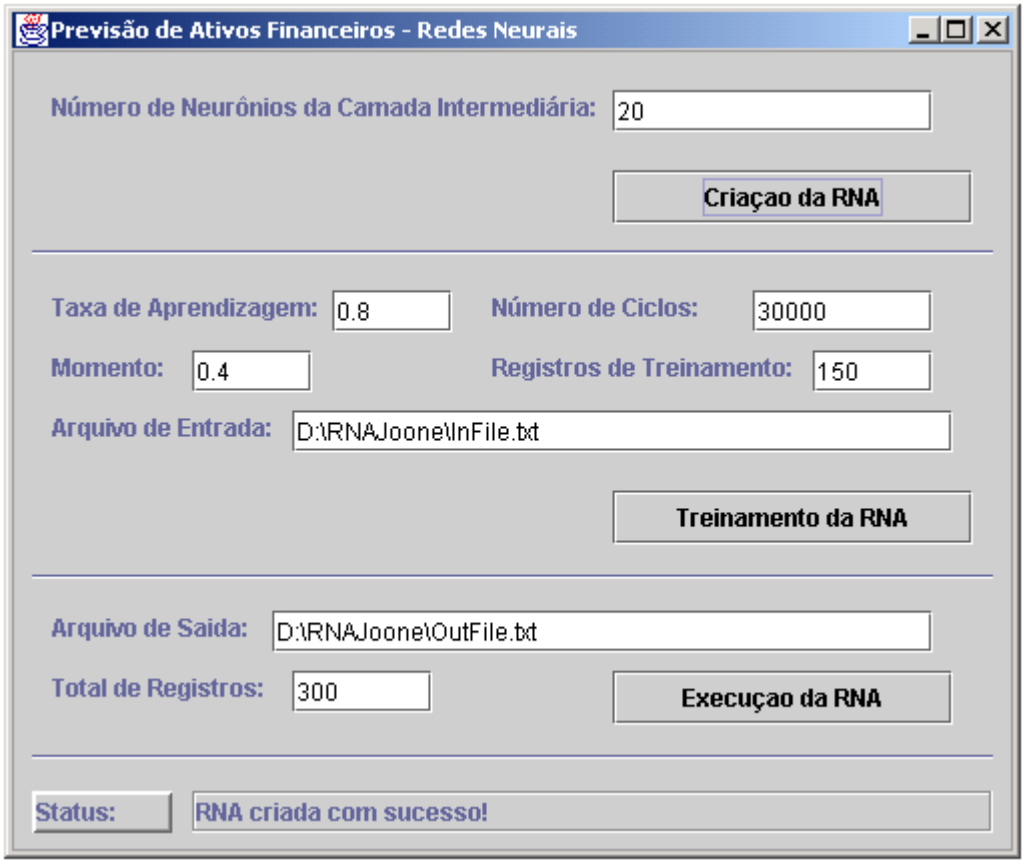

Figura 6. Interface Gráfica do Aplicativo Java executor da Rede Neural

Como é possível observar na interface criada, ficaram bastante distintas três divisões básicas do aplicativo. Primeiro a inicialização dos objetos e "montagem" geral da Rede Neural. Depois o processo de treinamento e seus respectivos parâmetros. E por fim a execução e assim o processamento efetivo dos dados.

Não somente a interface e aspectos gráficos, mas também a estrutura da classe e os métodos do aplicativo foram criados sob a divisão dessas três ações básicas: Criação, Treinamento e Execução.

O arquivo de entrada consiste, como já foi explanado anteriormente, num arquivo texto de 11 colunas onde as 10 primeiras são dados de entrada e última a saída desejada. São todos dados numéricos utilizando ponto como separador decimal e ponto-e-vírgula para separar os campos. O arquivo de saída também é um arquivo texto de coluna numérica única.

O Número Total de Registros consiste na quantidade de atualização de dados (linhas) disponíveis no arquivo texto. Para 360 linhas por exemplo, teríamos atualizações de minuto em minuto durante 6 horas.

É importante ressaltar também que o aplicativo Java precisa do pacote JOONE disponível na máquina para que o software funcione corretamente.

§ Simulações

As simulações realizadas foram feitas utilizando os dados coletados durante um dia de pregão na Bovespa. Em média o período adotado de observações foi de 6 horas, produzindo aproximadamente 300 registros de dados. Nas simulações realizadas foram utilizadas aproximadamente a metade desses dados - 150 registros – para treinamento, e o restante para execução e posterior comparação com dos valores de saída e com os esperados. Através dessas comparações foi possível avaliar a eficiência da Rede Neural construída.

A técnica utilizada para qualificar as indicações efetuadas pelo método descrito foi a seguinte: foram considerados Certos os movimentos através dos quais obteve-se Lucro, Errados aqueles trouxeram Prejuízo e Nulos aqueles que não causaram nem um nem outro. Todos os dados de entrada e os gerados pela rede passaram pelo Microsoft Excel para efeito de análise dos dados e processamento de algumas regras.

Foi utilizada para fins ilustrativos uma simulação feita no dia 06 de maio de 2004 com a empresa Usiminas, código USIM5. A figura abaixo mostra alguns dados de entrada e os respectivos Indicativos gerados de acordo com os métodos anteriormente citados:

0: Nulo
1: Compra
2: Venda

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Cotação	Compra	Venda	Oscilação	Volume	PP	SMI	Momento	IFR	Folego	Esperado	Saida RNA	Variação RNA	Indicativos
201	29,6	29,45	29,69	-4,17	9000	1	0,491	0,01	48,64	50	0	0,941686093	-0,051504594	0
202	29,46	29,45	29,5	-4,62	1000	0,2	0,531	0,24	45,49	40	1	0,993496116	0,051812024	0
203	29,5	29,45	29,53	-4,49	4000	0,625	0,456	0,11	38,13	40	0,5	0,992724054	-0,000774063	0
204	29,5	29,45	29,62	-4,49	0	0,714	0,599	0	35,82	50	0	0,62135361	-0,371370444	2
205	29,4	29,4	29,5	-4,82	7000	0	0,506	0,2	36,73	50	0,5	0,99359917	0,3722456	1
206	29,4	29,4	29,49	-4,82	2000	0	0,308	0,06	24,96	33,3333	0,5	0,99362009	2,09204E-05	0
207	29,4	29,38	29,48	-4,82	0	0,2	0,308	0,1	28,5	50	1	0,9469916836	-0,046704356	0
208	29,6	29,56	29,69	-4,17	105000	0,308	0,244	-0,1	66,52	50	0,5	0,99152179	0,044605954	0
209	29,6	29,56	29,69	-4,17	0	0,308	0,163	-0,2	66,52	50	0,5	0,8264487137	-0,165034653	2
210	29,6	29,66	29,6	-4,17	0	1	0,363	-0,2	100	100	0	0,400929347	-0,42555779	0

Figura 7. Modelo de Planilha gerado

Essa simulação em específico contou com um total de 303 registros, dos quais 150 foram utilizados como dados de treinamento da Rede Neural. O limite de variação para a qual a rede emitiria os indicativos ficou em 0.3, ou seja, um variação positiva maior que 0.3 gera um Indicativo de Compra (1) e uma variação negativa abaixo de –0.3 gera um indicativo de Venda (2).

O resultado obtido quantificado em Número de Movimentos Certos/Errados foi o seguinte:

- 12 Movimentos Nulos
- 6 Movimentos Certos
- 1 Movimentos Errados

  Saldo: R$ 0,71 - (aproximadamente 2,41% do valor inicial da Ação em cerca de 5 horas)

Visando acompanhar de forma gráfica os resultados da Rede Neural em contrapartida com as Variações da Ação em questão, essas variações foram normalizadas em valores de 0 a 1 e plotadas no mesmo gráfico de oscilações da Rede Neural, cujos valores gerados ficam no Intervalo de –1 a +1.

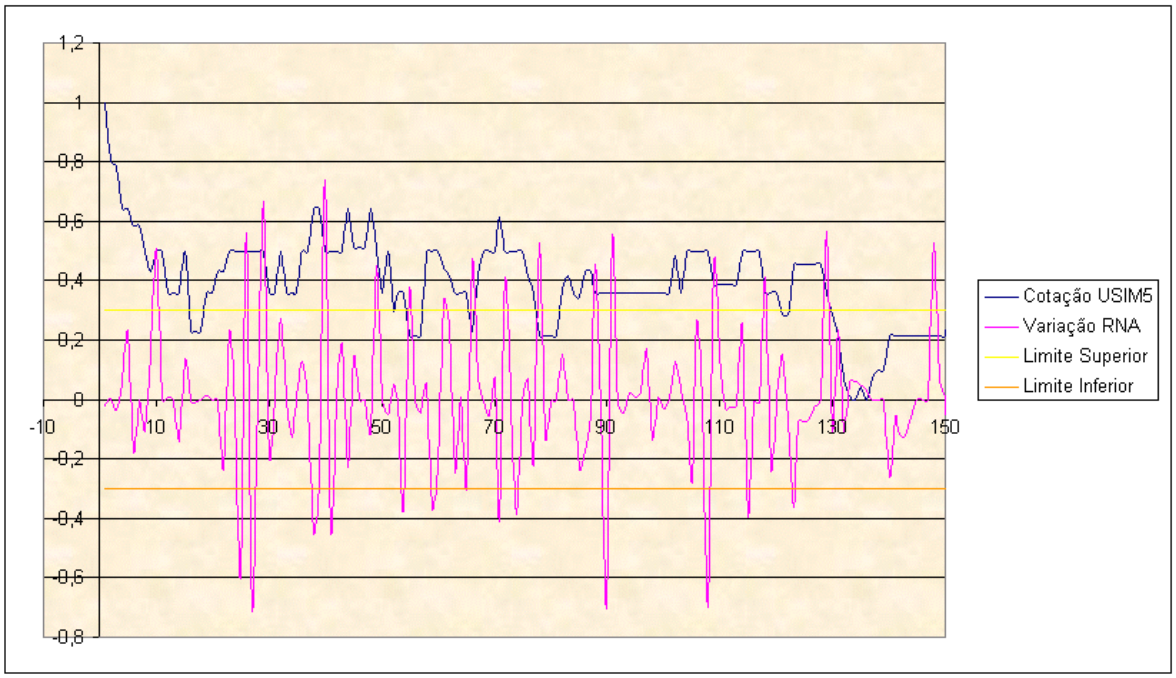

Figura 8. Gráfico para análise de eficiência da Rede Neural

No gráfico apresentado pode-se também observar os pontos em que foram gerados os Indicativos de Compra e Venda, justamente onde a Variação da Rede (linha rosa) ultrapassa as fronteiras do limite pré-estabelecido (-0,3 e + 0,3).

Capítulo VIII: Conclusão

Analisando os objetivos inicialmente propostos para o projeto e os resultados finalmente obtidos pode-se avaliar positivamente as ferramentas e métodos utilizados, já que as metas inicialmente propostas foram atingidas.

A identificação de tendências, principal objetivo do projeto, foi bem traduzida nas Ordens de Compra e Venda que puderam ser obtidas após a aplicação de algumas regras à saída obtida da Rede Neural. Dessa forma, todo a complexidade do processamento intermediário das informações pode ficar oculta sob o véu de simples Indicativos das transações a serem realizadas.

Baseado nos resultados produzidos pelas simulações realizadas pode-se avaliar também positivamente um balanço financeiro dos movimentos indicados pela Rede. Foi apurado nessas simulações que a rede produziu em média 35% de movimentos corretos - aqueles que geraram lucro - para apenas 5% de movimentos errados - que geraram prejuízo - enquanto os demais foram nulos, aqueles cujo valor compra e venda coincidiram. Considerando em termos monetários os retornos médios obtidos variaram de 1% a 3% do valor da ação, tomando uma simulação de um dia pregão, ou seja, aproximadamente 6 horas, onde metade desse tempo é utilizado para treinamento.

Comparando toda a metodologia utilizada - desde os dados de entradas e a arquitetura da rede às regras aplicadas aos dados de saída – com as referências bibliográficas encontradas, poderemos concluir que, embora semelhante em alguns aspectos, de maneira geral foi realizada uma abordagem bastante nova do problema em questão, em nenhum momento inteiramente coincidente com os métodos já existentes. Vê-se assim que não há um modelo pré-estabelecido ou alguma receita pronta para criar e modelar as Redes Neurais e os métodos subseqüentes de análise. Na maioria das vezes elas são construídas baseadas em pesquisas empíricas e na realização sucessivas de testes e análises, sendo exatamente como decorreu o presente trabalho.

A fase do projeto que demandou maior tempo e trabalho foi justamente a validação do modelo e a busca de aperfeiçoamento de resultados. Essa etapa foi realizada testando os vários tipos de camada de neurônios, sinapses, variando período de tempos e outras variáveis até que os resultados se aproximassem do desejado inicialmente. Ainda assim não descarta-se a possibilidade de aperfeiçoamento e afinamento dessas variáveis para obtenção de melhores resultados.

Finalmente podemos concluir que, embora o trabalho tenha sido árduo e difícil, exigindo pesquisa e testes sucessivos, os objetivos propostos foram atingidos gerando assim resultados satisfatórios de previsão.

Bibliografia

[1] BOSAIPO, C. R, Aplicação das Redes Neurais na Previsão de Mercados Financeiros, URL: http://www.revista.unicamp.br/infotec/artigos/claudia.html, 2001

[2] ZANETI JR, L. A., ALMEIDA, F. C, Exploração do uso de Redes Neurais na previsão do comportamento de Ativos Financeiros, URL: http://www.ead.fea.usp.br/semead/3semead/pdf/Finanças/Art061.PDF, 1998

[3] FREITAS, A.A.C., SILVA, I. N., Análise Técnica de Títulos Financeiros Através de Redes Neurais Artificiais, URL: http://www.ele.ita.br/cnrn/artigos-4cbrn/4cbrn_017.pdf, 1999

[4] LEMOS, L. S., SOUZA, J. C., RAW M., RIBEIRO, C. H. C., KIENITZ K. H., Aprendizagem Autômata para Gerenciamento de uma Bolsa de Valores simplificada. 2000.

[5] ABRAHAM , A. , BAIKUNTH N. , MAHANTI, P.K., Hybrid Intelligent Systems for Stock Market Analysis

[6] SLIM, C. , TRABELSI, A., Neural Network For Modeling Nonlinear Time Series: A New Approach

[7] MARRONE, P., The Complete Guide – All you need to know about Joone, URL: http://www.joone.org

[8] URL: http://www.bovespa.com.br

[9] URL: http://www.investshop.com.br

[10] URL: http://www.jooneworld.com

Apêndice I: Fonte do Aplicativo Java

```
import javax.swing.*;
import java.awt.event.*;

import org.joone.engine.*;
import org.joone.engine.learning.*;
import org.joone.io.*;


/*
 * RNAJoone.java, Previsão de Ativos Financeiros utilizando Redes Neurais
 */

/**
 *
 * @author  Daniel Carlos Guimarães Pedronette
 */


public class RNAJoone extends javax.swing.JFrame implements
ActionListener,NeuralNetListener {

    private double txLearn,txMomentum;
    private int PatternNumber,TrainCicles;
    private String InFile,OutFile;
    private SigmoidLayer input,hidden,output;
    private FullSynapse sinapse_IH,sinapse_HO;
    private Monitor monitor;
    private FileInputSynapse inputStream;
    private TeachingSynapse trainer;
    private FileInputSynapse samples;
    private FileOutputSynapse results;
    private FileOutputSynapse errors;


    /** Creates new form RNAJoone */
    public RNAJoone() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        buttonGroup1 = new javax.swing.ButtonGroup();
        jPanel1 = new javax.swing.JPanel();
        jBtnCreate = new javax.swing.JButton();
        jBtnTrain = new javax.swing.JButton();
        jBtnRun = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLblStatus = new javax.swing.JLabel();
        jTextArea1 = new javax.swing.JTextArea();
        jLabel2 = new javax.swing.JLabel();
```

```
        txtCamadaInter = new javax.swing.JTextField();
        jPanel2 = new javax.swing.JPanel();
        jSeparator1 = new javax.swing.JSeparator();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        txtArqEntrada = new javax.swing.JTextField();
        txtTxAprend = new javax.swing.JTextField();
        txtMomento = new javax.swing.JTextField();
        txtCiclos = new javax.swing.JTextField();
        txtRegistroTreina = new javax.swing.JTextField();
        jSeparator2 = new javax.swing.JSeparator();
        jSeparator3 = new javax.swing.JSeparator();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jSeparator4 = new javax.swing.JSeparator();
        jSeparator5 = new javax.swing.JSeparator();
        txtArqSaida = new javax.swing.JTextField();
        txtTotalRegistros = new javax.swing.JTextField();


        getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());

        setTitle("Previs\u00e3o de Ativos Financeiros - Redes Neurais");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        jPanel1.setLayout(null);

        jPanel1.setBorder(new javax.swing.border.TitledBorder(""));
        jBtnCreate.setText("Cria\u00e7ao da RNA");
        jBtnCreate.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jBtnCreateActionPerformed(evt);
            }
        });

        jPanel1.add(jBtnCreate);
        jBtnCreate.setBounds(300, 60, 180, 27);

        jBtnTrain.setText("Treinamento da RNA");
        jBtnTrain.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jBtnTrainActionPerformed(evt);
            }
        });

        jPanel1.add(jBtnTrain);
        jBtnTrain.setBounds(300, 220, 180, 27);
```

```
        jBtnRun.setText("Execu\u00e7ao da RNA");
        jBtnRun.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jBtnRunActionPerformed(evt);
            }
        });

        jPanel1.add(jBtnRun);
        jBtnRun.setBounds(300, 310, 170, 27);

        jLabel1.setText("Status:");
        jLabel1.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        jPanel1.add(jLabel1);
        jLabel1.setBounds(10, 370, 70, 21);

        jLblStatus.setText(" -");
        jLblStatus.setBorder(new javax.swing.border.EtchedBorder());
        jPanel1.add(jLblStatus);
        jLblStatus.setBounds(90, 370, 400, 21);

        jPanel1.add(jTextArea1);
        jTextArea1.setBounds(260, 120, 0, 17);

        jLabel2.setText("N\u00famero de Neur\u00f4nios da Camada
Intermedi\u00e1ria:");
        jPanel1.add(jLabel2);
        jLabel2.setBounds(20, 20, 272, 17);

        txtCamadaInter.setText("20");
        jPanel1.add(txtCamadaInter);
        txtCamadaInter.setBounds(300, 20, 160, 21);

        jPanel2.setBorder(new javax.swing.border.TitledBorder(""));
        jPanel1.add(jPanel2);
        jPanel2.setBounds(20, 110, 460, 0);

        jPanel1.add(jSeparator1);
        jSeparator1.setBounds(10, 100, 480, 10);

        jLabel3.setText("Taxa de Aprendizagem:");
        jPanel1.add(jLabel3);
        jLabel3.setBounds(20, 120, 133, 17);

        jLabel4.setText("Momento:");
        jPanel1.add(jLabel4);
        jLabel4.setBounds(20, 150, 56, 17);

        jLabel5.setText("N\u00famero de Ciclos:");
        jPanel1.add(jLabel5);
        jLabel5.setBounds(240, 120, 103, 17);

        jLabel6.setText("Registros de Treinamento:");
        jPanel1.add(jLabel6);
        jLabel6.setBounds(240, 150, 150, 17);

        jLabel7.setText("Arquivo de Entrada:");
```

```
        jPanel1.add(jLabel7);
        jLabel7.setBounds(20, 180, 110, 17);

        txtArqEntrada.setText("D:\\RNAJoone\\InFile.txt");
        jPanel1.add(txtArqEntrada);
        txtArqEntrada.setBounds(140, 180, 330, 21);

        txtTxAprend.setText("0.8");
        jPanel1.add(txtTxAprend);
        txtTxAprend.setBounds(160, 120, 60, 21);

        txtMomento.setText("0.4");
        jPanel1.add(txtMomento);
        txtMomento.setBounds(90, 150, 60, 21);

        txtCiclos.setText("30000");
        jPanel1.add(txtCiclos);
        txtCiclos.setBounds(370, 120, 90, 21);

        txtRegistroTreina.setText("150");
        jPanel1.add(txtRegistroTreina);
        txtRegistroTreina.setBounds(400, 150, 60, 21);

        jPanel1.add(jSeparator2);
        jSeparator2.setBounds(10, 100, 480, 10);

        jPanel1.add(jSeparator3);
        jSeparator3.setBounds(10, 262, 480, 10);

        jLabel8.setText("Arquivo de Saida:");
        jPanel1.add(jLabel8);
        jLabel8.setBounds(20, 280, 98, 17);

        jLabel9.setText("Total de Registros:");
        jPanel1.add(jLabel9);
        jLabel9.setBounds(20, 310, 106, 17);

        jPanel1.add(jSeparator4);
        jSeparator4.setBounds(10, 352, 480, 0);

        jPanel1.add(jSeparator5);
        jSeparator5.setBounds(10, 352, 480, 10);

        txtArqSaida.setText("D:\\RNAJoone\\OutFile.txt");
        jPanel1.add(txtArqSaida);
        txtArqSaida.setBounds(130, 280, 330, 21);

        txtTotalRegistros.setText("300");
        jPanel1.add(txtTotalRegistros);
        txtTotalRegistros.setBounds(140, 310, 70, 21);

        getContentPane().add(jPanel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 500, 400));

        pack();
    }//GEN-END:initComponents
```

```
    private void jBtnRunActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jBtnRunActionPerformed
        runRNA();
        jLblStatus.setText("Valores processados com sucesso!");
    }//GEN-LAST:event_jBtnRunActionPerformed

    private void jBtnTrainActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jBtnTrainActionPerformed
        trainRNA();
    }//GEN-LAST:event_jBtnTrainActionPerformed

    private void jBtnCreateActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jBtnCreateActionPerformed
        initRNA();
        jLblStatus.setText("RNA criada com sucesso!");
    }//GEN-LAST:event_jBtnCreateActionPerformed

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
        System.exit(0);
    }//GEN-LAST:event_exitForm

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        new RNAJoone().show();
    }

    /**
     *Inicializa Estrutura Básica da Rede Neural: Camadas e Sinapses
     */
    public void initBasicRNA () {
        //*************************************************************
        //Inicializando as Camdas da RNA
        //*************************************************************
        input = new SigmoidLayer ();
        input.setLayerName("Entrada de Dados");
        hidden = new SigmoidLayer ();
        hidden.setLayerName("Camada Intermediária");
        output = new SigmoidLayer ();
        output.setLayerName("Camada de Saída");
        //*************************************************************
        //Número de Neurônios em cada Camada
        //*************************************************************
        input.setRows(10);      //Entrada
        hidden.setRows(20);     //Intermediária
        output.setRows(1);      //Saída
        //*************************************************************
        //Inicializando as Sinapses
        //*************************************************************
        sinapse_IH = new FullSynapse();
        sinapse_IH.setName("Ligação da Camada Entrada/Intermediária");
        sinapse_HO = new FullSynapse();
        sinapse_HO.setName("Ligação da Camada Intermediária/Saída");
        //*************************************************************
```

```
        //Conectando as Camadas
        //*************************************************************
        // Conectando a camada de entrada à intermediária
        input.addOutputSynapse(sinapse_IH);
        hidden.addInputSynapse(sinapse_IH);
        // Conectando a camda intermediária à de saída
        hidden.addOutputSynapse(sinapse_HO);
        output.addInputSynapse(sinapse_HO);
    }

    /**
     *Inicializa valores padrão de configurações da Rede Neural
     */
    public void initRNAConfigVars (){
        //Inicializando configurações default
        txLearn = 0.8;
        txMomentum = 0.4;
        PatternNumber = 150;
        TrainCicles = 30000;
        InFile = new String("D:\\RNAJoone\\InFile.txt");
        OutFile = new String("D:\\RNAJoone\\OutFile.txt");
    }

    /**
     *Seta configurações do Monitor da Rede, de acordo com as
propriedades da classe
     */
    public void configMonitor(){
        monitor.setLearningRate(txLearn);
        monitor.setMomentum(txMomentum);
        monitor.setTrainingPatterns(PatternNumber);
        monitor.setTotCicles(TrainCicles);
    }

    /**
     * Inicializa e Configura o Monitor, Mecanismo de Aprendizagem,
     *      Interface das Camadas de Entrada e Saída com os arquivos
texto
     */
    public void initConfigRNA () {
        //*************************************************************
        //Inicializando o Monitor e seus Parâmetros de Aprendizagem
        //*************************************************************
        monitor = new Monitor();
        configMonitor();
        //Atribuindo o monitor as camadas da RNA
        input.setMonitor(monitor);
        hidden.setMonitor(monitor);
        output.setMonitor(monitor);
        monitor.addNeuralNetListener(this);
        //*************************************************************
        //Definindo as entradas da RNA - Arquivo texto
        //*************************************************************
        inputStream = new FileInputSynapse();
        inputStream.setFileName(InFile);
        //Selecionando os Campos
        //Cotaçao,Compra,Venda,Oscilaçao,Volume,PP,SMI,Momento,IFR,Folego
```

```
            inputStream.setFirstCol(1);
            inputStream.setLastCol(10);
            //Definindo a sinapse de Entrada
            input.addInputSynapse(inputStream);
            //*************************************************************
            //Definindo a Saída da RNA - Arquivo texto
            //*************************************************************
            results = new FileOutputSynapse();
            results.setFileName(OutFile);
            //*************************************************************
            //Definindo o Treinamento da Rede
            //*************************************************************
            trainer = new TeachingSynapse();
            trainer.setMonitor(monitor);
            //Arquivo de Saídas desejadas
            samples = new FileInputSynapse();
            samples.setFileName(InFile);
            //Definindo as colunas desejadas no arquivo
            samples.setFirstCol(11);
            samples.setLastCol(11);
            //Define os valores desjados para o trainer
            trainer.setDesired(samples);
        }

        /**
         * Atualiza Parâmetros do Formulário para a Criaçao da Rede
         */
        public void refreshToCreate () {
            int convert = Integer.parseInt(txtCamadaInter.getText());
            hidden.setRows(convert);
        }

        /**
         *  Executas procedimentos de Criaçao da Rede Neural
         */
        public void initRNA (){
            initRNAConfigVars();
            initBasicRNA();
            initConfigRNA();
            refreshToCreate();
        }

        /**
         * Inicializa Camadas da Rede Neural
         */
        public void startRNALayers(){
            input.start();
            hidden.start();
            output.start();
            trainer.start();
        }

        /**
         * Altera configurações do Monitor e Camada de Saída para o
Treinamento
         */
        public void setRNAtoTrain () {
```

```
        trainer.setMonitor(monitor);
        output.addOutputSynapse(trainer);
    }

    /**
     * Atualiza Parâmetros do Formulário para o Treinamento
     */
    public void refreshToTrain () {
        txLearn = Double.parseDouble(txtTxAprend.getText());
        txMomentum = Double.parseDouble(txtMomento.getText());
        PatternNumber = Integer.parseInt(txtRegistroTreina.getText());
        TrainCicles = Integer.parseInt(txtCiclos.getText());
        configMonitor();
        setInFile(txtArqEntrada.getText());
    }

    /**
     * Executa procedimentos do Treinamento da Rede
     */
    public void trainRNA(){
        refreshToTrain();
        setRNAtoTrain();
        startRNALayers();
        monitor.setLearning(true);
        monitor.Go();
    }

    /**
     * Configura Monitor, Camadas de Entrada e Saída e Interface de
Arquivos Texto
     *    para execuçao da Rede e processamento dos dados
     */
    public void setRNAtoRun(){
        output.removeOutputSynapse(trainer);
        inputStream.resetInput();
        samples.resetInput();
        results.setMonitor(monitor);
        output.addOutputSynapse(results);
    }

    /**
     * Atualiza parâmetros do Formulário para execução da Rede
     */
    public void refreshToRun () {
        setRegCount(Integer.parseInt(txtTotalRegistros.getText()));
        setOutFile(txtArqSaida.getText());
    }

    /**
     * Executa procedimentos para Execução da Rede Neural
     */
    public void runRNA () {
        setRNAtoRun();
        startRNALayers();
        refreshToRun();
        monitor.setTotCicles(1);
        monitor.setLearning(false);
```

```
        monitor.Go();
    }

    /**
     * Seta Caminho do Arquivo de Entrada
     */
    public void setInFile (String newInFile) {
        InFile = newInFile;
        inputStream.setFileName(InFile);
        samples.setFileName(InFile);
    }

    /**
    * Seta Caminho do Arquivo de Saída
    */
    public void setOutFile (String newOutFile) {
        OutFile = newOutFile;
        results.setFileName(OutFile);
    }

    /**
     * Seta contador de registros a serem processados
     */
    public void setRegCount (int regcount) {
        monitor.setTrainingPatterns(regcount);
        monitor.setValidationPatterns(regcount);
    }


  /**
   * Procedimento chamado pela Rede Neural quando acaba o treinamento
   */
  public void netStopped(NeuralNetEvent e) {
       jLblStatus.setText("RNA treinada com sucesso!");
  }

  /**
   * Procedimento executado pela rede visando acompanhar o andamento do
   *    treinamento
   */
  public void cicleTerminated(NeuralNetEvent e) {
    Monitor mon = (Monitor)e.getSource();
    long c = mon.getCurrentCicle();
    if (monitor.isLearning())
        jLblStatus.setText(c + " ciclos restantes | Erro Atual = " +
mon.getGlobalError());
  }

  public void netStarted(NeuralNetEvent e) {
  }

  public void errorChanged(NeuralNetEvent e) {
  }

  public void netStoppedError(NeuralNetEvent e, String str) {
  }
```

```
    public void actionPerformed(ActionEvent e){
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton jBtnCreate;
    private javax.swing.JButton jBtnTrain;
    private javax.swing.JButton jBtnRun;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLblStatus;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JTextField txtCamadaInter;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JTextField txtArqEntrada;
    private javax.swing.JTextField txtTxAprend;
    private javax.swing.JTextField txtMomento;
    private javax.swing.JTextField txtCiclos;
    private javax.swing.JTextField txtRegistroTreina;
    private javax.swing.JSeparator jSeparator2;
    private javax.swing.JSeparator jSeparator3;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JSeparator jSeparator4;
    private javax.swing.JSeparator jSeparator5;
    private javax.swing.JTextField txtArqSaida;
    private javax.swing.JTextField txtTotalRegistros;
    // End of variables declaration//GEN-END:variables

}
```