LUIZ PAULO RODRIGUES DE FREITAS PARREIRAS

MODELO GENÉTICO-NEURAL DE GESTÃO DE CARTEIRAS DE AÇÕES

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro de Produção.

São Paulo
2003

LUIZ PAULO RODRIGUES DE FREITAS PARREIRAS

MODELO GENÉTICO-NEURAL DE GESTÃO DE CARTEIRAS DE AÇÕES

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro de Produção.

Orientador:
Prof. Dr. Tamio Shimizu

São Paulo
2003

FICHA CATALOGRÁFICA

Parreiras, Luiz Paulo Rodrigues de Freitas
Modelo Genético-Neural de Gestão de Carteiras de Ações. São Paulo, 2003.
127 p.

Trabalho de Formatura – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Produção.

1. Finanças 2. Redes Neurais 3. Algoritmos Genéticos. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção II.t

Para meus pais Aurea e Alvaro.

AGRADECIMENTOS

Gostaria de agradecer a meus pais, Aurea e Alvaro, que infelizmente já não estão mais aqui – seu esforço instigou minha curiosidade intelectual e a vontade de superação. A minha irmã, Mariana, amiga e companheira, pela força nos momentos difíceis que passamos, e pelo apoio incessante. E ao resto de minha família, pelos momentos felizes, e os nem tanto - vocês são parte daquilo que eu sou.

Devo ainda agradecer ao Prof. Tamio Shimizu, cuja insistência em aplicação prática fez com que este trabalho fosse mais do que uma mera acumulação de teorias. Ao Prof. Ademar Ferreira, que me ensinou sobre redes neurais. A todos os meus colegas nestes anos de Poli – seu apoio foi indispensável.

À Hedging-Griffo, empresa em que trabalho, que me acolheu e me motivou a aprender e me esforçar. Ao Marcelo Cavalheiro e o time do Research, que me ensinaram boa parte do que sei sobre os mercados de ações. Ao Júlio, Dilton e toda a equipe de Gestão, que me receberam, me incentivaram e continuam me ensinando sobre os mercados, e porque não, sobre a vida. E ao Luís Stuhlberger, cujo talento é guia para todos nós.

Por fim, a todos aqueles que, direta ou indiretamente, contribuíram com a elaboração deste trabalho.

RESUMO

O presente trabalho busca reunir duas ferramentas de Inteligência Artificial, as redes neurais e os algoritmos genéticos, e colocá-las a serviço da construção de carteiras de ações. Dentro do paradigma teórico da Moderna Teoria de Portfólios, o problema de construção de carteiras de investimento tem como base de sua solução as expectativas futuras de retornos e riscos dos ativos, sendo então possível otimizar carteiras para obter mais retorno com menos risco. Dada a necessidade de ter boas previsões dos retornos e riscos futuros, devem-se buscar ferramentas adequadas para a realização de tais previsões.

Para prever os retornos futuros, serão testadas diversas configurações de redes neurais, com cinco algoritmos de treinamento diferentes, de maneira a determinar qual tem melhor performance e assim aplicá-la à previsão dos retornos de 25 ações diferentes, todas negociadas na Bolsa de Valores de São Paulo, a Bovespa. As redes serão treinadas receberão vários dados como *input*, e deverão aprender a dar como saída a cotação da ação no dia seguinte. Para o treinamento serão usadas cotações de 900 dias, para validação 100 dias subseqüentes, e para o teste do modelo, mais 100 dias.

Para prever os riscos futuros, será utilizado um modelo GARCH (1,1) com janela deslizante, de maneira a realizar previsões um dia à frente com base nos últimos 21 dias.

Obtidas as previsões de retornos e riscos futuros para cada uma das 25 ações, buscar-se-á então construir carteiras que otimizem a relação risco-retorno para o investidor. Dado o tamanho do espaço de soluções possíveis, é necessário uma heurística de busca eficiente, e para tanto foi escolhido um modelo de algoritmo genético, que se mostra eficiente em obter boas soluções em termos de retorno e risco para o investidor.

Ao fim, conclui-se que as redes neurais, quando propriamente modeladas e treinadas, podem realizar boas previsões de preços futuros de ações, e quando integradas num modelo integrado de risco-retorno baseado numa heurística de busca eficiente, geram retornos superiores. Em outras palavras, um modelo de Rede Neural – Algoritmo Genético é uma ferramenta adequada para a gestão de carteiras de ações.

ABSTRACT

This work represents an attempt to bring together two leading tools from Artificial Intelligence, namely Neural Networks and Genetic Algorithms, and put them to work on managing stock portfolios. Under the Modern Portfolio Theory framework, the task of building investment portfolios depends on the knowledge of future returns and risks for a given set of assets, which opens up the possibility of optimization in search of more returns with less risk. Therefore, good forecasting of future returns and volatility is essential to building good portfolios, and advanced tools such as the ones here discussed may be good candidates.

The forecasting of future returns is done with Neural Networks. Several configurations are tested, with five different training algorithms, in order to establish the best performing one, and then apply it to forecasting returns for 25 different stocks, all traded on São Paulo Stock Exchange (Bovespa). Inputs will contain several different data, and the networks will be trained to predict next day's stock price. Training will be done with 900 quotes, validation with the next 100 quotes, and the forecasts on the following 100 days.

Volatility (ie. risk) forecasting will be done by a GARCH (1,1) model, with a 21-day sliding window, and predictions one day ahead.

Given the forecasts for future returns and risks for each of the 25 stocks, under the mean-variance theoretical framework the next step is finding portfolios that provide the adequate mix of returns and risk. Given the sheer size of the search space, a good heuristic for finding portfolios is needed, and genetic algorithms are just that tool. A GA-based model is then built, and it demonstrates good performance and efficiency.

To sum up, the work clearly demonstrates the capabilities of neural networks applied to stock price forecasting. Furthermore, it shows that integrating such predictions under a Genetic Algorithm-based model creates portfolios with good returns and low risk. In other words, a Neural Network-Genetic Algorithm model provides an adequate tool for stock portfolio management.

Índice

4. Algoritmos Genéticos

5. Inteligência Artificial em Finanças: Análise da Pesquisa Prévia

6. Modelos & Experimentos

7. Conclusões

8. Bibliografia

Índice de Figuras

Índice de Tabelas

1. INTRODUÇÃO

"O que seria a história, senão o enorme prazer de se conhecer os detalhes, e não só os padrões abstratos?"

Stephen Jay Gould[1]

1.1. Motivação

Disse George Soros: "*Os mercados acionários são um excelente laboratório para o teste de teorias*"[2]. Em nenhum outro campo, as doses de risco são tão abundantes. Como também o são as potenciais recompensas e perdas. Em um ambiente em que a maioria pode se tornar pobre ou inimaginavelmente rica com o advento de uma notícia, ou com o piscar de uma cotação na tela – aí as oportunidades e os desafios intelectuais são enormes. Especialmente em mercados pouco maduros e constantemente voláteis como o brasileiro, onde as ferramentas ainda não foram plenamente desenvolvidas, e a sofisticação dos *players* ainda não atingiu os níveis de uma Wall Street ou da City de Londres.

Assim, a motivação básica por trás do presente trabalho é a tentativa de obter modelos que gerem bons retornos para o investidor. No caso específico aqui discutido, modelos baseados em ferramentas de Inteligência Artificial, tentando criar uma ponte entre o estado-da-arte em termos de técnicas de previsão e busca, com um conhecimento dos detalhes do dia-a-dia do mercado de ações. As técnicas de engenharia são úteis na abordagem do problema, no seu tratamento quantitativo, e também na análise das limitações dos resultados obtidos. Obter retornos em excesso aos do mercado não é tarefa simples (uma enormidade de recursos financeiros e intelectuais tem sido dedicada ao problema nas últimas décadas), e este trabalho não aspira, *a priori*, a resolver de uma vez por todas a questão. A idéia básica é trazer para o mercado brasileiro uma linha de pesquisa que muitos frutos rendeu (em trabalhos acadêmicos com certeza absoluta, em retornos financeiros já não se sabe com certeza) no exterior.

[1] Apud Bernstein, 1992.

[2] Apud Bass, 2000.

1.2. Objetivos

O objetivo fundamental deste trabalho é aplicar, de maneira realista, um modelo de previsão do mercado de ações baseado em Redes Neurais, utilizando toda a potencialidade da ferramenta para a obtenção de boa performance preditiva, e a partir daí integrar tais previsões a um modelo de alocação de ativos baseado em Algoritmos Genéticos, outra ferramenta poderosa de Inteligência Artificial, para obter carteiras de investimento gerenciadas dinamicamente, que resultem em retornos em excesso aos de mercado.

Embora tenha um cunho acadêmico, este trabalho é feito com uma preocupação acerca de sua aplicabilidade prática. Isso é o que o distingue fortemente de outros trabalhos de mesmo cunho[3]. Assim, é enfatizado o papel da liquidez no mercado de ações brasileiro, além de algumas características peculiares deste mercado, que introduzem vieses que devem ser tratados com cuidado, para que os resultados não se percam, por inúteis e inaplicáveis.

Vale dizer que, de um modo geral, são muito poucos os trabalhos brasileiros que se propuseram a tarefa semelhante à deste trabalho. A literatura de aplicações em finanças nos mercados brasileiros é razoavelmente escassa, e são poucos os autores que se aventuraram além do conhecido e bem-estabelecido pela experiência internacional. A literatura na interface Finanças-Inteligência Artificial então, é ainda mais escassa. A crônica falta de recursos para pesquisa no país provavelmente tem muito a ver com isso, mas talvez a razão de fundo seja mesmo as constantes crises econômicas que o país vivenciou na última década. Como deve ficar claro no item 2.3 deste trabalho, há uma relação intrínseca e simbiótica entre o desenvolvimento econômico e o desenvolvimento dos mercados financeiros em um país, e a variedade de crises e a falta de crescimento do país certamente não contribuíram para isso. De todo modo, é em ambientes de elevada volatilidade – como o mercado brasileiro –

[3] Lazo Lazo (2000) por exemplo, tem os mesmos fundamentos teóricos e modelos razoavelmente semelhantes aos aqui desenvolvidos. Contudo, parte de seus resultados não faz sentido por não levar em conta o problema de liquidez do mercado de ações. É disso que trata a frase “preocupação com a aplicabilidade prática”.

que ferramentas de boa qualidade para avaliação de riscos e retornos, além de boa alocação de ativos, permitem a obtenção consistente de performances positivas.

É nesse contexto que este trabalho é desenvolvido, visando aplicar uma perspectiva de engenharia ao projeto de um modelo consistente de gestão de carteiras de ações, usando as ferramentas mais avançadas disponíveis.

1.3. O Estágio

O estágio foi desenvolvido na Hedging-Griffo Corretora de Valores S.A. A Hedging-Griffo é uma corretora de valores e administradora de recursos. Especificamente, o estágio ocorreu nesta segunda área, onde são administrados diversas carteiras e fundos de investimentos.

A Hedging-Griffo foi fundada em 1981, inicialmente apenas como corretora de valores, especializando-se primeiramente em commodities agrícolas, e mais tarde se expandindo para os mercados de commodities financeiras e ações na Bovespa. Consolidou sua posição como uma das maiores do mercado ao longo das décadas de 80 e 90, e em 1992 iniciou as atividades da Hedging-Griffo Asset Management, o braço de administração de recursos. Em 1997 criou o fundo HG Verde FIF, o mais antigo *hedge fund* (fundo multimercado, que investe em diversos ativos diferentes simultaneamente) em atividade no país. Atualmente a empresa administra por volta de seis bilhões de reais em recursos de terceiros.

Inicialmente o estágio foi desenvolvido na área conhecida como *Research* dentro da empresa. Esta área é responsável por estudar e analisar empresas cujas ações são cotadas na Bolsa de Valores de São Paulo, a Bovespa. Além disso, a equipe, composta de cinco pessoas, administra fundos e carteiras com recursos da ordem de R$ 30 milhões, integralmente aplicados em ações. Nesta área, foram desenvolvidos estudos quantitativos e de análise fundamental de empresas, além de estudos e desenvolvimento de metodologias para melhor gestão cotidiana dos recursos administrados.

Posteriormente, o estágio passou a ser desenvolvido na área conhecida como *Gestão*. Esta área é responsável pela administração de uma grande gama de fundos de investimento, com recursos da ordem de R$ 2,5 bilhões, incluindo o carro-chefe

da empresa, o já citado HG Verde. Nesta área, foram desenvolvidos estudos e aplicações para melhor alocação e controle dos recursos investidos, e também para auxiliar na tomada de decisão sobre investimentos em diversos mercados diferentes (ações, renda fixa, derivativos, câmbio, entre outros).

Tanto a primeira quanto a segunda etapa do estágio permitiram ao autor aprofundar e expandir seu conhecimento sobre o mercado financeiro, seus detalhes intricados, qual tipo de modelagem é possível fazer, quais os reais limites de métodos quantitativos e qualitativos de tomada de decisão, e também observar, cotidianamente, como tais decisões podem ter impacto sobre a rentabilidade obtida dos recursos, e em última instância, o dinheiro que os investidores confiam à empresa para gerir. Tais conhecimentos foram fundamentais para o desenvolvimento deste trabalho.

1.4. Descrição e Organização do Trabalho

O trabalho é organizado em cinco partes básicas. A premissa básica é solidificar uma base teórica, apresentar as aplicações práticas passadas desta teoria, e por fim acrescentar a contribuição original.

A parte teórica começa no capítulo 2, onde são apresentados os conceitos fundamentais de finanças e dos mercados financeiros. Após uma introdução e um breve histórico, são discutidos os principais conceitos que envolvem os mercados financeiros, sua relação com a economia como um todo, e a famosa Hipótese dos Mercados Eficientes. Entram em cena então os conceitos por trás de tentativas de modelar e prever os mercados, seguidos da chamada Moderna Teoria de Portfólios, talvez o marco fundamental da teoria de finanças da segunda metade do século XX. Por fim, são discutidos os conceitos básicos de Risco aplicado ao mercado, e as principais métricas de avaliação de decisões.

No capítulo 3 são introduzidos as Redes Neurais artificiais. Novamente inicia-se com uma introdução e um histórico, passando aos conceitos básicos de redes neurais, quais sejam, o neurônio artificial e suas características básicas, os tipos básicos de redes e os principais processos de aprendizagem. Restringindo o foco, é apresentado

o algoritmo *Backpropagation*, base dos modelos discutidos adiante. Por fim, algumas variáveis importantes no projeto de redes de alta performance são discutidas.

No capítulo 4 é apresentada a outra ferramenta de Inteligência Artificial usada aqui: os Algoritmos Genéticos. O capítulo é construído de modo a facilitar a compreensão do desenvolvimento e das potencialidades do uso do algoritmo. Assim, ao histórico seguem-se seções sobre o relacionamento entre a biologia e a ferramenta, os elementos básicos de construção de um AG, um modelo genérico, e por fim uma comparação com métodos de busca mais tradicionais e estabelecidos.

Concluída a introdução teórica, passa-se a uma discussão acerca daquilo que já foi obtido por pesquisas passadas, na questão de conjugar Inteligência Artificial com aplicações de finanças. São abordadas separadamente as contribuições de Redes Neurais e de Algoritmos Genéticos à resolução de problemas da área, notadamente aqueles que envolvem previsão de mercados (no primeiro caso) e alocação em carteiras (no segundo). As principais conclusões tiradas da pesquisa prévia são então discutidas, levantando benefícios e limitações, e apontando caminhos para a construção dos modelos.

Finalmente, no capítulo 6, são apresentados os modelos construídos e aplicados. Primeiramente são apresentados os dados utilizados nos experimentos, suas limitações e características fundamentais. Depois, o modelo de Rede Neural desenvolvido é detalhado e aplicado aos dados, para a obtenção de previsões de retornos das ações. Aqui, são explicados em detalhe o pré e o pós-processamento, as principais escolhas de parâmetros e sua razão. A seguir, é discutido brevemente o modelo GARCH aplicado à previsão dos riscos associados a cada ação, e são apresentadas as previsões obtidas. Finalmente, o modelo de Algoritmos Genéticos é discutido, e são apresentados os resultados de alocação de carteiras obtidos pelo modelo. Finalmente, uma conclusão com os principais resultados é apresentada.

No capítulo 7, são apresentadas e discutidas as principais conclusões do trabalho. As falhas e pontos fortes são o tema fundamental, na tentativa de encaminhas pesquisas futuras e sugerir potenciais extensões do presente trabalho.

2. FINANÇAS & MERCADOS

"*Posso calcular os movimentos dos corpos celestes, mas não a loucura das pessoas"*

Sir Isaac Newton, após perder £20.000 no mercado de ações[1]

2.1. Introdução

Desde épocas imemoriais, tem sido um dos objetivos fundamentais do homem aumentar o seu bem-estar. Um dos meios que ele encontrou para isso foi através da obtenção de ganhos através dos mercados, inicialmente através de trocas de mercadorias físicas, mais tarde tendo a mediação do dinheiro, uma espécie de graxa que permite às rodas econômicas girarem. E como todo ativo, o dinheiro acabou por ter seu próprio mercado, o financeiro. Em torno deste se elaboraram teorias e pesquisas, usualmente englobadas no campo de finanças. É deste estudo que trata o presente capítulo.

O objetivo fundamental é entender os conceitos básicos por trás do funcionamento do mercado financeiro, e também quais as principais teorias que permitem entender e otimizar a busca de ganhos através da gestão adequada dos ativos financeiros.

2.2. Histórico

Mercados são praticamente tão antigos quanto o homem moderno. Traçar sua origem talvez seja impossível, e não é o propósito deste trabalho. Contudo, é possível apontar as origens dos modernos mercados financeiros (e das crises que os acompanham) é menos difícil. Talvez o primeiro relato completo dos tempos modernos seja na Holanda, com o mercado de tulipas. Ali é possível ver os elementos modernos que caracterizam os mercados financeiros: um local central para as operações, a tentativa de agregar os resultados das mesmas, e até a criação de instrumentos derivativos[2]. Posteriormente, os mercados financeiros ajudaram a

[1] Apud Dunbar, 2000.

[2] Para um relato detalhado do mercado de tulipas, ver *Devil Take the Hindmost*, de Edward Chancellor. Este livro traz um histórico completo e fundamentado de todas as grandes crises da história dos mercados financeiros.

financiar a expansão colonial inglesa (o que resultou na chamada "Bolha do Mar do Sul", ou *South Sea Bubble*) e a guerra de Secessão nos Estados Unidos, passando a ser parte integral do dia-a-dia econômico das maiores economias do mundo.

Embora os mercados sejam razoavelmente antigos, a moderna teoria de finanças encontra suas raízes apenas no começo do século[3]. É do francês Louis Bachelier o primeiro trabalho que busca explicar o comportamento dinâmico dos preços dos ativos financeiros. Isso ocorreu no começo do século vinte – mas o trabalho de Bachelier permaneceu relegado a segundo plano durante quase cinqüenta anos. Embora grandes economistas como Keynes tenham trabalhado na questão, nunca formalizaram os conceitos fundamentais da dinâmica dos preços. Coube a Harry Markowitz, em 1959, dar o passo fundamental nessa direção, estabelecendo o formalismo básico para o estudo de finanças, com o paradigma da média-variância. Seu aluno William Sharpe levou a teoria além, com a criação do *CAPM*, uma teoria que permite explicar o preço de qualquer ativo. Num contexto mais genérico, Samuelson e posteriormente Fama buscaram explicar o comportamento do mercado financeiro como um todo, formulando a controversa *Hipótese dos Mercados Eficientes*. Daí em diante, o nível de sofisticação analítica da teoria de finanças avançou a passos largos, com o desenvolvimento de instrumentos derivativos, como *swaps* e futuros, além das opções, cujo passo fundamental foi a descoberta, por Black, Merton e Scholes nos anos 70, da famosa fórmula de Black & Scholes. Hoje, a moderna teoria de finanças envolve conceitos de campos tão distintos quanto psicologia e cálculo estocástico, tendo se ramificado em diversos sub-campos de estudo.

2.3. Mercados Financeiros

Poucas coisas despertam reações tão distintas quanto os mercados financeiros: há aqueles que os consideram cassinos (normalmente os que não os conhecem), há aqueles que os consideram requisito fundamental para o desenvolvimento econômico (normalmente economistas excessivamente ortodoxos). E há uma enorme gama que fica entre estes dois pólos do espectro. De uma certa maneira, são ambas as coisas:

[3] Para um relato completo do desenvolvimento da teoria de finanças, ver Bernstein (1993) e também Sornette (2003). Esta seção é baseada fundamentalmente no primeiro livro.

têm um papel fundamental no desenvolvimento econômico, ao permitir o financiamento de negócios e de governos, mas também têm uma certa dose de jogo envolvida, pois são o *locus* onde se ganha e se perde dinheiro cotidianamente[4].

Nos itens a seguir, vai-se examinar o relacionamento entre a economia e os mercados, se delinear brevemente o mercado brasileiro, e por fim se examinar a famosa *Hipótese dos Mercados Eficientes*, um preâmbulo antes de focar nas tentativas de modelar e prever os mercados, e também das teorias modernas de portfólios.

2.3.1. Mercados Financeiros e a Economia

O que move os mercados? Há vários fatores que influenciam as cotações do mercado: o ambiente econômico, a bolsa de valores, as taxas de juros, entre outros. Estes são fatores de suma importância, mas há outros, que não podem ser negligenciados – assim, há que se levar em conta, por exemplo, os preços de produtos industrializados, como carros e máquinas. Estes, por sua vez, são determinados pelos preços de materiais básicos, como aço e plásticos – e as oscilações dos preços desses materiais são portanto importantes para a determinação dos lucros das empresas, e conseqüentemente, para os preços de suas ações. Assim, é possível dizer que de certo modo há um ciclo de realimentação dinâmica, que ajuda a determinar os preços dos ativos no mercado.

Contudo, essa observação não significa que os impactos são diretos, nem muito menos imediatos – os efeitos muitas vezes levam meses para serem sentidos e exercerem impacto, com uma variedade enorme de *lags* e atrasos impossivelmente difíceis de mensurar. Há um considerável *gap* portanto entre os fatos econômicos e os dados de alta freqüência dos mercados (o preço a preço, ou *tick* por *tick*[5]).

Portanto, é tarefa bastante difícil montar modelos *ab initio* dos efeitos de quaisquer combinações de variáveis econômicas sobre os preços dos mercados –

[4] Atribui-se ao famoso economista John Maynard Keynes a seguinte frase: "A sociedade normalmente considera que, para seu próprio bem, os jogos de cassino devem ser restritos a poucos. Pois bem, com os mercados financeiros deveria ser igual" (apud Bass, 1999).

[5] Um *tick* indica um negócio no mercado. Ele representa a combinação de oferta e demanda no mercado financeiro: se há mais oferta que demanda, o preço cai, logo ocorre um *downtick*, e vice-versa.

justamente porque é tão difícil mensurar a dinâmica desses efeitos. Além disso, deve-se levar em conta o efeito da psicologia dos investidores – vários trabalhos acadêmicos têm sido publicados nesse campo, e costumeiramente os mercados financeiros operam com um comportamento típico de manada, onde o sentimento geral sobre a direção dos preços pode ser muito mais importante que os fatores econômicos que os determinam[6].

2.3.2. O Mercado Brasileiro

O mercado financeiro no Brasil segue, em linhas gerais, os acontecimentos do mercado internacional. No aspecto regulatório e institucional, o tempo entre os acontecimentos nas Bolsas do Primeiro Mundo e nas brasileiras tende a ser maior, mas essa distância vem se encurtando nos últimos anos.

Em linhas gerais, as principais instituições do mercado financeiro brasileiro são[7]:

- *Governo*: o Banco Central do Brasil e a Comissão de Valores Mobiliários, que determinam o arcabouço jurídico-regulatório do mercado.
- *Bolsas*: até alguns anos atrás, o mercado de ações brasileiro era bipolar: havia a Bovespa (Bolsa de Valores de São Paulo) e a BVRJ, sua equivalente carioca. Até o fim dos anos 80, a bolsa do Rio de Janeiro predominava, até o advento do escândalo Naji Nahas, que iniciou um longo processo de declínio da mesma, até chegar aos dias de hoje, onde apenas a Bovespa tem relevância. Existe ainda a BM&F (Bolsa de Mercadorias e Futuros), onde são negociados a maioria dos contratos derivativos no país.
- *Instituições Financeiras*: de um modo geral, o mercado brasileiro tem três tipos de *players*: corretoras de valores, bancos (tanto de atacado quanto de varejo), e fundos de investimento. Muitas vezes, empresas congregam em suas estruturas os três "tipos" de instituição, mas de um modo geral é possível separar suas atividades e propósitos. Os fundos de investimento, em descrição simples, buscam aplicar rentavelmente o dinheiro captado de seus clientes, ganhando para isso taxas de

[6] Para uma introdução a esses trabalhos de psicologia dos mercados, ver Shiller, R. *Irrational Exuberance*, Nova York: Broadway Books, 2001. Esse campo de estudos em finanças é conhecido como "Finanças Comportamentais" (*Behavioral Finance* em inglês).

[7] Para detalhes e explicações mais aprofundadas, além de um histórico completo do desenvolvimento do mercado financeiro brasileiro, ver Fortuna (2002).

administração e/ou performance. Os bancos, tanto de atacado quanto de varejo, captam dinheiro de clientes e tentam rentabilizá-lo, ficando com a diferença conseguida. Já as corretoras são como que os intermediários de toda a estrutura: recebem e executam ordens de operação, recebendo para isso uma remuneração pelo serviço. A interação contínua e dinâmica desses *players*, junto da atuação governamental, e do constante fluxo de informações cotidianas, gera os movimentos de preços dos ativos negociados nos mercados (notadamente nas Bolsas, mas não só – há, por exemplo, o importantíssimo mercado interbancário, onde os bancos negociam entre si).

2.3.3. A Hipótese dos Mercados Eficientes

As tentativas de prever o mercado são praticamente tão antigas quanto o próprio mercado. Há mais de uma geração, pelo menos, a questão da eficiência dos mercados tem sido um dos pontos de mais interesse na área, e foco de intensa pesquisa. Ela tem sido debatida e testada basicamente na forma de duas teorias predominantes: a hipótese do *caminho aleatório*[8] e a *Hipótese dos Mercados Eficientes*[9]. Tendo isto em vista, examinar esta questão é de fundamental importância para o trabalho aqui apresentado, levantando as principais descobertas e argumentos teóricos, afinal, se é sabido *a priori* que não é possível prever os mercados, todo o trabalho de fazê-lo cai por terra.

Por mais de quarenta anos, pesquisadores têm deliberado acerca da natureza preditiva dos mercados financeiros, e proposto teorias para explicar os mecanismos subjacentes que determinam a dinâmica dos preços dos ativos financeiros. A pesquisa normalmente se foca na questão da previsibilidade dos retornos dos ativos, com a motivação vindo de um interesse notadamente econômico em entender como flutuações na economia influenciam os mercados financeiros e também com o

[8] Em inglês, conhecida como *random walk*. Formalmente introduzida por Paul Samuelson em 1965 no artigo "Proof that properly antecipated prices fluctuate randomly" *in Industrial Management Review 6*, pp. 41-49.

[9] Em inglês, referida como *Efficient Market Hypothesis*, ou simplesmente EMH. Formalmente introduzida por Eugene Fama e colaboradores em 1969, no artigo "The adjustment of stock prices to new information" *in International Economic Review 10*, pp. 1-21. Para uma visão completa dos argumentos teóricos e empíricos em favor da hipótese, vide Malkiel (2003).

interesse prático de se conseguir maneiras de melhorar os retornos obtidos com investimentos.

Mesmo após esse período de intensos debates e discussões, não é possível dizer que há consenso entre os acadêmicos, sobre o que explica o comportamento dos preços dos ativos financeiros. A visão dominante (pode-se dizer ortodoxa), é resumida em duas teorias inter-relacionadas, quais sejam, a Hipótese dos Mercados Eficientes, e a teoria do caminho aleatório. Esta assume que os preços são completamente estocásticos por natureza, enquanto aquela implica na ausência de oportunidades de lucro em mercados perfeitamente eficientes. Em essência, ambas as teorias implicam que, em mercados funcionando bem, os preços são inerentemente imprevisíveis e refletem plenamente toda a informação disponível no momento. A versão mais prática (mas também mais fraca) da hipótese diz que os preços refletem toda a informação disponível, até o ponto em que o lucro marginal de se operar usando uma informação é igual ao custo marginal de obtenção daquela informação. Isso quer dizer que em um mercado racional, onde a informação é livremente disponível e corretamente interpretada, os preços não deveriam se desviar da informação fundamental sobre a economia e o mercado.

Nos últimos dez anos, contudo, argumentos teóricos e pesquisas empíricas têm questionado seriamente ambas teorias, embora não haja, ainda, concordância ampla acerca da validade (ou não) das mesmas[10].

2.4. Modelagem e Previsão

Dado que existe algum grau de evidência de que os mercados financeiros são, até certo ponto, previsíveis, quaisquer desvios do modelo de "caminho aleatório" que os retornos de ativos apresentem podem fornecer a base teórica necessária para a construção de expectativas sobre esses retornos no futuro. Fazendo esta hipótese, as estimativas ingênuas, baseadas puramente em estatísticas do passado, podem ser substituídas por estimativas mais precisas, condicionadas mais fortemente nos dados

[10] Para um apanhado de tais argumentos, vide Shadbolt e Taylor (Eds.) (2003), pp. 24-33, e Campbell et. al (1997) para uma perspectiva mais aprofundada dos testes econométricos usados para provar (ou não) a Hipótese dos Mercados Eficientes. Para uma perspectiva de teoria do caos da questão – que tenta provar que os mercados são previsíveis, vide Sornette (2003).

recentes ou em conjuntos de dados fundamentais, que influenciem decisivamente tais retornos. Assim, expectativas não condicionadas dos retornos e riscos futuros são substituídas por modelos de previsão condicionados a um determinado conjunto de variáveis (o que implica tomar uma perspectiva bayesiana, por assim dizer, do problema de previsão – toda a previsão é condicionada a um determinado conjunto de informações que se opta por usar).

Uma enorme variedade de técnicas de modelagem e previsão podem ser aplicadas à tarefa de determinar a dinâmica futura dos retornos de ativos. A natureza altamente estocástica desses retornos, e o pequeno entendimento da dinâmica que guia os mesmos, levou ao desenvolvimento de técnicas cada vez mais complexas, que tentam capturar as influências não-lineares e variantes no tempo que podem estar presentes nas séries temporais de ativos financeiros.

Inicialmente, modelos lineares de previsão podem se provar uma ferramenta poderosa para a compreensão dos retornos de ativos. A gama de modelos à disposição do estudioso vai desde simples médias móveis, até modelos mais complexos, como técnicas ARIMA, métodos de regressão, cointegração e modelos de correção de erro, até técnicas no estado-da-arte, como modelos de espaço de estado[11]. Cada um desses modelos tem seus pontos fortes e fracos, tendo sido aplicados em maior ou menor grau ao problema de modelagem do mercado.

A evolução natural dos modelos lineares é a aplicação de modelos não-lineares. Novamente, a gama de modelos é vasta, começando nos modelos mais simples de regressão não-linear, e avançando por toda uma série de técnicas de fronteira, como teoria do caos e teoria de catástrofes,[12] redes neurais, algoritmos genéticos[13], lógica *fuzzy*, modelos de simulação – onde mercados virtuais são simulados computacionalmente na tentativa de capturar comportamentos do mercado real[14], e também misturas, combinações e "comitês" de modelos.

[11] Para um apanhado geral desses modelos, com indicação de farta bibliografia, e delineamento de aplicações em modelagem e previsão do mercado financeiro, ver Shadbolt e Taylor (Eds.) (2003), cap. 9, pp. 69-76.

[12] Para uma aplicação de teoria do caos e teoria de catástrofes à previsão do mercado, vide Sornette (2003).

[13] Vide capítulo 5 para detalhes.

[14] Para detalhes de como isso é feito, e resultados preliminares, vide Shadbolt e Taylor (Eds.) (2003), cap. 26, pp.247-258.

Além das técnicas de cunho quantitativo (basicamente de natureza estatística, econométrica), existem duas outras perspectivas consagradas para a tentativa de prever os preços de ativos. São elas as chamadas "análise fundamentalista" e "análise técnica". A primeira, também conhecida pelo nome *valuation*, parte do princípio de que os preços de ações refletem os fluxos de lucros projetados de uma empresa, descontados a uma taxa apropriada. Assim, ao analisar e projetar os balanços futuros da empresa, teoricamente seria possível determinar o valor "justo" das ações de uma empresa, e assim determinar se estão sub ou supervalorizadas[15]. Já a segunda técnica, também conhecida como "análise gráfica", parte do princípio de que os preços refletem os fluxos de compradores e vendedores no mercado, e assim é possível inferir tendências a partir da observação de gráficos de preços das ações.

De um modo geral, é possível dizer que o campo de estudo da modelagem e previsão dos mercados financeiros é ainda um campo aberto. Não há paradigmas estabelecidos, e dificilmente é possível afirmar com alguma dose de certeza que há uma maneira ótima de prever o comportamento futuro dos preços. Além disso, dado o dinamismo do mercado, o modelo "quasi-ótimo" de hoje pode ser inútil amanhã. Nesse contexto, talvez seja útil se referir, dada a discussão da seção 2.3.3, à versão de Lo e MacKinlay para a Hipótese dos Mercados Eficientes: "não é possível obter, sistematicamente, retornos em excesso do mercado, sem algum tipo de vantagem competitiva (por exemplo, tecnologia superior, informação proprietária, métodos avançados), sobre outros participantes do mercado" (apud Shadbolt e Taylor (Eds.), 2003). Em resumo, lucros acima do mercado são a remuneração marginal do esforço intelectual de desenvolvimento dos modelos e técnicas que os geraram.

2.5. Moderna Teoria de Portfólio[16]

Nesta seção apresenta-se brevemente a teoria moderna de carteiras de investimento, descrevendo-se: os fundamentos estatísticos para a formação de

[15] A técnica de análise fundamentalista foi desenvolvida pelo americano Benjamin Graham. Seu texto mais famoso se chama *Security Analysis*. Os resultados do investidor Warren Buffet (que foi aluno de Graham), o segundo homem mais rico do mundo, costumam ser o melhor marketing dos que advogam seu uso. Vale dizer que o autor trabalhou, durante seis meses, como analista de ações, fazendo justamente análise de balanço e *valuation* de empresas cotadas na Bovespa.

[16] A seção 2.5 é baseada em Gruber et al (2003), caps. 4-16, e Lazo Lazo (2000), cap. 2.

carteiras (proposta inicialmente por Harry Markowitz em 1959), os diferentes modelos daí derivados, propostos para descrever o comportamento do mercado (como o *Capital Asset Pricing Model* e o *Arbitrage Pricing Theory*).

2.5.1. Markowitz e a Média-Variância

Harry Markowitz formula a teoria de investimento como um problema de maximização de utilidade do investidor, sob condições de incerteza, a partir das preferências descritas por Von Neumann e Morgenstern. Inicialmente, foi estudado apenas o caso especial em que as preferências do investidor podem ser definidas pelos dois primeiros momentos da distribuição do retorno de uma carteira em um único período, constituindo o critério de Média-Variância.

Markowitz reconheceu o retorno como uma variável aleatória com distribuição normal, ou seja, completamente descrita pelos seus dois primeiros momentos: média e variância. Supõe-se que existe apenas um intervalo de tempo, ou seja, o mundo é uniperiódico, que a estrutura de covariâncias (matriz de Variâncias-Covariâncias) é constante e que o investidor é avesso ao risco. Pode-se definir que um ativo i é dominante em relação a um ativo j , pelo critério de Média-Variância, se:

$$\begin{aligned} E[R_i] &\geq E[R_j] \\ Var[R_i] &\leq Var[R_j] \end{aligned} \tag{2.1}$$

onde E[R] e Var[R] são o valor esperado e a variância da variável R respectivamente; logo, um ativo i é dominante ou preferível a um ativo j se a expectativa de retorno (média) é maior (ou igual) que a expectativa de retorno j e a variância é menor (ou igual) que a variância de j. Note-se que a variância é o indicador de risco. Para qualquer par de ativos cujas médias e variâncias guardem relações diferentes das contempladas na equação 2.1, o critério de Média–Variância concede-lhes a condição de indiferentes. O critério de Média–Variância é uma ótima regra de decisão quando se assume uma distribuição normal para os retornos da carteira.

Assim, o critério de Média–Variância emprega a média dos retornos passados para estimar o retorno futuro de um ativo e o risco é representado pela variância. Usando a notação de somatório, temos que o valor esperado, dados M retornos com

igual probabilidade para um ativo i, é expresso pela equação 2.2. Para resultados com probabilidades diferentes, sendo P_{ij} a probabilidade de retorno *j* de um ativo *i*, o valor esperado é representado pela equação 2.3.

$$\overline{R_i} = \sum_{j=1}^{M} \frac{R_{ij}}{M} \tag{2.2}$$

$$\overline{R_i} = \sum_{j=1}^{M} P_{ij} \cdot R_{ij} \tag{2.3}$$

A variância é usada como uma medida de risco, pois representa o grau de dispersão dos retornos. A equação 2.4 mostra a variância dos retornos do ativo i para retornos com igual probabilidade e a equação 2.5 mostra a variância se as observações não tiveram igual probabilidade.

$$\sigma_i^2 = \sum_{j=1}^{M} \left(\frac{(R_{ij} - \overline{R_i})^2}{M - 1} \right) \tag{2.4}$$

$$\sigma_i^2 = \sum_{j=1}^{M} P_{ij} \cdot (R_{ij} - \overline{R_i})^2 \tag{2.5}$$

Como o investidor tem muitas opções disponíveis e não só a escolha entre um ou outro ativo, ele pode investir apenas parte de seu dinheiro em cada ativo. Na construção de carteiras considera-se a combinação de ativos, assim, o risco da combinação de ativos é diferente do risco individual ou da média do risco individual de cada ativo. Portanto, a variância de uma combinação de dois ou mais ativos pode ser menor do que a variância de qualquer um dos ativos.

O retorno de uma carteira de ativos é um somatório dos pesos vezes o retorno de cada ativo. O peso aplicado para cada retorno é a fração da carteira investida nesse ativo. Se R_{pj} é o j-ésimo retorno na carteira e X_i é a fração do fundos investidos no i-ésimo ativo, temos (equação 2.6).

$$R_{pj} = \sum_{i=1}^{N} X_i R_{ij} \tag{2.6}$$

A expectativa de retorno é também uma média ponderada da expectativa de retorno dos ativos individuais e denomina-se Retorno Médio (equação 2.7):

$$\overline{R_P} = \sum_{i=1}^{N} X_i \cdot \overline{R_i} \tag{2.7}$$

O risco de uma carteira é estimado pela variância da carteira, designada por que é o valor esperado do quadrado dos desvios do retorno da carteira, em relação à média de retorno da carteira (equação 2.8); para o caso de uma carteira com dois ativos a variância é dada pela equação 2.9.

$$\sigma_P^2 = E(R_P - \overline{R_P})^2 \tag{2.8}$$

$$\sigma_P^2 = X_1^2\sigma_1^2 + X_2^2\sigma_2^2 + 2X_1X_2\sigma_{12} \tag{2.9}$$

Onde é a covariância que mede o comportamento dos retornos dos ativos ao moverem-se juntos. Para muitos propósitos utiliza-se o coeficiente de correlação, definido como a covariância dividida pelo produto do desvio padrão de cada ativo (equação 2.10), que varia no intervalo de –1 a +1.

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i\sigma_j} \tag{2.10}$$

Pode-se generalizar a equação da variância para mais de dois ativos considerando a covariância (equação 2.11) ou o coeficiente de correlação (equação 2.12):

$$\sigma_P^2 = \sum_{i=1}^{N} X_i^2\sigma_i^2 + \sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} X_iX_j\sigma_{ij} \tag{2.11}$$

$$\sigma_P^2 = \sum_{i=1}^{N} X_i^2\sigma_i^2 + \sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} X_iX_j\sigma_i\sigma_j\rho_{ij} \tag{2.12}$$

Para procurar uma carteira eficiente devem ser observadas as características de risco e retorno das combinações de ativos, com a ajuda de uma interpretação geométrica das mesmas. Esta geometria usada é denominada curva da fronteira eficiente, que representa a relação risco-retorno de uma carteira, para cada combinação de possíveis pesos de cada ativo. Para construir tal curva, o valor do coeficiente de correlação é de suma importância, como visto na equação 2.12.

O valor do coeficiente de correlação varia entre –1 a +1. Assim, um valor de +1 indica que os ativos movimentam-se em uníssono, indicando que o risco e o retorno da carteira são uma combinação linear do risco e retorno de cada ativo individual que a componha, isto é, que para maior retorno se tem maior risco. Neste caso de perfeita correlação entre os ativos da carteira, o retorno e o risco da carteira nada mais são do que a média ponderada do retorno e risco de cada ativo individual, ou seja, não há benefícios de diversificação.

Por outro lado, um valor do coeficiente de correlação de –1 indica que os comportamentos dos ativos são extremadamente opostos. Neste caso o risco da carteira é sempre pequeno em comparação com o caso anterior onde o valor da correlação foi +1 (para uma carteira de dois ativos é possível obter risco zero). Na prática, não existem ativos cuja correlação é perfeitamente positiva ou negativa. Na Figura 1 a seguir são apresentadas algumas curvas de fronteira eficiente, para diferentes valores do coeficiente de correlação.

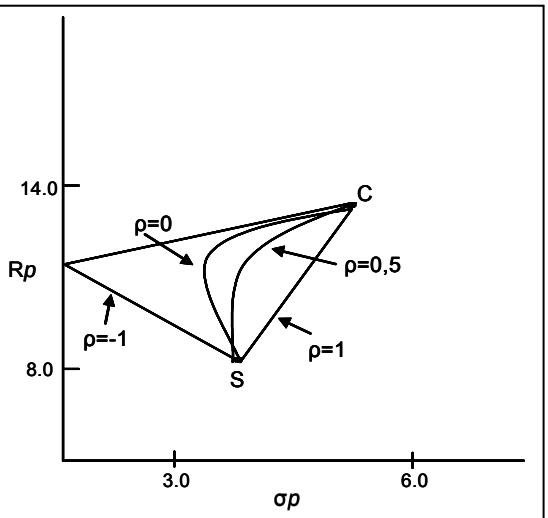


Figura 1 – Curvas de Fronteira Eficiente para várias Correlações (Lazo Lazo, 2000)

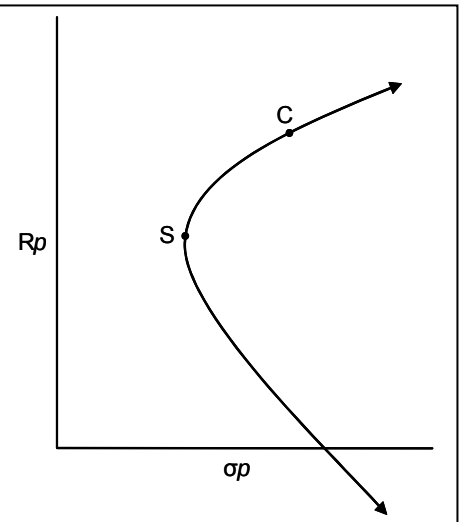


Figura 2 – Fronteira Eficiente de uma Carteira (Gruber et. al, 2003)

Na Figura 2 mostra-se uma curva de fronteira eficiente de uma carteira formada pela combinação de ativos, considerando que o investidor prefere o maior retorno com o menor risco possível. *"Claramente pode-se observar que a carteira eficiente deverá estar em algum ponto da parte superior esquerda da curva de fronteira eficiente, indicada pelos pontos S-C; esta parte da curva é a que apresenta os maiores retornos com menor risco"* (Gruber et al., 2003).

Já para o investidor que prefere qualquer retorno com o menor risco possível, isto é, o investidor deseja fundamentalmente minimizar o risco, esperando só algum retorno, a carteira eficiente está no ponto S da fronteira eficiente. Dado que o investidor, ao investir em uma carteira, procura ter um retorno maior que o obtido por um investimento em renda fixa, na formação da fronteira eficiente introduz-se o ativo livre de risco para indicar a procura de retornos superiores aos de renda fixa. R_F é o retorno do ativo livre de risco (renda fixa), onde o desvio padrão destes fundos é

zero, ou seja, o risco é zero. Logo, assume-se que, para a carteira A, a fração original de fundos que o investidor coloca na carteira é *X*, e (1-*X*) é a fração de fundos colocados no ativo livre de risco. A expectativa de retorno e o risco da combinação são dados pelas equações 2.13 e 2.14, respectivamente:

$$\overline{R_C} = (1 - X)R_F + X\overline{R_A} \tag{2.13}$$

$$\sigma_C = \left[(1 - X)^2 \sigma_F^2 + X^2 \sigma_A^2 + 2X(1 - X)\sigma_A \sigma_F \rho_{AF}\right]^{1/2} \tag{2.14}$$

Mas, como, por definição, $\sigma_F = 0$ (risco zero – ativo livre de risco), tem-se as equações 2.15 e 2.16):

$$\sigma_c = \left(X^2 \sigma_A^2\right)^{1/2} = X\sigma_A \tag{2.15}$$

$$X = \frac{\sigma_C}{\sigma_A} \tag{2.16}$$

Substituindo na expressão para a expectativa de retorno da combinação, tem-se a equação 2.17 que representa uma linha reta, onde a tendência da linha é $(R_A - R_F)/\sigma_A$ e a interseção com o eixo de retorno é R_F.

$$\overline{R}_C = R_F + \left(\frac{\overline{R_A} - R_F}{\sigma_A}\right) \cdot \sigma_C \tag{2.17}$$

Esta linha reta pode ser deslocada até o ponto de tangência (B), assim, tem-se o maior retorno para algum risco, como mostra a Figura 3 a seguir.

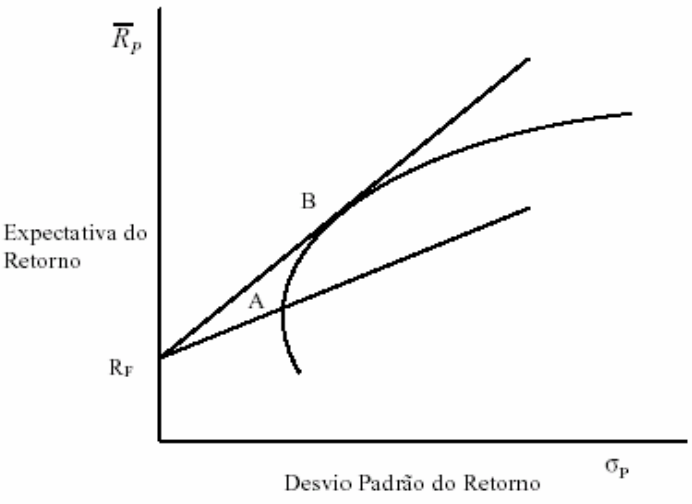


Figura 3 – Combinação de Ativos de menor risco em uma carteira (Gruber et al., 2003 – reelaborado pelo autor)

Para o caso da figura anterior, a carteira sobre o raio R_F-*B* é preferida a todas as outras carteiras, pelo que se denomina Carteira Eficiente, e a fronteira eficiente é a longitude inteira do raio estendido desde R_F até *B*. Para o cálculo deste ponto é suficiente maximizar a função (equação 2.18):

$$\theta = \frac{\sum_{i=1}^{N} X_i (\overline{R_i} - R_F)}{\left[\sum_{i=1}^{N} X_i^2 \sigma_i^2 + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} X_i X_j \sigma_{ij} \right]^{1/2}} \tag{2.18}$$

Para o investidor que deseja obter um retorno de modo que o risco seja mínimo, o objetivo é minimizar a equação do risco total da carteira (ver equações 2.11 ou 2.12), dado como restrição um determinado nível de retorno esperado (esse é um problema típico de Pesquisa Operacional, otimização não-linear).

2.5.2. Sharpe e o *CAPM*

O Capital Asset Pricing Model (CAPM) proposto por William Sharpe em 1964, baseia-se no paradigma da Média-Variância originado por Markowitz. Partindo da equação do modelo Diagonal, equação 2.19:

$$r_i = \alpha_i + \beta_i r_M + e_i \tag{2.19}$$

e aplicando o operador variância nos dois lados da equação acima, obtém-se a seguinte relação (equação 2.20):

$$\sigma_i^2 = \beta_i^2 \sigma_M^2 + \sigma_{e_i}^2 \tag{2.20}$$

A equação 2.20 mostra que o risco total do ativo, ou sua variância, é composto por duas parcelas. A primeira delas, o produto do quadrado do beta (uma definição de beta será dada mais adiante) e a variância do retorno de mercado, dita risco sistemático, de mercado ou não diversificável, reflete a correlação entre os retornos do ativo e do mercado; portanto, não pode ser diversificada. A segunda parcela, variância de resíduo, representa todos os efeitos dos demais fatores além do retorno de mercado. Nota-se que este risco, dentro de uma carteira, pode ser cancelado por movimento de direção contrária de outro ativo. Denomina-se risco próprio, não

sistemático ou diversificável. Portanto, o risco total de qualquer ativo é a soma do risco de mercado com o risco próprio.

Como o beta é o único fator característico do ativo na parcela do risco sistemático (a variância do retorno de mercado é igual para todos os ativos), este pode ser encarado como uma medida do risco sistemático do ativo. Assim, a maior mensagem do CAPM é que apenas o risco não-diversificável é recompensado em termos de excesso de retorno, já que o risco próprio pode ser diversificável dentro de uma carteira.

Além das hipóteses de Markowitz, o CAPM aceita outras relativas ao comportamento do consumidor e ao funcionamento e competitividade do mercado:

- Investidores são racionais e maximizadores da utilidade da riqueza final;
- Investidores são avessos ao risco e diversificam seus portfólios segundo o critério Média-Variância;
- Existe uma taxa livre de risco, r_f , igual para tomadores e investidores em qualquer quantidade de capital;
- Não existem taxas nem custos de transação e o custo de falência é desconsiderado;
- A informação está gratuitamente à disposição de todos os investidores;
- Todos os ativos são divisíveis e têm liquidez;
- Vendas a descoberto são ilimitadas;
- A quantidade de cada ativo no mercado é pré-fixada;
- O mercado é competitivo de forma que os investidores não acreditam que possam influenciar as cotações, ou seja, as decisões de um investidor qualquer não influenciam os preços.

Sob todas estas hipóteses, o CAPM indica o retorno que o investidor deveria esperar de um ativo dado seu risco não-diversificável (beta). Assim, sugere-se uma relação de funcionalidade linear entre o retorno esperado e o beta (equação. 2.21), onde o coeficiente angular é o prêmio de risco.

$$r_i = r_F + \beta_i \cdot (r_M - r_F) \tag{2.21}$$

Esta relação para os ativos do mercado é vista, graficamente, na *Linha do Mercado de Capitais* (*LMC*). A interseção entre a LMC e o eixo dos retornos esperados é o retorno da renda fixa, e sua inclinação é a diferença entre o retorno

esperado do mercado e da renda fixa. Os ativos em equilíbrio, segundo a CAPM, estão na LMC. Ativos que estiverem abaixo da linha estão desvalorizados; aqueles acima, supervalorizados

2.5.3. *Arbitrage Pricing Theory*

Segundo o *Arbitrage Pricing Theory*, APT, toda a estrutura do retorno, suas componentes sistemática e particular, é definida a *priori*. Isto é possível através de uma estrutura linear de fatores exógenos. Propriedade fundamental e natural do APT é sua multiperiodicidade. O APT é um dos resultados imediatos da teoria de arbitragem e é baseado no Teorema Fundamental da Formação de Preços de Ativos. O teorema assegura a equivalência das três afirmativas a seguir: (i) ausência de arbitragem; (ii) existência de uma regra de formação de preços linear positiva; (iii) existência de uma demanda ótima para qualquer agente racional.

A elaboração teórica do APT é uma construção razoavelmente complexa[17]. Envolve a aplicação de conceitos de não-arbitragem, e algumas hipóteses sobre a distribuição dos fatores que compõe os preços dos ativos. É possível resumir o edifício teórico da teoria nas equações 2.22 e 2.23:

$$r_i = r_F + \beta_{i1}(\delta_1 - r_F) + ... + \beta_{iJ}(\delta_J - r_F) \tag{2.22}$$

$$\beta_{ij} = \frac{\text{cov}(r_i, \delta_j)}{\text{var}(\delta_j)} \tag{2.23}$$

A equação 2.22 é a generalização multifatorial do modelo do CAPM. Em termos simples, o que ela faz é atribuir a *J* fatores diferentes o retorno de um ativo, ao invés de apenas um fator, como fazia Sharpe. A equação 2.23 do mesmo modo, explica como se compõe cada um dos *J* betas da carteira.

Como o APT é uma generalização do CAPM, ele guarda relação com a eficiência segundo o critério de Média-Variância:

- O APT não faz nenhuma suposição acerca da distribuição empírica dos retornos, enquanto o CAPM impõe normalidade;

[17] Para um desenvolvimento completo e detalhado da teoria por trás do APT, ver Gruber et al. (2003), cap. 16, pp. 364-394. Esta seção é baseada nesta referência.

- O APT não faz nenhuma hipótese forte sobre a função utilidade do indivíduo (apenas prevê não-saciedade e aversão ao risco);
- O APT permite que o equilíbrio dos retornos dos ativos dependa de vários fatores, não apenas de um como o retorno do mercado no CAPM;
- O APT fornece uma estrutura de preços relativos de qualquer subconjunto de ativos. Assim, não é preciso utilizar todo o universo de ativos para testar a teoria;
- Não há regra especial para a carteira de mercado no APT. Já no CAPM exige-se que ela seja eficiente;
- O APT é facilmente estendido para uma estrutura multiperiódica; o CAPM não.

A mensagem central é que tanto a construção teórica, quanto os testes empíricos apontam o APT como o melhor modelo de formação de preços disponível[18].

2.6. Risco[19]

Risco é um dos conceitos mais fundamentais da história da humanidade. Está envolvido em literalmente todos os empreendimentos humanos, seja a descoberta das Américas, as Cruzadas ou a fundação da USP. Desde a Idade Média, seu estudo tem gerado profundas implicações para o desenvolvimento dos mercados, vindo, mais tarde, a se tornar talvez o seu paradigma maior[20].

O Risco pode ser definido como a volatilidade de resultados inesperados[21], normalmente relacionada ao valor de ativos ou passivos de interesse. As perdas podem ocorrer pela combinação de dois fatores: volatilidade da variável financeira-objeto e exposição a essa fonte de risco. Embora as empresas não tenham controle sobre a volatilidade das variáveis financeiras, elas podem ajustar suas exposições a tais riscos, através de derivativos, por exemplo. O valor do risco captura o efeito combinado da volatilidade com a exposição aos riscos financeiros.

[18] Para testes econométricos do modelo, ver Campbell et al. (1997).

[19] Esta seção é baseada em Jorion (1997), Gruber et al. (2003), Lazo Lazo (2000) e para os detalhes do modelo GARCH, em The Mathworks (1999).

[20] Para uma abordagem geral do estudo do risco, incluindo o histórico do seu desenvolvimento, ver Bernstein (1998). Há uma edição brasileira deste livro, chamada *Desafio aos Deuses.*

[21] Definição de Jorion (1997).

A medida de risco não é constante ao longo do tempo. Além disso, a variação do risco no decorrer do tempo poderia explicar o fato da distribuição empírica dos retornos não se ajustar, de maneira exata, a uma distribuição normal.

Em relação ao modelo normal, a distribuição real com freqüência contém mais observações na parte central e nas caudas. Essas caudas grossas podem ser explicadas através de dois pontos de vista alternativos: o primeiro argumenta que a distribuição real é estacionária e que realmente contém caudas grossas, caso em que uma aproximação normal é nitidamente inadequada; o segundo, diz que a distribuição realmente muda com o tempo. Por conseguinte, em períodos de turbulência, um modelo estacionário poderia considerar *outliers* as grandes observações, quando, de fato, são provenientes de uma distribuição com dispersão temporária menor.

Na prática, existe certa dose de verdade em ambas as explicações. Por conta disso, a previsão da volatilidade é particularmente frutífera para o gerenciamento de risco. Nesta seção atenta-se para as abordagens tradicionais baseadas na modelagem de séries temporais "paramétricas".

A estimativa da volatilidade acabou tendendo à aplicação de modelos que atribuem maior peso às informações mais recentes. O primeiro deles foi o heterocedástico autoregressivo generalizado (*Generalized Autoregressive Conditional Heteroskedastic* ou simplesmente *GARCH*), proposto por Engle e Bollerslev em 1982.

O modelo GARCH pressupõe que a variância dos retornos siga um processo previsível. A variância condicional depende da inovação mais recente e, também da variância condicional anterior. Define-se h_t como a variância condicional, usando-se as informações até o instante *t-1*, e r_{t-1} como o retorno do dia anterior. O modelo mais simples desse tipo é o processo GARCH (1,1):

$$h_t = \alpha_0 + \alpha_1 r_{t-1}^2 + \beta h_{t-1} \tag{2.24}$$

A média, ou variância incondicional, é encontrada estabelecendo-se:

$$E[r_{t-1}^2] = h_t = h_{t-1} = h \tag{2.25}$$

Resolvendo para *h*, encontra-se:

$$h = \frac{\alpha_0}{1 - \alpha_1 - \beta} \tag{2.26}$$

Para que esse modelo seja estacionário, a soma dos parâmetros deve ser menor que um. Essa soma também é denominada persistência.

A beleza desta especificação está no fato de fornecer um modelo parcimonioso, com poucos parâmetros, que parece adequar-se muito bem aos dados. Os modelos GARCH tornaram-se a base das análises de séries temporais dos mercados financeiros, que demonstram, sistematicamente, períodos de "aglutinação" de volatilidade. Literalmente centenas de trabalhos aplicaram os modelos GARCH a retornos de ações, de taxas de juro e de moedas. Econometristas também criaram muitas variantes para o modelo GARCH[22], com a maioria apresentando apenas melhorias marginais em relação ao original. A desvantagem dos modelos GARCH é sua não-linearidade. Os parâmetros devem ser estimados pela maximização da função de verossimilhança, que envolve otimização numérica. Vale dizer que o modelo também pode ser usado para calcular volatilidade sobre vários horizontes. Por razões de simplicidade e por estar fora do escopo deste trabalho, não se vai estender o modelo aqui.

2.7. Métricas de Desempenho

Mensurar desempenho vem da necessidade de comparação entre os fundos para que se possa identificar os superiores, segundo a variável de desempenho preestabelecida, e entre estes e outras opções de investimento.

A literatura[23] apresenta alguns parâmetros para medir o desempenho das carteiras, ressalte-se que a qualidade destes parâmetros está sob julgamento do critério Média-Variância. A seguir algumas destas medidas de desempenho das carteiras:

- *Por Retorno e Risco:* para medir o desempenho da carteira empregam-se duas dimensões: uma para mensurar o retorno - retorno médio do investimento - e a outra para medir o índice de risco associado com este retorno - variância. Para comparar o desempenho de duas carteiras, comparam-se o retorno médio e o risco com o retorno

[22] Para um apanhado geral desses modelos, ver Campbell et al. (1997).

[23] As referências básicas são Gruber et al. (2003) e Lazo Lazo (2000). Existem ainda outros índices, como o de Modigliani e o de Sortino. Contudo, sua aplicação é bastante restrita, e portanto optou-se por não incluí-los aqui.

médio e o risco de um índice geral (o IBOVESPA no caso brasileiro por exemplo) ou outra carteira formada pelos ativos do mercado (embora esta carteira seja sempre uma *proxy*). A equação 2.27 representa o retorno acumulado do fundo *i* no período *T*, e a equação 2.28 representa o retorno médio do fundo *i* no período.

$$r_{iT} = \prod_{i=1}^{T}(1 + r_{it}) - 1 \tag{2.27}$$

$$\bar{r}_{iT} = (1 + r_{iT})^{1/T} - 1 \tag{2.28}$$

➢ *Variância Total versus o Beta como Índice de Risco*: para comparar o desempenho de duas carteiras usando o Beta como índice de risco, este beta está definido pela equação 2.29):

$$\beta_i = \frac{Cov(R_i, R_m)}{\sigma_m^2} \tag{2.29}$$

Onde R_i e R_m são os retornos do ativo *i* e da carteira de mercado, respectivamente; σ_m^2 é a variância da carteira de mercado, a covariância é o fator de risco que distingue um ativo de menor risco de outro e mede o movimento do retorno *i* com o mercado.

➢ *Índice de Sharpe*: proposto por Sharpe, este índice é o coeficiente de variação do excesso de retorno durante o período T. O índice de Sharpe está representado pela equação 2.30:

$$IS = \frac{r_i - r_f}{\sigma_i} \tag{2.30}$$

Onde r_i é retorno do ativo ou carteira *i*; r_f é o retorno do ativo livre de risco e σ_i é o desvio padrão do retorno do ativo ou carteira *i*. A medida de Sharpe é extensão direta do critério de média-variância, por tanto, suas hipóteses são as mesmas deste critério.

➢ *Índice de Sharpe Generalizado*: Sharpe propôs a generalização de seu índice para o ambiente multidimensional dos modelos multifatoriais como o *APT*. No índice de Sharpe generalizado, o excesso de retorno, d_i, representa a diferença entre o retorno do fundo e o retorno de um ativo similar segundo um modelo de J fatores (equação 2.31):

$$d_i = \tilde{r}_i - \beta_{i1}(\tilde{F}_1 - \tilde{r}_f) - \beta_{i2}(\tilde{F}_2 - \tilde{r}_f) - \ldots - \beta_{iJ}(\tilde{F}_J - \tilde{r}_f) \quad (2.31)$$

Define-se, o índice de Sharpe generalizado como (equação 2.32):

$$ISg = \frac{d_i}{\sigma_{d_i}} \quad (2.32)$$

Gruber et al. (2003) afirmam que, se a estimação das sensibilidades do portfólio padrão (*benchmark)* é feita por análise de séries temporais de um modelo de múltiplos fatores, então o Índice Sharpe Generalizado torna-se a Taxa de Informação. Uma vez fixado o conjunto de fatores do modelo, o alfa e o índice de Sharpe generalizado identificam as mesmas carteiras como aquelas de desempenho superior.

- *Índice de Treynor*: Esta medida é similar ao índice de Sharpe, no sentido de medir o excesso de retorno por unidade de risco. No entanto Sharpe utiliza o desvio padrão do retorno da carteira como índice de risco; Treynor utiliza o beta característico da carteira, βi. Define-se, o índice de Treynor como (equação 2.33):

$$IT = \frac{r_i - r_f}{\beta_i} \quad (2.33)$$

Treynor aceita o CAPM e suas hipóteses, ao empregar o beta como medida de risco.

3. REDES NEURAIS[1]

"Qualquer tecnologia suficientemente avançada é indistinguível da mágica."
Arthur C. Clarke[2]

3.1. Introdução

O trabalho em redes neurais artificiais, normalmente chamadas apenas de redes neurais, tem sido motivado desde o começo pelo reconhecimento de que o cérebro processa informações de um modo inteiramente diferente do computador digital convencional. O cérebro é um "computador" (sistema de processamento de informação) altamente complexo, não-linear e paralelo. Ele tem a capacidade de organizar seus constituintes estruturais, chamados de neurônios, de forma a realizar determinados processamentos (tarefas como reconhecimento de padrões, percepção, controle motor) muito mais rapidamente que o mais rápido computador digital existente. A visão humana é um exemplo interessante: a função do sistema visual é fornecer uma representação do ambiente à nossa volta, e fornecer a informação necessária para interagir com este ambiente. Mais especificamente, o cérebro realiza rotineiramente tarefas de reconhecimento perceptivo (reconhecer um rosto familiar em uma cena não-familiar por exemplo) em aproximadamente 100-200 ms, ao passo que tarefas de complexidade muito menor podem levar dias para serem executadas em um computador convencional.

Como é possível que o cérebro humano faça isso? No momento do nascimento, um cérebro tem uma grande estrutura e a habilidade de desenvolver suas próprias regras através daquilo que é comumente denominado de "experiência". Na verdade, a experiência vai sendo acumulada com o tempo, sendo que o mais dramático desenvolvimento (i.e., por ligações físicas) do cérebro humano acontece durante os dois primeiros anos de vida; de todo modo, o desenvolvimento continua por muito mais tempo além disso.

[1] O presente capítulo é baseado em grande medida em Haykin (2001), cap. 1, pp.27-70.

[2] Apud Bass (2000).

Um neurônio em desenvolvimento é sinônimo de um cérebro plástico: a plasticidade permite que o sistema nervoso em desenvolvimento se adapte ao seu meio ambiente. Assim como o processamento de informação do cérebro humano, também ela o é com relação às redes neurais construídas com neurônios artificiais. Na sua forma mais geral, uma rede neural é uma máquina projetada para modelar a maneira como o cérebro realiza uma tarefa particular ou função de interesse; a rede é normalmente implementada usando-se componentes eletrônicos ou é simulada por programação em um computador digital. Para alcançarem bom desempenho, as redes neurais empregam uma interligação maciça de células computacionais simples denominadas neurônios, ou unidades de processamento. É possível então dar a seguinte definição de uma rede neural, vista como uma máquina adaptativa[3]: *Uma rede neural é um processador paralelo maciçamente distribuído, constituído de unidades de processamento simples. Assemelha-se ao cérebro em dois pontos: (i) o conhecimento do ambiente externo é incorporado à rede via um processo de aprendizagem e (ii) forças de conexão entre os neurônios são usadas para armazenar esse conhecimento.*

O método utilizado para realizar o processo de aprendizagem é chamado de algoritmo de aprendizagem, cuja função é modificar as forças de conexão (também chamadas de pesos sinápticos) da rede de forma ordenada para alcançar um determinado objetivo desejado.

A modificação dos pesos sinápticos é o procedimento tradicional para o projeto de redes neurais. Entretanto, é possível também para uma rede neural modificar sua própria topologia, o que é motivado pelo fato de os neurônios no cérebro humano poderem morrer e novas conexões poderem nascer[4].

É evidente que as redes neurais extraem seu poder computacional, primeiro, de sua estrutura paralela maciçamente distribuída e segundo, de sua habilidade de aprender e portanto de generalizar. A generalização se refere ao fato de a rede neural produzir saídas adequadas para entradas que não estavam presentes durante o treinamento (aprendizagem). Estas duas capacidades de processamento de informação tornam possível para as redes neurais resolver problemas complexos de

[3] Definição adaptada de Haykin (2001), que por sua vez é adaptada de Aleksander e Morton (1990).

[4] Essa é a distinção básica entre treinamento supervisionado e não-supervisionado, conforme se verá adiante.

grande escala, que são normalmente intratáveis por métodos mais tradicionais. Na prática, contudo, as redes neurais não são capazes de fornecer soluções trabalhando individualmente. Em vez disso, elas precisam ser integradas em uma abordagem consistente de engenharia: um problema complexo de interesse é decomposto em tarefas relativamente simples, e atribui-se às redes neurais as tarefas que coincidem com suas capacidades inerentes[5].

O uso de redes neurais oferece as seguintes propriedades úteis:

- *Não-linearidade*: um neurônio artificial pode ser linear ou não linear. Uma rede neural, constituída por conexões de neurônios não-lineares é ela mesma não-linear. Além disso, a não-linearidade é de um tipo especial, no sentido de ela ser distribuída por toda a rede. Esta é uma propriedade muito importante, particularmente se o mecanismo responsável pela geração do sinal de entrada for inerentemente não-linear[6].
- *Mapeamento de Entrada-Saída:* um paradigma popular de aprendizagem chamado *aprendizagem com professor* ou *aprendizagem supervisionada* envolve a modificação dos pesos sinápticos de uma rede neural pela aplicação de um conjunto de amostras de treinamento rotuladas ou exemplos da tarefa. Cada exemplo consiste de um sinal de entrada único e de uma resposta desejada correspondente. Apresenta-se para a rede um exemplo escolhido ao acaso do conjunto, e os pesos sinápticos (parâmetros livres) da rede são modificados para minimizar a diferença entre a resposta desejada e a resposta real da rede, produzida pelo sinal de entrada, de acordo com um critério estatístico apropriado. O treinamento da rede é reproduzido para muitos exemplos do conjunto, até que a rede alcance um estado estável onde não haja mais modificações significativas nos pesos sinápticos. Os exemplos de treinamento previamente aplicados podem ser reaplicados durante a sessão de treinamento, mas em ordem diferente. Assim, a rede aprende dos exemplos ao construir um mapeamento entrada-saída para o problema considerado. Há uma analogia próxima

[5] Como ficará claro ao longo deste trabalho, é justamente esta a perspectiva aqui adotada. As redes neurais serão usadas para resolver apenas uma parte do problema de gestão de portfólios, qual seja, a da previsão dos retornos dos ativos. Embora, como ficará claro no capítulo 5, elas sejam capazes de resolver outras partes (como a da otimização da carteira), não é o que se fará no presente trabalho.

[6] É possível dizer que as séries temporais de retornos de ativos financeiros são inerentemente não-lineares. Para uma formulação interessante do problema, vide Sornette (2003).

entre o processo de mapeamento entrada-saída e o conceito de inferência estatística não-paramétrica.

- *Adaptabilidade:* as redes neurais têm uma capacidade inata de adaptar seus pesos sinápticos a modificações do meio ambiente. Em particular, uma rede neural treinada para operar em um ambiente específico pode ser facilmente retreinada para lidar com pequenas modificações nas condições operativas do meio ambiente. Além disso, quando está em um ambiente não-estacionário (i.e.,onde as estatísticas mudam com o tempo), uma rede neural pode ser projetada para modificar os seus pesos sinápticos em tempo real. A arquitetura natural de uma rede neural para classificação de padrões, processamento de sinais e aplicações de controle, aliada à capacidade de adaptação da rede, a torna uma ferramenta muito útil para classificação adaptativa de padrões, processamento adaptativo de sinais e controle adaptativo. Como regra geral, pode-se dizer que quanto mais adaptativo se fizer um sistema, assegurando que ele permaneça estável, mais robusto será o seu desempenho quando o sistema for exigido a operar em um ambiente não-estacionário. Contudo, deve ser enfatizado, que adaptabilidade nem sempre resulta em robustez, e na verdade pode resultar no contrário. Um sistema adaptativo com constantes de tempo pequenas, por exemplo, pode se modificar rapidamente e assim tender a responder a perturbações espúrias, causando uma drástica degradação no desempenho do sistema. Para aproveitar todos os benefícios da adaptabilidade, as constantes de tempo principais do sistema devem ser grandes o suficiente par que o sistema ignore perturbações espúrias mas ainda assim pequenas o suficiente para responder a mudanças significativas no ambiente – esse *trade-off* normalmente é chamado de *dilema estabilidade-plasticidade*.
- *Resposta a Evidências*: no contexto de classificação de padrões, uma rede neural pode ser projetada para fornecer informação não somente sobre qual padrão particular selecionar, mas também sobre a confiança ou crença na decisão tomada.Esta última informação pode ser usada para rejeitar padrões ambíguos, caso eles estejam presentes, e com isso melhorar o desempenho de classificação da rede.
- *Informação Contextual*: o conhecimento é representado pela própria estrutura e estado de ativação da rede neural. Cada neurônio da rede é potencialmente afetado pela atividade de todos os outros neurônios da rede. Conseqüentemente, a informação contextual é tratada naturalmente pela rede.

- *Tolerância a Falhas*: uma rede neural tem o potencial de ser inerentemente tolerante a falhas, ou capaz de realizar computação robusta, no sentido de que seu desempenho se degrada suavemente sob condições de operação adversas. Se um neurônio ou suas conexões é danificado, por exemplo, a recuperação de um padrão armazenado é prejudicada em qualidade. Contudo, devido à natureza distribuída da informação armazenada na rede, o dano deve ser extenso para que a resposta global da rede seja degradada seriamente. Assim, a principio, uma rede neural exibe uma degradação suave do desempenho em vez de apresentar uma falha catastrófica. Há algumas evidências para a computação robusta, mas geralmente ela não é controlada (Haykin, 2001). Para se assegurar que uma rede neural seja de fato tolerante é necessário adotar-se medidas corretivas no projeto do algoritmo utilizado para treinar a rede (Kerlirzin e Vallet, 1993).

3.2. Histórico

A era moderna das redes neurais começou com o trabalho pioneiro de McCulloch e Pitts (1943). McCulloch era um psiquiatra e neuroanatomista por treinamento, que passou 20 anos refletindo sobre a representação de um evento no sistema nervoso. Pitts era um matemático, que se associou a McCulloch em 1942. No se clássico artigo *"A logical calculus of the ideas immanent in nervous activity"*, eles descrevem um cálculo das redes neurais que unificava os estudos de neurofisiologia e lógica matemática. Eles assumiam que seu modelo formal de um neurônio seguia uma lei "tudo ou nada". Com um número suficiente dessas unidades simples e com conexões sinápticas ajustadas apropriadamente e operando de forma síncrona, os autores mostraram que uma rede assim constituída realizaria, a principio, a computação de qualquer função (computável). Este era um resultado muito significativo e com ele é geralmente aceito o nascimento das disciplinas de redes neurais e inteligência artificial.

O próximo desenvolvimento significativo das redes neurais veio em 1949, com a publicação do livro de Hebb *"The Organization of Behavior"*, no qual foi apresentada pela primeira vez uma formulação explicita de uma regra de aprendizagem fisiológica para a modificação sináptica. Especificamente, Hebb

propôs que a conectividade do cérebro é continuamente modificada conforme um organismo vai aprendendo tarefas funcionais diferentes e que agrupamentos neurais são criados por tais modificações. Além disso, também apresentou o "postulado da aprendizagem", que afirma que a eficiência de uma sinapse variável entre dois neurônios é aumentada pela ativação de um neurônio pelo outro, através daquela sinapse.

Ao longo dos anos seguintes muitos grandes nomes da computação deram contribuições ao estudo das redes neurais. Cientistas como Minsky[7], Gabor[8] e Von Neumann[9] trabalharam em aspectos do problema. No entanto o avanço mais significativo foi alcançado em 1958 por Rosenblatt, com o desenvolvimento do *perceptron*, hoje o modelo básico de neurônio artificial. Além disso, Rosenblatt também introduziu o famoso *teorema de convergência do perceptron*[10]. Em 1960 Widrow e Hoff introduziram o algoritmo *LMS* (*Least Mean-Square*, ou mínimos quadrados médios, um algoritmo de aprendizagem para redes neurais simples), e o estudo e desenvolvimento de redes neurais ganharam forte impulso.

Mas então veio o livro de Minsky e Papert[11], que utilizaram a matemática para demonstrar que existem limites fundamentais para aquilo que os perceptrons de camada única podem calcular. Em uma breve seção sobre perceptrons de múltiplas camadas, eles afirmavam que não havia razão para supor que qualquer uma das limitações do perceptron de camada única poderia ser superada na versão de múltiplas camadas. Foi um banho de água fria em um crescente campo de pesquisa. O problema básico encontrado no projeto de um perceptron de múltiplas camadas é o problema de *atribuição de crédito* (i.e., o problema de atribuir aos neurônios escondidos da rede o crédito pelo erro). A conjunção do livro de Minsky e Papert

[7] Seu artigo de 1961 *"Steps toward Artificial Intelligence"* contém uma grande seção sobre redes neurais.

[8] Mais conhecido como inventor da holografia, foi um dos pioneiros da teoria da comunicação e propôs a idéia do *filtro adaptativo não-linear*.

[9] Um dos pais da computação, realizou em 1957 as famosas Palestras Silliman, postumamente publicadas no livro *The Computer and the Brain* (1958).

[10] Esse teorema garante que um perceptron de camada única consegue, após um número finito de iterações de treinamento, classificar dois conjuntos de dados linearmente separáveis.

[11] *Perceptrons*, MIT Press, 1969, republicado em 1988.

com a ausência de uma idéia clara sobre como resolver esse problema fez com que as redes neurais ficassem adormecidas por mais de uma década.

Alguns desenvolvimentos relevantes aconteceram no fim dos anos 70 e inicio dos 80, como a criação dos *mapas auto-organizáveis* de Kohonen[12], e das *máquinas de Boltzmann*[13], mas foi apenas em 1986, com a publicação, por Rumelhart, Hinton e Williams do artigo "*Learning representations of back-propagation erros*"[14], que o estudo das redes neurais voltou a ganhar força. O trio conseguiu resolver definitivamente o problema de atribuição de crédito, deixando as objeções de Minsky e Papert para trás e o algoritmo de retropropagação emergiu como o paradigma básico para aprendizagem, sendo o mais popular e conhecido algoritmo para treinamento de redes de múltiplas camadas.

Em 1988, Broomhead e Lowe descreveram um procedimento para o projeto de redes alimentadas adiante, em camadas utilizando *funções de base radial*, que fornecem uma alternativa aos modelos de perceptrons de múltiplas camadas. No início dos anos 90, Vapnik e co-autores desenvolveram uma classe de redes de aprendizagem supervisionada poderosa do ponto de vista computacional, chamada de *máquinas de vetor de suporte*, para ser utilizada em reconhecimento de padrões, regressão e problemas de estimação.

Este é apenas um breve histórico dos principais desenvolvimentos no contexto das redes neurais. Em resumo, pode-se dizer que o trabalho pioneiro de McCulloch-Pitts e Rosenblatt constitui o cerne de uma primeira era das redes neurais, e o trabalho de Rumelhart e parceiros trouxe o ressurgimento do campo, para uma – ainda corrente – segunda era das redes neurais.

3.3. O Neurônio Artificial

Um *neurônio* é uma unidade de processamento de informação que é fundamental para a operação de uma rede neural. O diagrama em blocos da Figura 4 mostra o modelo de um neurônio, que forma a base para o projeto de redes neurais artificiais.

[12] Um tipo de rede neural com treinamento não-supervisionado.

[13] Redes inspiradas pela mecânica estatística, onde há aprendizado estocástico.

[14] Revista *Nature*, vol. 323, pp.533-536.

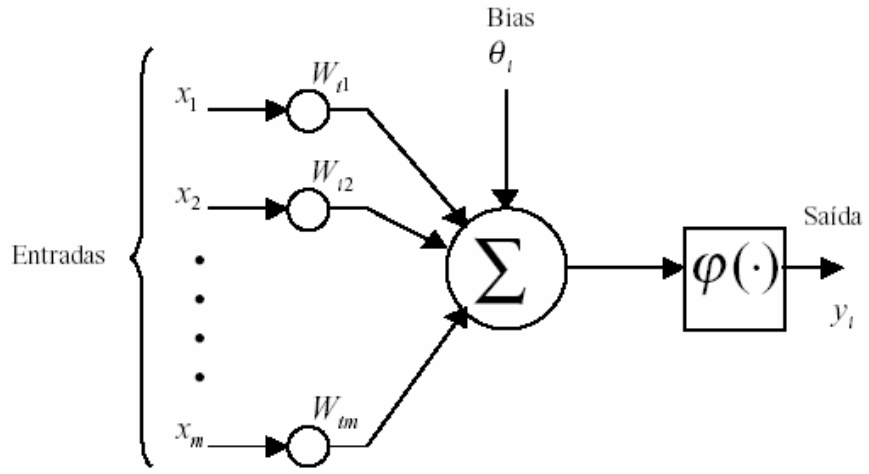


Figura 4: Modelo de um Neurônio Artificial (Haykin, 2000)

Há três elementos básicos no modelo neuronal:

- Um conjunto de *sinapses* ou elos de conexão, cada uma caracterizada por um *peso* próprio.Especificamente um sinal x_j na entrada da sinapse *j* conectada ao neurônio *k* é multiplicado pelo peso sináptico w_{kj}. É importante notar a maneira como são escritos os índices do peso sináptico w_{kj}. O primeiro índice se refere ao neurônio em questão e o segundo se refere ao terminal de entrada da sinapse ao qual o peso se refere. Ao contrário de uma sinapse do cérebro, o peso sináptico de um neurônio artificial pode estar em um intervalo que inclui valores negativos bem como positivos.
- Um *somador* para somar os sinais de entrada, ponderados pelas respectivas sinapses do neurônio; essas operações caracterizam um combinador linear.
- Uma *função de ativação* para restringir a amplitude da saída de um neurônio. A função de ativação é também referida como função restritiva já que restringe o intervalo permissível de amplitude do sinal de saída a um valor finito. Tipicamente o intervalo normalizado da amplitude da saída de um neurônio é escrito como o intervalo unitário fechado [0,1] ou alternativamente [-1,1].

O modelo neuronal da figura 4 inclui também um *bias* (viés) aplicado externamente, representado por b_k. O viés tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente.

Em termos matemáticos, podemos descrever um neurônio *k* escrevendo o seguinte par de equações:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3.1}$$

e,

$$y_k = \varphi(u_k + b_k) \tag{3.2}$$

onde, x_j são os sinais de entrada; w_{kj} são os pesos sinápticos do neurônio *k*; u_k é a saída do combinador linear devido aos sinais de entrada; b_k é o viés; φ é a função de ativação e y_k é o sinal de saída do neurônio. O uso do viés b_k tem o efeito de aplicar uma transformação afim à saída u_k do somador linear no modelo da figura 4, como mostrado por:

$$v_k = u_k + b_k \tag{3.3}$$

Em particular, dependendo se o viés b_k é negativo ou positivo, a relação entre o *campo local induzido* ou *potencial de ativação* v_k do neurônio *k* e a saída do combinador linear u_k é modificada: a partir desta transformação afim, o gráfico de v_k em função de u_k não passa mais pela origem.

O viés b_k é um parâmetro externo do neurônio artificial *k*. É possível considerar sua presença como na equação 3.2. Equivalentemente se podem formular as equações 3.1 até 3.3 como segue:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{3.4}$$

$$y_k = \varphi(v_k) \tag{3.5}$$

Na equação (3.4) se adicionou uma nova sinapse. A sua entrada e o seu peso são dados, respectivamente, por:

$$x_0 = +1 \tag{3.6}$$

$$w_{k0} = b_k \tag{3.7}$$

É possível, portanto, reformular o modelo do neurônio *k* (da Figura 4). onde o efeito do viés é levado em conta de duas maneiras: (i) adicionando-se um novo sinal de entrada fixo em +1 e (ii) adicionando-se um novo peso sináptico igual ao viés b_k.

3.3.1. Função de Ativação

A função de ativação, representada por $\varphi(v)$, define a saída de um neurônio em termos do campo local induzido *v*. Há três tipos básicos de função de ativação:

- *Função de Limiar*: para este tipo de função de ativação, tem-se:

$$\varphi(v) = \begin{cases} 1, se & v \geq 0 \\ 0, se & v < 0 \end{cases} \tag{3.8}$$

Na literatura de engenharia, esta forma de função de limiar é normalmente referida como *função de Heaviside* (Haykin, 2000). A saída do neurônio *k* que emprega esse tipo de função pode ser expressa como:

$$y_k = \begin{cases} 1 & se \quad v_k \geq 0 \\ 0 & se \quad v_k < 0 \end{cases} \tag{3.9}$$

onde v_k é o campo local induzido do neurônio. Tal neurônio é referido na literatura como *modelo de McCulloch-Pitts*, a partir do trabalho dos pioneiros de redes neurais (Haykin, 2000). Esse tipo de neurônio apresenta uma característica do tipo *tudo-ou-nada*.

- *Função Linear por Partes*: para este tipo de função de ativação, tem-se:

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{3.10}$$

onde assume-se que o fator de amplificação dentro da região linear de operação é a unidade. Esta forma de função de ativação pode ser vista como uma *aproximação* de um amplificador não-linear (Lazo Lazo, 2000). A função de limiar pode ser considerada um caso especial desta, onde o amplificador é tomado infinitamente grande.

- *Função Sigmóide*: a função sigmóide, cujo gráfico tem a forma de *S*, é de longe a forma mais comum de função de ativação utilizada na construção de redes neurais artificiais. Ela é definida como uma função estritamente crescente que exibe um balanceamento adequado entre comportamento linear e não-linear (Haykin, 2000). Um exemplo de função com essas características é a *função logística*, dada por:

$$\varphi(v) = \frac{1}{1+\exp(-av)} \tag{3.11}$$

onde *a* é o parâmetro de inclinação da função sigmóide. Uma característica importante para o desenvolvimento da teoria subjacente às redes neurais é a diferenciabilidade da função de ativação. Nesse aspecto, funções sigmóides são claramente melhores candidatas que funções de limiar ou lineares por partes (a necessidade da diferenciabilidade está ligada ao cálculo dos erros, que guia o processo de aprendizado de uma rede neural). Algumas vezes é desejável que a função de ativação se extenda de –1 a +1 (as anteriores vão de 0 a +1), ou seja, sejam anti-simétricas em relação a origem. Para a função sigmóide deste tipo, tem-se a *função tangente hiperbólica*:

$$\varphi(v) = \tanh(v) \tag{3.12}$$

De um modo geral, a função logística e a tangente hiperbólica são as mais usadas em aplicações reais de redes neurais artificiais.

3.3.2. Topologia de Rede

A maneira pela qual os neurônios de uma rede neural estão estruturados está intimamente ligada com o algoritmo de aprendizagem usado para treinar a rede. Assim, analisar a topologia de redes neurais sem estudar os algoritmos de aprendizagem pode não fazer muito sentido. Em seções subseqüentes, vai-se mostrar os principais tipos de aprendizagem, mas antes uma breve introdução aos tipos básicos de estruturas de rede é devida.

Em uma rede neural em *camadas*, os neurônios estão organizados na forma de camadas. Na forma mais simples de uma rede deste tipo, há uma camada de entrada de nós de fonte que se projeta sobre uma camada de saída de neurônios (nós computacionais), mas não vice-versa. Em outras palavras, é uma rede alimentada adiante (*feedforward*).

A segunda classe de rede neural alimentada adiante se distingue pela presença de uma ou mais *camadas ocultas*, cujos nós computacionais são chamados correspondentemente de *neurônios ocultos.* A função destes é intervir entre a entrada externa e a saída da rede de uma maneira útil. De um modo bastante geral, os

neurônios da camada escondida servem para armazenar o conhecimento contido nos padrões de treinamento, de maneira a que a rede seja capaz de generalizar quando apresentada a padrões desconhecidos.

Já uma rede neural *recorrente* se distingue pelo fato de ter pelo menos um laço de realimentação. Isso se refere a uma situação em que pelo menos uma saída da rede é realimentada para sua entrada. De um modo geral, os laços de realimentação servem para introduzir um componente de memória na rede neural, de maneira a que ela considere os resultados passados que produziu no seu processo de aprendizagem. As figuras 5 e 6, a seguir, ilustram os principais tipos de rede descritos.

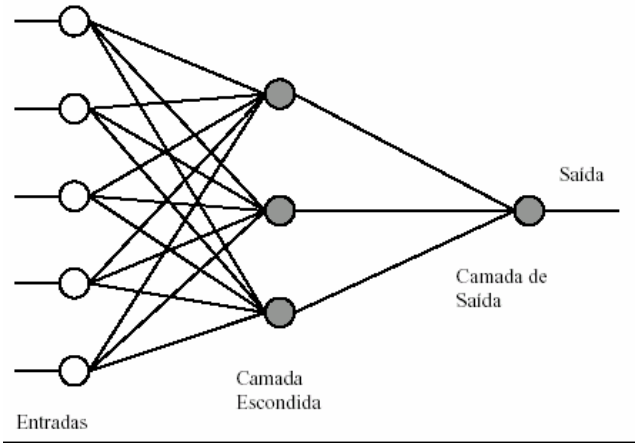


Figura 5: Exemplo de Rede Não-Recorrente com 1 camada escondida (Lazo Lazo, 2000)

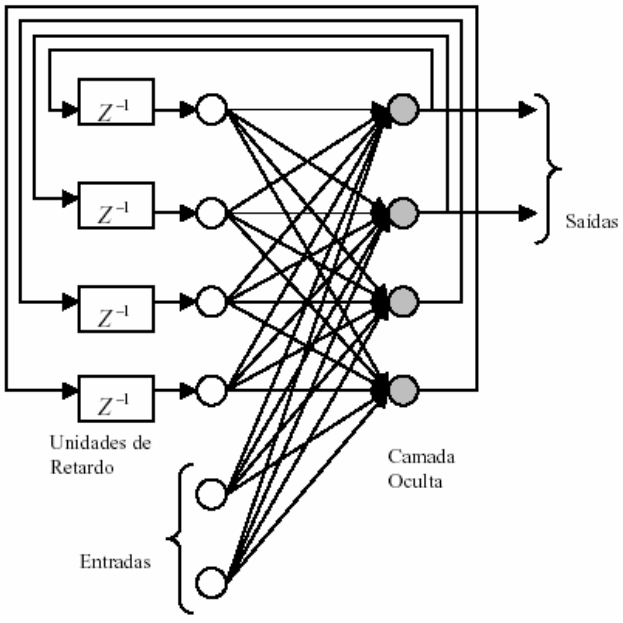


Figura 6: Exemplo de Rede Neural Recorrente (Haykin, 2000)

3.4. Processos de Aprendizagem

Aprendizagem, no sentido de redes neurais, é o processo de calcular os pesos sinápticos de uma rede. Os pesos, como já discutido, são um fator crucial, definindo o valor da saída de um neurônio, e portanto definindo qual o resultado que a rede obtém. Num sentido mais genérico, é possível dizer que os pesos "são" o conhecimento, já que todos os exemplos apresentados à rede são armazenados à medida que são apresentados durante o treinamento. Com a exceção do viés e de técnicas mais avançadas de poda de rede (explicadas mais adiante), os pesos são a característica básica da rede que é alterada ao longo do processo de aprendizagem. Há dois tipos básicos de processos de aprendizagem: a supervisionada e a não-supervisionada, que vão ser descritas a seguir.

3.4.1. Aprendizagem Supervisionada

A aprendizagem supervisionada necessita de um par de vetores composto da entrada e do vetor alvo que se deseja como saída. Juntos, estes vetores são chamados de par de treinamento ou vetor de treinamento, sendo que geralmente a rede é treinada com vários vetores de treinamento. O processo de aprendizagem é feito da seguinte maneira: o vetor de entrada é aplicado, a saída da rede é calculada e comparada com o correspondente vetor alvo. O erro encontrado é então realimentado através da rede e os pesos são atualizados de acordo com um algoritmo determinado a fim de minimizar este erro. Este processo de treinamento é repetido até que o erro, para todos os vetores de treinamento, tenha alcançado o nível especificado.

3.4.2. Aprendizagem Não-Supervisionada

A aprendizagem não-supervisionada não requer vetor alvo para as saídas. O conjunto de treinamento modifica os pesos da rede de forma a produzir saídas que sejam consistentes, isto é, tanto a apresentação de um dos vetores de treinamento, como a apresentação de um vetor que é suficientemente similar, irão produzir o

mesmo padrão nas saídas. O processo de treinamento extrai as propriedades estatísticas do conjunto de treinamento e agrupa os vetores similares em classes.

3.5. O Algoritmo *Backpropagation*

O algoritmo de retropropagação do erro (Backpropagation) consiste, basicamente, em determinar as variações nos pesos sinápticos da rede neural, tendo como objetivo minimizar o erro obtido na saída através do aprendizado do vetor de treinamento (entrada-saída). A característica inovadora desse algoritmo é dada pela sua capacidade de atribuir os erros obtidos na saída da rede às camadas ocultas.

O algoritmo trabalha em duas etapas: um passo de computação "para frente" (*forward-propagation*), onde é calculada a saída da rede, e um passo de computação "para trás" (*backpropagation*), onde os erros são calculados e os pesos são recalculados, de maneira a minimizar o erro.

O passo para a frente funciona da seguinte maneira: dado um exemplo de treinamento representado por (*x*, *d*), como o vetor de entrada *x* aplicado à camada de entrada de nós da rede, e o vetor saída desejada apresentado à camada de nós de saída da rede. Primeiramente devem ser calculados os campos locais induzidos e os sinais funcionais, prosseguindo através da rede, camada por camada. O campo local induzido $v_j^{(l)}$ para o neurônio *j* na camada *l* é:

$$v_j^{(l)} = \sum_{i=0}^{m_0} w_{ji}^{(l)} \cdot y_i^{(l-1)} \tag{3.13}$$

onde $y_i^{(l-1)}$ é o sinal (função) de saída do neurônio *i* na camada anterior *l-1*, e $w_{ji}^{(l)}$ é o peso sináptico do neurônio *j* na camada *l*, que é alimentado pelo neurônio *i* da camada *l-1*. Assumindo-se uma função de ativação sigmóide (vide item 3.3.1), o sinal de saída do neurônio *j* da camada *l* é dado por:

$$y_j^{(l)} = \varphi_j(v_j^{(l)}) \tag{3.14}$$

Se o neurônio *j* está na primeira camada da rede (ou seja, *l*=1), tem-se:

$$y_j^{(0)} = x_j \tag{3.15}$$

onde *xj* é o j-ésimo elemento do vetor de entrada *x*. Se o neurônio *j* está na camada de saída da rede (ou seja, *l=L*, onde *L* é chamado de *profundidade* da rede), tem-se:

$$y_j^{(L)} = o_j \tag{3.16}$$

O passo para a frente termina com o cálculo do erro na camada de saída:

$$e_j = d_j - o_j \tag{3.17}$$

O passo para trás se refere à atribuição de "responsabilidades" pelo erro na saída. Ele se inicia com o cálculo dos gradientes locais em cada camada da rede, definidos por:

$$\delta_j^{(l)} = e_j^{(L)} \cdot \varphi'_j(v_j^{(L)}) \text{ para o neurônio } j \text{ na camada de saída } L \tag{3.18}$$

$$\delta_j^{(l)} = \varphi'_j(v_j^{(l)}) \cdot \sum_k \delta_k^{(l+1)} \cdot w_{kj}^{(l+1)} \text{ para o neurônio } j \text{ da camada oculta } l \tag{3.19}$$

onde o apóstrofe em $\varphi'(\cdot)$ representa a diferenciação em relação ao argumento. Os pesos sinápticos da rede são ajustados então com base na seguinte regra:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta \cdot \delta_j^{(l)}(n) \cdot y_i^{(l-1)}(n) \tag{3.20}$$

onde η é a taxa de aprendizagem.

Os passos de computação para frente e para trás são então iterados sucessivamente, até que algum critério de parada seja alcançado (normalmente número de iterações ou erro pré-estabelecido).

Embora o algoritmo pareça razoavelmente simples (talvez aí resida sua força), ele está por trás da maior parte das aplicações contemporâneas de redes neurais. Como é possível depreender do histórico (seção 3.2), seu desenvolvimento tornou o campo das redes neurais artificiais um dos mais estudados em inteligência artificial.

3.5.1. Taxas de Aprendizagem

O algoritmo de retropropagação fornece uma "aproximação" para a trajetória no espaço de pesos calculada pelo método da descida pelo gradiente. Quanto menor for o parâmetro da taxa de aprendizagem η, menor serão as variações dos pesos sinápticos da rede, de uma iteração para a outra, e mais suave será a trajetória no espaço de pesos. Esta melhoria, entretanto, é obtida à custa de uma taxa de aprendizagem lenta. Por outro lado, se o parâmetro da taxa de aprendizagem for colocado muito grande, para acelerar o processo de treinamento, as grandes modificações nos pesos sinápticos resultantes podem tornar a rede instável (Haykin

(2000) chama de “oscilatória”). Resolver este dilema não é tarefa simples, e requer uma de duas atitudes: (i) experimentação com o algoritmo, ou seja, alteração “manual” da taxa de aprendizagem de maneira a obter resultados melhores, ou (ii) o uso de técnicas mais avançadas, que contornam engenhosamente o problema, introduzindo alterações no algoritmo básico, de maneira a obter melhor performance. Tais técnicas serão discutidas na seção 3.6.2 adiante.

3.6. Desempenho de Redes Neurais

Apesar de serem uma ferramenta poderosa, as redes neurais artificiais são extremamente sensíveis. Assim, sua aplicação em problemas práticas requer toda uma carga de preparação e cuidados *a priori*, e a análise *a posteriori* também requer uma boa dose de cuidado, de modo a não tomar por certo aquilo que não necessariamente é. Neste item, vai-se discutir alguns fatores fundamentais para a boa performance de redes neurais na resolução de problemas práticos, e quais passos devem ser tomados para preparar os dados, treinar a rede, validar os resultados obtidos e analisá-los.

3.6.1. Generalização

Na aprendizagem por retropropagação, começa-se tipicamente com uma amostra de treinamento e se usa o algoritmo de retropropagação para calcular os pesos sinápticos de um perceptron de múltiplas camadas carregando (codificando) tantos exemplos de treinamento quanto possível para dentro da rede. Espera-se que a rede neural assim projetada seja capaz de generalizar. Diz-se que uma rede generaliza bem quando o mapeamento de entrada-saída da rede for correto (ou aproximadamente correto) para dados de teste não-utilizados para a criação ou treinamento da rede; o termo “generalização” é tomado emprestado da psicologia. Aqui se assume que os dados de teste são tirados da mesma população usada para gerar os dados de treinamento.

O processo de aprendizagem (i.e., treinamento de uma rede neural) pode ser visto como um problema de “ajuste de curva”. A própria rede pode ser considerada

simplesmente como um mapeamento não-linear de entrada-saída. Este ponto de vista nos permite considerar a generalização não como uma propriedade mística das redes neurais, mas simplesmente como o efeito de uma boa interpolação não-linear sobre os dados de entrada (Haykin, 2001). A rede realiza boa interpolação fundamentalmente porque perceptrons de múltiplas camadas com funções de ativação contínuas produzem funções de saída que também são contínuas.

Uma rede neural, que é projetada para generalizar bem, produzirá um mapeamento de entrada-saída coreto, mesmo quando a entrada for um pouco diferente dos exemplos usados para treinar a rede. Entretanto, quando uma rede neural aprende um número excessivo de exemplos de entrada-saída, a rede pode acabar memorizando os dados de treinamento. Ela pode fazer isso encontrando uma característica (devido ao ruído, por exemplo) que está presente nos dados de treinamento, mas não na função subjacente que deve ser modelada. Este fenômeno é conhecido como *excesso de ajuste* ou *excesso de treinamento*[15]. Quando a rede é treinada em excesso, ela perde a habilidade de generalizar entre padrões de entrada-saída similares.

Normalmente, carregar dados desta forma em um perceptron de múltiplas camadas requer o uso de mais neurônios ocultos do que é realmente necessário, resultando que contribuições indesejáveis no espaço de entrada devido a ruído sejam armazenadas nos pesos sinápticos da rede. A "memorização" é essencialmente uma "tabela de consulta", o que implica que o mapeamento de entrada-saída computado pela rede neural não é suave. A suavidade do mapeamento de entrada-saída está intimamente relacionada com critérios de seleção de modelos do tipo *Navalha de Occam*, cuja essência é selecionar a função "mais simples" na ausência de qualquer conhecimento prévio contrário. No contexto da generalização, a função mais simples significa a função mais suave que aproxima o mapeamento para um dado critério de erro, porque esta escolha geralmente demanda os menores recursos computacionais. É, portanto, importante procurar um mapeamento não-linear suave para relações de entrada-saída mal-formuladas, de modo que a rede seja capaz de classificar corretamente novos padrões em relação aos padrões de treinamento.

[15] Tradução do inglês *overfitting* ou *overtraining*.

Além da discussão da generalização, é interessante mencionar um outro ponto: normalmente espera-se que uma rede neural se torne bem-treinada de modo que aprenda o suficiente do passado para generalizar no futuro. Desta perspectiva, o processo de aprendizagem se transforma em uma escolha de parametrização da rede para este conjunto de dados. Mais especificamente, é possível ver o problema de seleção da rede como a escolha, dentre um conjunto de estruturas de modelo candidatas (parametrizações), a "melhor" de acordo com um certo critério. Esta é uma maneira apenas semanticamente diferente de colocar a questão, em relação a abordagem de suavidade usada anteriormente.

Nesse sentido, uma ferramenta padrão da estatística conhecida como *validação cruzada* fornece um princípio orientador atraente (Haykin, 2001). Primeiramente, o conjunto de dados disponível é dividido aleatoriamente em um conjunto de treinamento e um conjunto de teste. O conjunto de treinamento é dividido adicionalmente em dois subconjuntos distintos: (i) *Subconjunto de estimação*, usado para selecionar o modelo; (ii) *Subconjunto de validação*, usado para testar ou validar o modelo.

A motivação é validar o modelo com um conjunto de dados diferente daquele usado para estimar os parâmetros. Desta forma, é possível usar o conjunto de treinamento para avaliar o desempenho de vários modelos candidatos e assim, escolher o "melhor". Há, entretanto, uma possibilidade considerável de que o modelo assim selecionado, com os valores de parâmetros com melhor desempenho, possa acabar ajustando excessivamente o subconjunto de validação. Para se resguardar dessa possibilidade, o desempenho de generalização do modelo selecionado é medido sobre o conjunto de teste, que é diferente do subconjunto de validação.

O uso de validação cruzada é atrativo particularmente quanto é necessário projetar uma rede grande cujo objetivo seja uma boa generalização[16]. É possível, por exemplo, utilizar a validação cruzada para determinar o perceptron de múltiplas camadas com o melhor número de neurônios ocultos e quando é melhor parar o treinamento. Portanto, a validação cruzada é uma ferramenta útil no projeto de boas redes neurais e na busca por modelos com alta capacidade de generalização.

[16] No capítulo 6, vai-se tentar alcançar este objetivo.

3.6.2. Algoritmos de Treinamento

O algoritmo de treinamento baseado em retropropagação trabalha normalmente com o método de descida por gradiente. Para a maioria dos problemas práticos, esse método é muito lento. Assim, ao longo do tempo foram sendo criados diversos algoritmos de alta performance para o treinamento de redes neurais com retropropagação, que convergem entre dez e cem vezes mais rapidamente que o algoritmo de descida por gradiente tradicional.

Estes algoritmos de alta performance podem ser divididos basicamente em duas categorias (Demuth e Beale, 2001): aqueles que usam técnicas heurísticas, desenvolvidas a partir de uma análise cuidadosa do algoritmo de descida por gradiente básico; e aqueles que utilizam métodos de otimização numérica, para acelerar a convergência dos pesos da rede.

Dentro do primeiro grupo, vale citar três diferentes métodos: (i) o uso de *momentum*, ou seja, a inclusão de um parâmetro adicional na equação de mudança de pesos da rede, de modo a acelerar a convergência em regiões onde o gradiente é pronunciado, e vice-versa em regiões onde a rede aprende pouco; (ii) o algoritmo com Taxa de Aprendizagem Variável, onde o parâmetro η da rede pode variar ao longo do processo de treinamento, evitando os problemas de instabilidade advindos de uma escolha *a priori* errônea deste parâmetro; e (iii) o algoritmo de Retropropagação Resiliente[17], que tenta evitar o problema de magnitude das derivadas parciais usadas para realizar a atualização dos pesos da rede (note-se nas equações do "passo para trás" do algoritmo *backpropagation*, que se usa constantemente as derivadas da função de ativação), e faz uso apenas do sinal destas derivadas, que indicam apenas a direção que se deve tomar na superfície de erro, deixando o problema da magnitude de variação dos pesos da rede para um outro parâmetro. De um modo geral, estas três modificações no algoritmo básico tendem a melhorar consideravelmente a performance do processo de treinamento, sem incorrer em substancial aumento dos recursos computacionais utilizados.

Já no segundo grupo, há também três técnicas diferentes que permitem obter ganhos no processo de aprendizado: Gradiente Conjugado, quasi-Newton e

[17] *Resilient Backpropagation*.

Levenberg-Marquardt. O algoritmo do Gradiente Conjugado parte da premissa de que, embora uma função decresça mais rapidamente na direção do negativo de seu gradiente (esta é a premissa do algoritmo básico de descida pelo gradiente), não necessariamente esta direção é que produz a melhor convergência global. Assim, o algoritmo faz uma busca numérica nas direções conjugadas à negativa do gradiente, tentando otimizar o resultado (numa analogia simples, é como o jogador de xadrez que pensa duas, três jogadas adiante). Há diversas maneiras de realizar tal busca, e daí surgem diversas variantes do algoritmo: Fletcher-Reeves, Polak-Ribiére, Powell-Beale, Golden Search, entre outros[18]. Os métodos quasi-Newton partem da mesma premissa básica das técnicas de gradiente conjugado, apenas utilizam métodos de otimização numérica diferentes, baseados em variantes do algoritmo de Newton. Da mesma maneira, o algoritmo de Levenberg-Marquardt busca otimização numérica com uma aproximação do método de Newton (em linhas gerais, o método de Newton envolve o cálculo do Hessiano de uma função – os métodos quasi-Newton e Levenberg-Marquardt buscam aproximar o Hessiano). Segundo Demuth e Beale (2001), este último tem se mostrado ser o algoritmo mais rápido para o treinamento de redes alimentadas adiante (*feedforward*) de tamanho moderado (algumas centenas de pesos).

3.6.3. Treinamento Seqüencial e por Lote

Em uma aplicação prática do algoritmo de retropropagação, o aprendizado resulta das muitas apresentações de um determinado conjunto de exemplos de treinamento para o perceptron de múltiplas camadas. Como mencionado anteriormente, uma apresentação completa do conjunto de treinamento inteiro é denominada uma *época*. O processo de aprendizagem é mantido em uma base de época em época até os pesos sinápticos e os níveis de viés se estabilizarem e o erro médio quadrado sobre todo o conjunto de treinamento convergir para um valor mínimo. É uma boa prática tornar aleatória a ordem de apresentação dos exemplos de treinamento (Haykin, 2001), de uma época para a seguinte. Esta aleatoriedade tende a tornar a busca no espaço de

[18] Para detalhes de cada um destes algoritmos, ver Demuth e Beale (2001), ou Haykin (2001), pp. 262-271.

pesos estocástica sobre os ciclos de aprendizagem, evitando assim a possibilidade de ciclos limitados, na evolução dos vetores de pesos sinápticos.

Para um dado conjunto de treinamento, a aprendizagem supervisionada pode então ocorrer de uma dentre duas formas básicas:

- *Modo Seqüencial*: o modo seqüencial da aprendizagem é também chamado de *modo online*, *modo padrão* ou *modo estocástico* (Haykin, 2001). Neste modo de operação, a atualização dos pesos é realizada após a apresentação de cada exemplo de treinamento.
- *Modo por Lote*: no modo por lote da aprendizagem supervisionada, o ajuste dos pesos é realizado após a apresentação de todos os exemplos de treinamento que constituem uma época. Para uma época particular, define-se a função de custo como:

$$\varepsilon_{médio} = \frac{1}{2N}\sum_{n=1}^{N}\sum_{j\in C} e_j^2(n) \tag{3.21}$$

onde o sinal de erro $e_j(n)$ é relativo ao neurônio de saída *j* do exemplo de treinamento *n*. O erro é igual à diferença entre a saída computada pela rede e a saída original do exemplo, ou seja, o *j-ésimo* elemento do vetor resposta desejada e o valor correspondente da saída da rede. Na equação anterior, o somatório interno em relação a *j* é realizado sobre todos os neurônios da camada de saída da rede, enquanto que o somatório externo em relação a *n* é realizado sobre todo o conjunto de treinamento da época considerada. Para um parâmetro de taxa de aprendizagem η, o ajuste aplicado ao peso sináptico w_{ji}, conectando o neurônio *i* ao neurônio *j*, é definido pela regra delta:

$$\Delta w_{ji} = -\eta\frac{\partial \varepsilon_{med}}{\partial w_{ji}} = -\frac{\eta}{N}\sum_{n=1}^{N} e_j(n)\frac{\partial e_j(n)}{\partial w_{ji}} \tag{3.22}$$

De acordo com essa equação, no modo por lote, o ajuste do peso w_{ji} é feito somente após o conjunto de treinamento inteiro ter sido apresentado à rede.

Do ponto de vista operacional *online*, o modo seqüencial de treinamento é preferível em relação ao modo por lote, porque requer menos armazenamento local para cada conexão sináptica (Haykin, 2001). Além disso, dado que os parâmetros são apresentados à rede de uma forma aleatória, o uso de ajuste de pesos de padrão torna a busca no espaço de pesos estocástica, o que torna menos provável que o algoritmo de treinamento fique preso em um mínimo local. Da mesma forma, a natureza

estocástica do modo seqüencial torna mais difícil de estabelecer as condições teóricas para a convergência do algoritmo (Haykin, 2001).

Em suma, apesar de o modo seqüencial apresentar várias desvantagens, ele é bastante usado por duas razões práticas importantes: (i) é simples de implementar e (ii) fornece soluções efetivas a problemas grandes e difíceis.

3.6.4. Técnicas de Poda de Rede

Para resolver problemas do mundo real com redes neurais, normalmente é necessário o uso de redes de tamanho bastante grande, altamente estruturadas. Uma questão prática que surge neste contexto é a da minimização do tamanho da rede, mantendo bom desempenho. É menos provável que uma rede neural com tamanho mínimo aprenda as idiossincrasias ou ruído dos dados de treinamento e, pode assim generalizar melhor sobre novos dados. É possível alcançar este objetivo de projeto de duas formas:

- Pelo *crescimento da rede*[19], começando com um perceptron de múltiplas camadas pequeno (para realizar a tarefa em questão), e então vai-se adicionando um novo neurônio ou uma nova camada de neurônios ocultos somente quando não se satisfizerem as especificações de projeto.
- Pela *poda da rede*[20], começando com um perceptron de múltiplas camadas grande, com um desempenho adequado para o problema em questão, e então o podando pela redução ou eliminação de certos pesos sinápticos de uma forma seletiva e ordenada.

As principais abordagens para a realização da poda da rede são: (i) baseada em uma forma de *regularização* e (ii) baseada em *eliminação* de certas conexões sinápticas da rede.

[19] Normalmente chamado de *cascading* ou *cascade-correlation* na literatura (Zeki-Susac, 1999).

[20] Normalmente chamado de *pruning* na literatura (Haykin, 2001 e Zeki-Susac, 1999).

A regularização agrupa uma variedade de métodos de poda da rede[21], com um princípio motivador em comum. No projeto de um perceptron de múltiplas camadas por qualquer método que seja, está-se de fato construindo um modelo não-linear de um fenômeno responsável pela geração de exemplos de entrada-saída usados para treinar a rede. Na medida em que o projeto da rede é de natureza estatística, se deseja um compromisso adequado entre a confiabilidade dos dados de treinamento e a qualidade do modelo (ou seja, um método adequado para resolver o dilema bias-variância (Haykin, 2001)). No contexto de aprendizagem supervisionada (*backpropagation* ou qualquer outro algoritmo), é possível realizar esse compromisso minimizando o risco total, expresso como:

$$R(w) = \varepsilon_S(w) + \lambda \varepsilon_C(w) \tag{3.23}$$

onde o primeiro termo é a *medida de desempenho* da rede (ou também o erro da rede) e o segundo termo é a *punição por complexidade* da rede, multiplicada pelo *parâmetro de regularização*. A punição por complexidade define o quanto a rede pode aprender a partir dos exemplos de treinamento – em outras palavras, qual a confiabilidade dos dados disponíveis, no sentido de definir bem a rede. Os algoritmos de poda de rede baseados em regularização trabalham para diminuir este parâmetro, evitando redes (e pesos sinápticos) muito grandes e assim permitindo redes que aprendam melhor e portanto generalizem melhor.

A abordagem por eliminação parte da idéia básica de podar a rede a partir da informação sobre as derivadas de segunda ordem da superfície de erro, de forma a estabelecer um compromisso entre a complexidade da rede e o desempenho do erro de treinamento. Em particular, constrói-se um modelo local da superfície de erro para prever analiticamente o efeito de perturbações sobre os pesos sinápticos, e a partir deste modelo tenta-se obter melhor performance de generalização da rede do que a normalmente seria obtida com uma técnica do tipo descida pelo gradiente. O objetivo é identificar um conjunto de parâmetros cuja eliminação do perceptron de múltiplas camadas causa o menor aumento do erro médio da rede, e a partir daí eliminar estes parâmetros.

[21] “Decaimento de Pesos”, “Eliminação de Pesos” e “Suavizador Aproximativo” são exemplos (Haykin,2001).

3.6.5. Pré e Pós-Processamento

Um dos resultados mais importantes no trabalho com redes neurais é a prova de que redes neurais são aproximadores universais de funções[22]. Em outras palavras, dado um número suficientemente grande de parâmetros livres, garantidamente o processo de treinamento vai achar um mapeamento entre qualquer conjunto de variáveis independentes e dependentes. Este é um resultado poderoso que garante que redes neurais podem atacar uma enorme gama de problemas, mas há um porém: elas também vão achar relações onde elas não existem. Portanto, o processo de selecionar e tratar as variáveis de um problema tem de ser parte integrante do processo de projeto de uma rede neural.

Independentemente da eficiência do algoritmo de aprendizagem em termos de convergência, generalização e estabilidade, o indicador de performance último de um estimador neural vai depender na relevância das variáveis independentes escolhidas e na qualidade dos dados usados. Esta é uma maneira rebuscada de dizer "Entra lixo, sai lixo"[23]. Além disso, trabalhar com muito poucas variáveis independentes vai diminuir o espaço de busca excessivamente, e introduzir vieses no processo de modelagem que em geral levam a generalizações pobres. De outro lado, variáveis demais aumentam a dimensionalidade do espaço de busca e tornam qualquer algoritmo computacionalmente ineficiente e portanto inútil.

A seleção de variáveis é uma etapa importante do processo de modelagem antes de se iniciar a computação propriamente dita. Outra etapa crucial é o pré-processamento dos dados, de modo que a rede neural tenha mais facilidade para trabalhar. De um modo geral, esse pré-processamento envolve basicamente três etapas (Haykin, 2001 e Refenes, 1995):

- *Remoção da média*: cada variável deve ser pré-processada de modo que seu valor médio, calculado sobre todo o conjunto de treinamento ou seja próximo de zero, ou seja pequeno comparado com o desvio-padrão. Para avaliar o significado prático

[22] Esse resultado é o famoso *Teorema da Aproximação Universal*. Para uma formulação do mesmo, vide Haykin, 2001, pp. 234-235. Quem provou o teorema, no contexto de perceptrons de múltiplas camadas, foi Cybenko, em 1989, no artigo "Approximation by Superposition of a sigmoidal funcion", *in Mathematics of Control, Signals and Systems,* vol. 2, pp. 303-314.

[23] Uma tradução livre do famoso ditado "*Garbage in, garbage out*".

desta regra, considere-se o caso extremo, onde as variáveis de entrada são positivas de modo consistente. Nesta situação, os pesos sinápticos de um neurônio na primeira camada oculta podem apenas crescer juntos ou decrescer juntos. Conseqüentemente, se o vetor peso daquele neurônio deve mudar de direção, ele só pode fazer isso ziguezagueando seu caminho através da superfície de erro, o que é tipicamente lento e deve ser evitado (Haykin, 2001).

- *Descorrelação*: as variáveis de entrada contidas no conjunto de entrada não devem ser correlacionadas, o que pode ser feito através de uma técnica chamada Análise de Componentes Principais[24].
- *Equalização da Covariância*: as variáveis de entrada descorrelacionadas devem ser escaladas para que suas covariâncias sejam aproximadamente iguais, assegurando-se com isso que os diferentes pesos sinápticos da rede aprendam aproximadamente com a mesma velocidade.

Além do pré-processamento, é obviamente fundamental realizar um pós-processamento dos resultados, de maneira a ter resultados interpretáveis dada uma saída qualquer da rede neural. O pós-processamento segue as mesmas etapas que o pré-processamento, mantendo as características do processo gerador intactas.

3.7. Conclusões

Nesta breve introdução ao vasto campo das redes neurais, buscaram-se fundamentalmente dois objetivos: (i) apresentar os principais conceitos envolvidos na concepção e projeto de uma rede neural artificial e (ii) dar uma medida dos fatores que podem fazer com que uma rede tenha um bom desempenho na resolução de problemas complexos. O uso desta ferramenta não é trivial, pelo contrário, logo a obtenção de bons resultados, como já mencionado, é profundamente dependente de um processo de projeto e modelagem cuidadoso. A teoria é um guia poderoso para o projetista, mas apenas a experimentação com os algoritmos e os diferentes parâmetros torna os resultados consistentes.

[24] Por restrições de espaço e escopo do trabalho, não se vai detalhar aqui esta ferramenta. Para detalhes sobre seu funcionamento, vide Haykin (2001) ou Demuth e Beale (2001).

4. ALGORITMOS GENÉTICOS[1]

"Você acha que um dia as máquinas pensarão? Pode apostar que sim. Sou máquina, você é máquina e ambos pensamos, não?"

Claude Shannon[2]

4.1. Introdução

Assim como as redes neurais, que se discutiu no capítulo anterior, o próprio nome dos algoritmos genéticos desperta mistério e curiosidade. Talvez pelo fato de a evolução ser um dos conceitos mais antiintuitivos do corpo de conhecimentos comum das pessoas – há uma tendência por demais natural ao lamarckismo (acreditar que o aparecimento de determinadas características é "forçado" pelo ambiente) e não ao darwinismo. De todo modo, o apelo intelectual da genética (vide a ampla divulgação do Projeto Genoma) é algo contagiante, o que acaba por resvalar para a área dos Algoritmos Genéticos.

Ao longo das próximas páginas, se buscará estabelecer as bases do funcionamento e do uso dos algoritmos genéticos, porque são eles uma ferramenta poderosa para o tratamento de complexos problemas de busca, e como eles se comparam com outras ferramentas mais tradicionais.

4.2. Perspectiva Histórica

Nos anos 50 e 60, diversos cientistas da computação independentemente estudaram sistemas evolutivos, baseados na idéia de que a evolução poderia ser usada como uma ferramenta de otimização em problemas de engenharia. A idéia em todos esses sistemas era evoluir uma população de solução possíveis, usando operadores inspirados pela genética e pela seleção natural.

Nos anos 60, Rechenberg (1965, 1973) introduziu as "estratégias de evolução"[3], um método usado para otimizar parâmetros reais de mecanismos como aerofólios,

[1] Este capítulo é baseado, em grande medida, em Mitchell, M. (1999) e Silva, A.P.A. (2002).

[2] Apud Bass, 2000.

por exemplo. Essa idéia foi levada adiante e desenvolvida por Schwefel (1975, 1977). O estudo de estratégias de evolução continua uma ativa área de pesquisa, se desenvolvendo quase que independentemente do campo de algoritmos genéticos. Fogel, Owens e Walsh (1966) desenvolveram a "programação evolutiva"[4], uma técnica na qual soluções candidatas para determinados problemas eram representadas como máquinas de estado-finito, que eram evoluídas através da mutação aleatória de seus diagramas de transição de estado, para ao fim o mais adaptado ser escolhido. Ao longo do tempo, a programação evolutiva recebeu formulação mais ampla, e também se converteu em ativa área de pesquisa. Juntas, as áreas de "estratégias de evolução", "programação evolutiva" e algoritmos genéticos formam a base do campo de pesquisas em computação evolutiva.

Diversas outros estudiosos, trabalhando ao longo dos anos 50 e 60, desenvolveram algoritmos inspirados no mecanismo da evolução, usados para otimização e aprendizado de máquinas. Box (1957), Friedman (1959), Bledsoe (1961), Bremermann (1962), e Reedd, Toombs e Baricelli (1967) trabalharam nessa área, embora suas pesquisas tenham ficado algo esquecidas. Além disso, vários biólogos usaram computadores para simular a evolução, na tentativa de reproduzir experimentos controlados (eg. Baricelli 1957 e Martin & Cockerman 1960). Naquela época de nascimento dos computadores, a evolução certamente era um tópico de interesse.

Especificamente, os Algoritmos Genéticos foram inventados por John Holland nos anos 60, e desenvolvidos por ele e seus estudantes e colegas na Universidade de Michigan ao longo dos anos 60 e 70. Em contraste com os campos de estratégias de evolução e programação evolutiva, o objetivo original de Holland não era projetar algoritmos para resolver problemas específicos, mas sim estudar formalmente o fenômeno da adaptação, a maneira como ela ocorre na natureza, e simultaneamente desenvolver maneiras de importar para os computadores esse mecanismo. Com a publicação do livro *Adaptation in Natural and Artificial Systems*[5] em 1975, Holland

---

[3] Tradução do inglês *evolution strategies*.

[4] Tradução do inglês *evolutionary programming.*

[5] Infelizmente, não se encontrou tradução para o português desse livro. A edição original em inglês se esgotou, e foi republicada, revisada e expandida, em 1992 pela *MIT Press*, com o título *Adaptation In Natural And Artificial Systems: An Introductory Analysis With Applications To Biology, Control, And Artificial Intelligence*.

apresentou o algoritmo genético como uma abstração da evolução biológica e colocou o quadro teórico para a adaptação dos algoritmos genéticos[6]. O AG de Holland é um método para ir de uma população de cromossomos (ou seqüências de uns e zeros, ou *bits*), para uma nova população através do uso de um tipo de seleção natural, juntamente com os operadores (inspirados na genética) de *crossover*, mutação e inversão. Cada cromossomo consiste de uma série de genes (ou *bits*), cada gene sendo uma instância de um particular alelo (0 ou 1). O operador de seleção escolhe os cromossomos da população que poderão se reproduzir, e na média, os cromossomos mais adaptados produzem mais descendentes do que os menos adaptados. O operador de *crossover* troca partes dos cromossomos, copiando a recombinação biológica entre dois organismos de filamento únicos (haplóides, no jargão biológico). O operador de mutação troca os valores de alguns alelos, de forma aleatória. O operador de inversão troca a ordem completa de uma seção contígua de um cromossomo, trocando a ordem em que os genes são arranjados. Note-se que esta é apenas uma introdução num contexto histórico do funcionamento desses mecanismos. Um desenvolvimento mais formal será realizado no item 4.5.

A introdução feita por Holland de um algoritmo baseado em população, com *crossover*, inversão e mutação foi uma inovação importante (por exemplo, as estratégias de evolução de Rechenberg começavam com uma população de dois indivíduos: um pai e um descendente, sendo este uma versão mutada daquele; populações com muitos indivíduos e *crossover* não faziam parte da teoria então. Do mesmo modo, a programação evolutiva de Fogel, Owens e Walsh usava apenas mutação para prover a variação). Além disso, Holland foi o primeiro a tentar colocar a computação evolutiva em bases teóricas sólidas. Até recentemente essas bases teóricas, baseada na noção de *schemas*, era a base de todo o desenvolvimento subseqüente no campo.

Nos anos mais recentes tem havido uma crescente interação entre os pesquisadores estudando os vários sub-campos de computação evolutiva, e as separações entre AGs, estratégias de evolução, programação evolutiva e outros ramos do campo tem se dissolvido. Hoje, pesquisadores usam o termo algoritmo genético para descrever coisas bastante distantes da concepção original de Holland.

[6] Daqui por diante, vai-se usar a expressão *Algoritmo Genético* e a sigla *AG* indistinta e intercambiavelmente.

4.3. Evolução como Ferramenta

Porque usar a evolução como inspiração para resolver problemas computacionais? Para pesquisadores de computação evolutiva, os mecanismos da evolução parecem apropriados para os problemas mais significativos de muitos campos. Muitos problemas requerem uma busca entre um enorme número de possibilidades de solução. Um exemplo é o problema de engenharia de proteínas, no qual necessita-se de um algoritmo que busque entre o vasto número de seqüências de aminoácidos, que gerem a proteína desejada. Tais problemas de busca normalmente podem se beneficiar do uso eficiente do paralelismo, no qual muitas possibilidades diferentes são examinadas simultaneamente de maneira eficiente. Por exemplo, ao procurar por proteínas com determinadas propriedades, ao invés de avaliar uma seqüência de aminoácidos por vez, seria muito mais rápido avaliar vários simultaneamente. O que é preciso para se fazer isso é tanto paralelismo computacional (ou seja, muitos processadores rodando ao mesmo tempo) e uma estratégia inteligente para se escolher o próximo grupo a ser examinado.

Muitos problemas computacionais requerem um programa que seja adaptativo – ou seja, que consiga continuar a funcionar eficientemente sob condições ambientais diferentes. Isso pode ser tipificado por problemas em controle robótico, no qual um robô tem de executar tarefas em ambientes variáveis. Outros problemas requerem programas que seja inovativos, que construam soluções totalmente novas e originais. Finalmente, vários problemas requerem soluções complexas que são extremamente difíceis de serem programadas “a mão”. Um exemplo notável é o problema de se criar inteligência artificial. Nos primórdios da pesquisa nesse campo, pensava-se que seria possível codificar as regras que conferem inteligência em um programa (o campo de sistemas especialistas é resultado desse otimismo inicial). Mais recentemente, muitos estudiosos do campo acreditam que as regras subjacentes à inteligência são complexas demais para serem codificadas diretamente, numa perspectiva *top-down*. Ao contrário, eles acreditam que a melhor maneira de se chegar à inteligência artificial seja através de uma perspectiva *bottom-up,* na qual se programam apenas algumas regras simples, e comportamentos complexos como a inteligência emergem através da aplicação e interação massivamente paralela dessas

regras. Tanto o desenvolvimento e estudo de redes neurais (que se examinou no capítulo 3 desse trabalho), quanto o estudo da computação evolutiva são reflexo desse paradigma *bottom-up* de busca de soluções adaptativas para problemas complexos.

A biologia e a evolução são fontes atraentes para a busca de inspiração para atacar essa classe de problemas. De certo modo, a evolução é um método de se buscar entre um número enorme de soluções possíveis. Na biologia, o grande conjunto de possibilidades é a variedade de diferentes seqüências genéticas, e a solução desejada são os indivíduos bem adaptados – capazes de sobreviver e se reproduzir no ambiente. Além disso, a evolução pode ser vista como uma maneira de se *projetar* soluções inovadores para problemas complexos. Por exemplo, o sistema imunológico dos mamíferos é uma solução engenhosa para o problema de germes invadindo o corpo. Olhando por esse prisma é que o mecanismo da evolução inspira métodos computacionais de busca. A *utilidade*[7] de um indivíduo biológico depende de muitos fatores diferentes, por exemplo, quão bem se adapta ao clima do meio ambiente, ou quão bem consegue competir e/ou cooperar com os outros organismos a sua volta. O critério de utilidade muda constantemente à medida que os organismos evoluem, e portanto a evolução é um mecanismo de busca dinâmico num conjunto sempre mutável de possibilidades. Assim, a busca por boas soluções[8] em face a condições mutáveis é precisamente o que se requer de algoritmos computacionais adaptativos. Indo além, é possível dizer que a evolução é um mecanismo massivamente paralelo de busca: ao invés de testar uma espécie por vez, ela trabalha sobre milhares, quiçá milhões de espécies diferentes ao mesmo tempo. E também, em termos das regras que a governam, a evolução é impressionantemente simples: as espécies evoluem através de mutação randômica (mutação, recombinação e operadores semelhantes), seguido pela seleção dos mais adaptados, que tendem a sobreviver e se reproduzir, propagando então sua herança genética para gerações futuras. Tais regras são simples, e no entanto são capazes de explicar a enorme diversidade que vemos na biosfera.

[7] Aqui vai-se utilizar o termo *utilidade* como uma *proxy* do termo *fitness*, jargão padrão da área. Há, sem dúvida, uma inspiração do jargão comum em economia por trás dessa escolha.

[8] Note-se o uso da expressão “boas soluções” e não da expressão “soluções ótimas”, para reforçar o caráter heurístico dos AGs.

4.4. Terminologia Biológica

Nesse momento é relevante introduzir algumas das terminologias biológicas que serão usadas daqui por diante neste trabalho. No contexto dos algoritmos genéticos, tais termos são usados de acordo com a analogia com a biologia real, embora as entidades a que se refiram sejam um tanto mais simples que as correspondentes biológicas reais.

Todos os organismos vivos consistem de células, e cada célula contém o mesmo conjunto de um ou mais *cromossomos* – filamentos de DNA – que servem como a impressão digital, por assim dizer, de um organismo. Um cromossomo pode ser conceitualmente dividido em *genes* – blocos funcionais de DNA, cada um dos quais codifica uma particular proteína. Grosseiramente, pode-se imaginar um gene como o código de uma particular característica, digamos cor dos olhos. As diferentes possibilidades para uma característica (olhos verdes, azuis, castanhos) são chamadas de *alelos*. Cada gene está localizado em um particular *locus* (posição) no cromossomo.

Muitos organismos têm múltiplos cromossomos em cada célula. O conjunto completo do material genético (todos os cromossomos juntos) é chamado de o *genoma* de um organismo. O termo *genótipo* se refere ao grupo particular de genes contido em um genoma. Por exemplo, João e José têm genótipos diferentes (excluída a possibilidade de serem gêmeos univitelinos, mas aqui se busca apenas uma introdução, não um curso completo de biologia), mas seu material vem do mesmo genoma. O genótipo dá origem, através do desenvolvimento fetal e fases posteriores, ao *fenótipo* – o conjunto de características de um particular indivíduo.

Organismos cujos cromossomos estão dispostos em pares são chamados *diplóides*, enquanto organismos cujos cromossomos não são pareados são chamados *haplóides*. Na natureza, a maioria dos organismos que se reproduzem sexualmente são diplóides, incluído aí o ser humano, que tem 23 pares de cromossomos em cada célula do corpo. Durante a reprodução sexual, a *recombinação* (ou *crossover*) ocorre: em cada pai, os pares de cromossomos trocam genes e formam um *gameta* (cromossomo único), e gametas dos dois pais são pareados para formar um novo organismo diplóide. Na reprodução haplóide, a recombinação acontece diretamente

entre os filamentos cromossômicos dos pais. Além disso, o descendente está sujeito a *mutação*, na qual *nucleotídeos* (blocos elementares de DNA) são trocados do pai para o descendente, sendo que tais mudanças normalmente resultam de erros de cópia no processo reprodutivo. Ao fim, a utilidade do organismo é tipicamente definida como a probabilidade de que ele vai viver para se reproduzir (ou *viabilidade*), ou ainda como uma função do número de descendentes que ele vai deixar (ou *fertilidade*).

Nos AGs, o termo cromossomo se refere tipicamente a uma solução candidata para um problema, normalmente codificada como uma seqüência de *bits*. Os genes são ou *bits* únicos ou pequenos blocos de *bits* adjacentes, que codificam um particular elemento da solução candidata (por exemplo, no contexto de otimização de funções multivariadas, cada *bit* pode ser um determinado parâmetro da função). Um alelo numa seqüência de *bits* é ou 0 ou 1. *Crossover* tipicamente consiste na troca de material genético entre dois pais haplóides. A mutação consiste na troca do *bit* de um locus selecionado aleatoriamente.

A maioria das aplicações de AGs faz uso de indivíduos haplóides, particularmente indivíduos com apenas um cromossomo[9]. O genótipo de um indivíduo é então simplesmente a configuração de *bits* no cromossomo dele. Normalmente não há noção de *fenótipo* no contexto dos AGs, embora mais recentemente haja referências nesse sentido, onde se usa AGs em que há tanto um nível genotípico e um nível fenotípico (por exemplo no contexto do uso de algoritmos genéticos para otimizar os parâmetros de uma rede neural, em que o genótipo é o AG e o fenótipo é o comportamento emergente resultante da rede neural).

4.5. Elementos Básicos de um Algoritmo Genético

Se há uma ironia envolvida no estudo de algoritmos genéticos, é talvez o fato de não haver exatamente uma definição precisa do termo “algoritmo genético”. No entanto, a maioria daquilo que se pode chamar de AGs tem ao menos as seguintes características em comum: populações de cromossomos, seleção de acordo com a utilidade, *crossover* para produzir novos descendentes, e mutação aleatória. A

[9] Uma exceção interessante é a codificação que W. Daniel Hillis fez do problema de classificação de árvores de busca de 16 elementos, onde ele codificou os indivíduos de forma diplóide (W. Daniel Hillis 1990).

inversão – um dos elementos presentes no algoritmo original de Holland – é raramente usada atualmente, e não há conclusões acerca de quaisquer vantagens que tal procedimento traga para o resultado final do algoritmo.

Os cromossomos de uma população em um AG normalmente tomam a forma de seqüências de *bits*. Cada *locus* no cromossomo têm duas possíveis formas, ou alelos: 0 ou 1. Cada cromossomo pode ser visto como um ponto no espaço de busca de soluções candidatas. O algoritmo genético processa seqüencialmente as populações de cromossomos, substituindo progressivamente uma população pela outra. Normalmente se faz uso de uma função de utilidade[10] que dá a cada cromossomo um valor (a utilidade dele). A utilidade de um cromossomo depende basicamente de quão bem ele consegue resolver o problema em questão. Idealmente, a solução ótima dentro de um determinado espaço de busca corresponde àquela com a maior utilidade.

4.5.1. Codificação

De maneira a aplicar um AG para um determinado problema, a primeira decisão que deve ser tomada trata do tipo de genótipo que o problema requer. Isso quer dizer que se deve decidir acerca de uma maneira de mapear os parâmetros do problema em um conjunto finito de símbolos, ou genes (que podem ter comprimento constante ou dinâmico), codificando uma solução candidata no espaço de soluções para o problema. O conjunto de símbolos usado normalmente é binário (formando uma sequência de *bits*), mas outras representações também podem ser usadas, incluindo algumas com caracteres (letras) ou outros valores reais.

Na maioria das aplicações de AGs, as sequências são binárias e seu comprimento é mantido constante ao longo de todo o processo evolutivo. Além disso, todos os parâmetros são codificados no mesmo espaço de valores e alocados a genes do mesmo tamanho dentro da sequência de *bits*. Há um problema quando um gene pode ter apenas um número finito de valores válidos, se uma representação binária é utilizada. Se o número de valores válidos não é uma potência de dois, então alguns dos códigos binários serão redundantes, ou seja, eles não corresponderão a nenhum

[10] Como comentado na nota 5, utilidade se refere a *fitness*, e portanto função de utilidade é o termo usado para *fitness function*.

gene válido. A maneira mais popular de se contornar esse problema é através do mapeamento de um código inválido para um válido (ou seja, dois genes diferentes correspondem ao mesmo valor de um parâmetro).

4.5.2. Funções de Utilidade

Uma aplicação comum de AGs é em otimização de funções, onde o objetivo é encontrar um conjunto de parâmetros que maximize, por exemplo, uma determinada função multivariada complexa. Como um exemplo simples, tome-se a seguinte função, que se deseja maximizar:

$$f(y) = y + |sen(32y)| \qquad (4.1)^{11}$$

aqui as soluções candidatas são valores de *y*, que podem ser codificados como seqüências de *bits* representando números reais. O cálculo da função de utilidade transforma uma dada seqüência binária *x* em um número real *y* e então avalia a função naquele ponto. Nesse caso simples, a utilidade é justamente o valor da função naquele ponto.

Para um caso não-numérico, considere-se o problema de encontrar uma seqüência de 50 aminoácidos que se dobrem para formar uma estrutura protéica tridimensional desejada. Um AG poderia ser aplicado a este problema para buscar dentre uma população de soluções candidatas, codificando cada solução como uma seqüência de 50 letras, por exemplo

IHCCVASASDMIKPVFTVASYLKNWTKAKGPNFEICISGRTPYWDNFPGI[12]

onde cada letra representa um de 20 aminoácidos possíveis. Uma maneira de definir a utilidade de uma seqüência candidata é com o valor negativo da energia potencial da estrutura candidata, com respeito à estrutura desejada. A energia potencial é uma medida de quanta resistência física a seqüência apresentaria ao processo de dobramento para chegar à estrutura desejada – assim, quanto menor a energia potencial, maior a utilidade. Obviamente, não se deseja dobrar fisicamente todas as seqüências da população e medir sua resistência – seria extremamente difícil, para não dizer impossível. A solução nesse caso é usar algum conhecimento de biofísica e

[11] Riolo (1992) apud Mitchell (1996), adaptado pelo autor.

[12] Mitchell (1996).

estimar a energia potencial da estrutura através dos cálculos das forças relevantes agindo sobre cada aminoácidos, de modo que o cálculo da função de utilidade seja funcionalmente factível.

Esses dois exemplos mostram dois contextos diferentes na qual as soluções candidatas para um determinado problema são codificadas como cromossomos, ou seqüências de bits (ou símbolos), com funções de utilidade definidas no espaço de símbolos resultante. Em suma, um algoritmo genético é um método para se buscar, nesses espaços de utilidade, seqüências altamente adaptadas.

Um problema comum em AGs, chamado "enganação"[13], ocorre quando genes de alguns indíviduos bem-adaptados (mas não ótimos) passam a dominar a população após apenas algumas iterações (ou gerações), causando um máximo local. Quando a população alcança esse ponto, a capacidade do AG de continuar buscando soluções melhores é praticamente eliminada. Apenas maneira de evitar isso é controlando o número de oportunidades que cada indíviduo tem de se reproduzir (Silva, 2002).

4.5.3. Operadores

Todo algoritmo genético envolve basicamente três tipos de operadores: seleção, *crossover* e mutação. A seguir, breves descrições de seu funcionamento:

O operador de *seleção* escolhe cromossomos na população para se reproduzirem. Quanto maior a utilidade de um cromossomo (ou seja, quão mais bem adaptado no espaço de soluções ele é), maior a probabilidade de ele ser selecionado para se reproduzir.

O operador de *crossover* seleciona, aleatoriamente, um *locus* e troca as subseqüências antes e depois desse *locus* entre dois cromossomos, de maneira a criar dois descendentes que são a mistura dos códigos genéticos dos dois pais. Assim, por exemplo, as seqüências 10000100 e 11111111 podem sofrer *crossover* após o terceiro *locus* de cada uma, gerando os descendentes 10011111 e 11100100. O operador de *crossover* imita (grosseiramente) a recombinação biológica do material genético de dois organismos haplóides.

[13] Tradução do inglês *deception*.

O operador de *mutação* troca, aleatoriamente, alguns dos *bits* de um cromossomo. Por exemplo, a seqüência 00000100 pode ser mudada na sua segunda posição, passando a ser 01000100. A mutação pode ocorrer em cada *bit* de uma seqüência de acordo com alguma probabilidade, normalmente bastante pequena (por exemplo, 0,01%).

4.6. Um Algoritmo Genético Básico

Dado um problema bem-definido, e uma seqüência de *bits* representativa das soluções candidatas, um algoritmo genético básico funcionaria da seguinte maneira:

- Iniciar com uma população gerada aleatoriamente, com *n* cromossomos de tamanho *l* (soluções candidatas para o problema dado).
- Calcular a função de utilidade *f(x)* para cada cromossomo *x* na população.
- Repetir os seguintes passos até que *n* descendentes tenham sido criados:
  - Selecionar um par de cromossomos pais dentro da população atual, sendo a probabilidade de seleção uma função crescente do valor da utilidade. A seleção é feita com substituição, ou seja, o mesmo cromossomo pode ser escolhido mais de uma vez para ser pai de um descendente.
  - Dada a probabilidade p_c (a probabilidade de *crossover* ou taxa de *crossover*), realizar o cruzamento da seqüência de *bits* dos dois pais em um ponto escolhido aleatoriamente[14], de maneira a formar dois descendentes. Se não acontece o *crossover* (probabilidade 1- p_c), os dois descendentes serão cópias idênticas dos pais. (Note-se que a taxa de *crossover* é definida como a probabilidade de que os dois pais vão ter cruzamento em um ponto único. Determinadas implementações de AGs usam "*crossover* múltiplo", no qual a taxa é o número de pontos no qual acontece cruzamento).
  - Realizar mutação dos dois descendentes gerados, com probabilidade p_m (probabilidade de mutação ou taxa de mutação) para cada *locus* das seqüências geradas.
  - Incluir os descendentes gerados na população.
- Substituir a população atual pela nova população.

[14] Da maneira delineada no item 4.5.2 anterior.

- Retornar ao passo 2.

Cada iteração desse processo é chamada de geração. Normalmente, um algoritmo genético é iterado entre 50 e 500 gerações. O conjunto completo de gerações é chamado de *rodada*[15]. Ao fim de uma rodada, espera-se que haja um ou mais indivíduos altamente adaptados na população (indivíduos com alta taxa de utilidade). Dado que há uma forte dose de aleatoriedade em cada geração, e portanto em cada rodada, duas rodadas com geradores de números aleatórios (ou sementes de geradores) diferentes vão resultar, geralmente, em comportamentos diferentes da população. Por isso é comum se trabalhar com resultados estatísticos de muitas rodadas diferentes de um algoritmo para o mesmo problema.

O procedimento anterior é a base para a maioria das aplicações de AGs. Há uma variedade de detalhes não-desprezíveis que precisam ser levados em conta, como o tamanho da população e a escolha das taxas de *crossover* e mutação, e muitas vezes o sucesso do algoritmo na resolução do problema depende fortemente da escolha acertada desses parâmetros. Além disso, é possível trabalhar com versões mais complexas de AGs, como por exemplo fazer uso de cromossomos codificados de maneira que não a seqüência simples de *bits*, ou usar operadores de *crossover* e mutação com características diferentes (Mitchell, 1998).

4.7. Algoritmos Genéticos *vs.* Métodos de Busca Tradicionais

Até aqui, o termo busca foi usado diversas vezes para descrever aquilo que os AGs fazem. É importante, portanto, distinguir esse significado de busca dos outros significados que o termo adquire no contexto da ciência da computação. É possível mencionar pelo menos três sentidos que a palavra adquire, de certa maneira sobreposta:

- **Busca por dados guardados**, onde o problema é se achar eficientemente informação guardada em memória. Por exemplo, tome-se uma base de dados bastante grande de nomes e endereços, ordenados de alguma maneira pré-determinada. Qual a melhor maneira de buscar o registro correspondente a um

[15] Tradução do inglês *run*.

dado sobrenome? “Busca Binária” é um método para se achar eficientemente o registro desejado.

- **Busca de caminhos**, onde o problema é encontrar eficientemente um conjunto de ações que levam de um estado inicial dado para um objetivo desejado. Esse tipo de busca é crucial para vários ramos do estudo de inteligência artificial. Normalmente se trabalha com uma árvore de possibilidades, onde o nó “raiz” representa o estado inicial do problema e as ramificações representam todos os possíveis estados subseqüentes do sistema a partir daquele nó. Os algoritmos de busca usados nesse contexto são métodos para encontrar de maneira eficiente o melhor (no caso, o mais curto) caminho entre o nó “raiz” e o estado desejado, ou nó objetivo. Alguns algoritmos típicos para tanto são *branch and bound*, *A**, entre outros.
- **Busca de soluções**, que é uma classe mais geral do que a anterior. A idéia aqui é encontrar eficientemente uma solução para um problema dentro de um grande espaço de soluções candidatas. É para esse tipo de problemas que são usados algoritmos genéticos.

Há claramente uma grande diferença entre o primeiro tipo de busca e os dois seguintes. O primeiro trata basicamente de encontrar uma informação que já está claramente guardada em algum lugar. Nos outros dois, o problema é encontrar informação que não está explicitamente guardada. Ao contrário, soluções candidatas vão sendo criadas a medida que o processo de busca avança. Para a maioria dos problemas em que há uma árvore de busca, existem simplesmente ramos demais para que se guarde todos na memória. Assim, a árvore de busca é criada passo a passo de maneira que depende do algoritmo particular que está sendo aplicado, e o objetivo subjacente é encontrar uma solução ótima ou quase-ótima através do exame de apenas uma pequena porção da árvore total.. Do mesmo modo, quando se está examinando o espaço de soluções candidatas com um Algoritmo Genético, nem todas as possíveis soluções candidatas são criadas e depois avaliadas em sua utilidade. Na verdade, o AG é uma ferramenta para se encontrar uma solução ótima ou quasi-ótima através do exame de uma pequena fração das possíveis candidatas.

De uma certa maneira, a classificação “Busca por soluções” engloba a classificação “Busca de caminhos”, já que o caminho através de uma árvore de busca

normalmente pode ser codificado como uma solução candidata. No entanto, a separação é feita porque a maioria dos problemas do tipo "Busca por caminhos" é resolvida mais facilmente por técnicas do tipo busca em árvore (como o *branch and bound* por exemplo) do que por técnicas do tipo Algoritmo Genético ou assemelhadas, onde as soluções candidatas são explicitamente geradas e avaliadas.

No entanto, nem todos os problemas são tratáveis com algoritmos do tipo busca em árvore (ou mais genericamente, busca em grafos). Um exemplo é o problema de previsão da estrutura tridimensional de uma proteína a partir da sua sequência de aminoácidos. Para resolver esse problema não necessariamente se precisa saber a exata sequência de movimentos em que o dobramento da sequência ocorre – apenas se deseja o formato final da proteína.

O Algoritmo Genético é uma técnica geral para resolução de problemas do tipo "Busca de soluções". Outros exemplos de técnicas que se prestam ao mesmo propósito são: *simulated annealing*, *tabu search* e *hill climbing*[16]. No quadro geral das teorias de Inteligência Artificial, técnicas gerais (que se prestam a uma grande classe problemas) são chamadas "técnicas fracas", para diferenciá-las das "técnicas fortes", que são desenvolvidas especialmente para problemas particulares. De um modo geral, as técnicas de "Busca de soluções" funcionam do seguinte modo: (i) inicialmente se gera um conjunto de soluções candidatas; (ii) seguindo algum critério de utilidade, avaliam-se as soluções candidatas; (iii) baseado na avaliação, decide-se quais soluções são mantidas e quais são descartadas e finalmente; (iv) novas variantes são produzidas através do uso de algum operador sobre o conjunto restante.

O que torna os Algoritmos Genéticos uma técnica distinta das outras é uma particular combinação de elementos, quais sejam: busca maciçamente paralela dentro da população, seleção estocástica juntamente com mutação e *crossover* estocásticos. Vários outros métodos incluem um ou vários desses elementos, mas não essa combinação exata. E é justamente ela que faz dos AGs uma técnica útil para o problema que vai se resolver neste trabalho.

[16] Nos três casos se optou pelo termo original em inglês do que traduções como "recozimento simulado".

5. INTELIGÊNCIA ARTIFICIAL EM FINANÇAS: ANÁLISE DA PESQUISA PRÉVIA

"Os mercados acionários são um excelente laboratório para o teste de teorias"
George Soros[1]

5.1. Introdução

Após introduzir a teoria relevante, tanto na área de finanças quanto na área de inteligência artificial (que obviamente não se preocupa apenas com redes neurais e algoritmos genéticos, mas este é o foco do presente trabalho), se faz necessário agora analisar aquilo que já foi estudado sobre o assunto, ou seja, aplicações tanto de redes neurais quanto de algoritmos genéticos a problemas de finanças. Para tanto, será primeiro necessário classificar os problemas e modelos utilizados na pesquisa prévia, para então restringir o foco e examinar mais detidamente o que já se estudou em relação ao problema de previsão do mercado de ações e da seleção de portfólios.

É relevante mencionar que em alguns poucos trabalhos, pesquisadores adotaram abordagens semelhantes à que se propõe utilizar nesse trabalho[2], mas não necessariamente os detalhes destes trabalhos serão aplicados aqui. Justamente por isso, neste capítulo vamos dividir a análise: primeiramente abordar o aspecto das redes neurais, tentando tirar conclusões sobre como proceder adiante, e posteriormente examinar o aspecto dos algoritmos genéticos, e aí também tomar conclusões sobre benefícios e limitações de resultados passados.

5.2. Redes Neurais em Finanças

Uma das aplicações mais comuns de redes neurais é justamente na área de finanças e investimentos. Alguns dos problemas mais representativos sendo resolvidos com elas são: previsão de falências, avaliação de risco de empréstimos e hipotecas, previsão de mercado (tanto ações quanto títulos de renda fixa, preços de opções, retornos de capital, análise técnica entre outras variantes), prognóstico de

[1] Apud Bass, 2000.

[2] Por tal abordagem, entenda-se usar redes neurais para prever o mercado e algoritmos genéticos para realizar a alocação da carteira.

investimentos (retornos), entre outros menos significativos (Zeki-Susac, 1999). Empresas como o Chase Manhattan Bank (Marose, 1990 apud Zeki-Susac, 1999) e American Express (Zahedi, 1993) são algumas das empresas que aplicam eficientemente redes neurais no dia-a-dia. No Brasil, há reportagens mencionando a aplicação de redes neurais[3] em alguns bancos, além de alguns poucos artigos e teses[4] acadêmicos, mas não está claro nem pôde ser estabelecido confiavelmente o uso corrente de redes neurais em grandes empresas.

As várias pesquisas sobre aplicação de redes neurais em finanças provaram, ao longo do tempo, as vantagens do uso da inteligência artificial sobre o uso de métodos tradicionais. De acordo com Wong et al. (1997)[5], as áreas mais freqüentes de aplicação de redes neurais são: produção / operações (53,5%) e finanças (25,4%). Justamente por isso, a literatura é razoavelmente abundante, e , a seguir, tentar-se-á delinear as principais aplicações e os principais resultados levantados.

Um ponto de partida interessante para o estudo de redes neurais aplicadas à previsão do mercado é o trabalho de Lawrence (1997). O autor proporciona um histórico breve e interessante, delineia adequadamente o problema de previsão do mercado e porque as redes são adequadas para a resolução desse problema. O artigo de Zhang, embora seja de 1998, proporciona uma visão geral do uso de redes neurais em problemas de previsão (não só em finanças), fornecendo um compêndio de resultados obtidos por diversos autores, a partir de uma bibliografia bastante extensa. Pela amplitude que cobre, é referência obrigatória, e boa parte das conclusões a que chega serão mencionadas mais adiante.

Partindo daí, é interessante citar alguns trabalhos genéricos, quase teóricos, sobre como modelar e projetar redes neurais para prever séries temporais de ativos financeiros. Boyd e Kaastra (1997) definem 8 etapas para a construção de um bom

[3] A extinta revista "Negócios Exame" publicou em 2000 a reportagem "Papo Cabeça" (edição 02) onde mencionava o Banco Votorantim como um usuário de redes neurais na previsão de mercado. Conversas do autor com pessoas que trabalham na referida instituição levaram à conclusão de que a reportagem talvez tenha sido apenas uma questão de marketing, e que se o banco realmente utilizou Redes Neurais, já as abandonou.

[4] Notadamente Freitas & Souza (2002) que aplicam redes neurais à precificação de opções, Raposo e Cruz (1999), que tentam realizar análise fundamentalista com redes neurais, e Lazo Lazo (2000), cujo trabalho é o mais bem desenvolvido, que usa redes neurais e algoritmos genéticos em carteiras de ações. Há ainda o trabalho de McNelis (1996), que não é um autor brasileiro, e usou redes neurais para estudar determinados padrões de comportamento do mercado brasileiro à época da chamada crise "Tequila", ocorrida em 1995 por conta da desvalorização do peso mexicano.

[5] Reproduzido em Zeki-Susac (1999).

modelo de previsão, e Yao e Tan (2002 e 2002b) determinam 7 passos para a construção de modelos de previsão com redes neurais. De um modo geral, ambos enfatizam a importância do adequado pré-processamento dos dados, da escolha e teste dos parâmetros de treinamento da rede, da determinação de dados de treinamento, validação e teste dos resultados. A maioria das recomendações desses trabalhos serão utilizadas no capítulo 6.

Estreitando o foco, há uma diversidade de trabalhos de aplicação de redes neurais a previsão em finanças. Alguns desses trabalhos, pelo escopo e fôlego, valem ser citados: Refenes (1995) é talvez o primeiro trabalho denso sobre a aplicação das redes ao mercado de capitais – o autor traz capítulos sobre diversos tipos de problemas, desde previsão de falências até de preços de ações, detalhando técnicas de pré-processamento, definição de janela de experimentação, etc., quase sempre com bons resultados das previsões; Zeki-Susac (1999), que testa diversas configurações de redes neurais de maneira controlada sobre o mesmo problema (previsão dos preços das ações da IBM) e conclui que as redes *Backpropagation* são as melhores; e Shadbolt e Taylor (2003), cujo livro trata de todo o processo de previsão em finanças, discutindo técnicas estatísticas preliminares, aplicações de redes *Backpropagation*, o uso de comitês de redes neurais, técnicas de otimização e micro-modelagem de mercado – os autores são donos de uma empresa que comercializa, por assim dizer, os resultados de suas previsões.

Além destes trabalhos de maior extensão e portanto mais densos, há uma grande variedade de *papers* de *journals* internacionais, com aplicações variadas e metodologias idem. Boa parte destes trabalhos apresenta bons resultados em termos de retornos obtidos a partir das previsões realizadas. A amplitude de problemas a que as redes neurais são aplicadas permite constatar a flexibilidade e robustez.

A tabela 1 a seguir traz um compêndio destes trabalhos. Estão descritos o tipo de aplicação com que o artigo lida, o tipo de rede neural que usa para resolver o problema (incluídos detalhes de metodologia relevantes) e os tipos e quantidades de *inputs* aplicados às redes. Estes detalhes são importantes para poder traçar linhas comuns à maioria dos trabalhos, e tomar indicações para o projeto do sistema de previsão baseado em redes neurais do capítulo 6.

Tabela 1: Pesquisa Prévia de Aplicações de Redes Neurais

Trabalho	Objetivo	Rede Neural	*Inputs*
Armano et al., 2002	Previsão de ações do CAC 40 (índice da Bolsa de Paris).	Rede *Backpropagation* , com momentum de treinamento e 4 camadas.	Preços passados de 10 dias.
Buscema e Sacco, 2000	Previsão de taxas de câmbio e títulos de renda fixa de vários países.	Rede *Backpropagation* , 80-12-12-1, incorpora propriedades como *Auto-momentum* , "Regra de Freud" e "Regra de Jung" para alterar a função objetivo da rede.	80 Preços passados e Indicadores técnicos.
Chan et al., 2002	Previsão dos preços de ações da Bolsa de Shangai.	Rede *Backpropagation,* algoritmo de treinamento por gradiente conjugado, inicialização de pesos por regressão múltipla.	10 Indicadores técnicos.
Chen et al, 2001	Previsão de oportunidades de arbitragem no Nikkei 225 (Bolsa de Tóquio).	Rede *Backpropagation,* nº de neurônios na camada escondida = (nº *inputs* + nº *outputs*)/2.	Preços passados.
Giles et al., 1997	Previsão de taxas de câmbio de várias moedas.	Rede Recorrente 3-5-2.	Preços passados agrupados por Mapas de Kohonen.
Harland, 2000	Previsão do futuro de T-Bond (título de renda fixa dos EUA).	Redes *Backpropagation* , com Algoritmos Genético para treinamento e otimização da topologia da rede.	5 Indicadores técnicos.
Kalyvas, 2001	Previsão dos valores do FTSE e S&P 500.	Redes *Backpropagation* resiliente, com Algoritmos Genéticos para treinamento e otimização da topologia da rede.	Preços Passados e indicadores técnicos tratados com PCA.
Lazo Lazo, 2000	Prever os retornos de ações da Bovespa, e usá-los como *input* para um sistema de AGs para alocação da carteira.	Redes *Backpropagation,* com e sem Filtro de Kalman, 8~12 neurônios na camada escondida.	Preços passados dos 10 últimos dias..
Moisão e Pires, 2000	Previsão de preços das ações da AT&T.	Sistema Conjugado: Mapas Auto-Organizáveis de Kohonen e Rede de Boltzmann.	Preços em D-1.
Raposo e Cruz, 2000	Previsão dos retornos de ações do setor têxtil na Bovespa.	Rede *fuzzy* -neural com 5 camadas.	Dados trimestrais de balanços das empresas.
Refenes (Ed.), 1995	Previsão de diversos ativos financeiros (taxas de câmbio, títulos de renda fixa, ações).	Redes *Backpropagation* (diversas configurações).	Conjunto de preços passados, indicadores técnicos e indicadores fundamentalistas.
Shadbolt e Taylor (Eds.), 2003	Construir um sistema de previsão de retornos de títulos de renda fixa.	Combinação de ferramentas estatísticas com modelos de redes neurais *Backpropagation* trabalhando em "comitê".	Preços passados e dados econômicos tratados com PCA..
Skabar e Cloete, 2001	Previsão dos valores do S&P 500, Dow Jones, Nasdaq.	Redes *Backpropagation* , com Algoritmos Genético para treinamento e otimização da topologia da rede, função objetivo não-contínua.	2 Médias Móveis de preços, curto e longo prazo.
Thawornwong et al., 2002	Previsão de preços de 3 ações do Dow Jones.	2 Tipos: Rede *Backpropagation* resiliente, 10~20 neurônios na camada escondida & Rede Recorrente, 10~13 neurônios na camada escondida.	8 Indicadores técnicos.
Verleyesen et al., 1999	Previsão do índice SBF 250 (Bolsa de Paris).	Rede de Função de Base Radial (RBF).	Preços passados manipulados com CCA (*Curvilinear Component Analysis*).
Watada e Oda, 2000	Previsão de preços de 15 ações da Bolsa de Tóquio.	Combinação de Rede de Hopfield (probabílistica) e Rede de Boltzmann.	Preços em D-1.
Xu e Cheng, 1997	Previsão de taxas de câmbio de várias moedas.	Redes de Função de Base Radial (RBF), com regularização por função *Softmax* .	Indicadores técnicos.
Zeki-Susac, 1999	Prever os retornos das ações da IBM.	Vários tipos: *Backpropagation* , Função de Base Radial (RBF), Recorrente, Modular, *General Regression* , Probabílistica. Usa técnicas de poda de rede.	Indicadores fundamentalistas e técnicos, Preços passados

Como é possível constatar da tabela, o algoritmo Backpropagation é o mais utilizado, com diversas metodologias e características diferentes. De um modo geral, os parâmetros são definidos de maneira heurística, tanto o número de camadas escondidas quanto o número de neurônios dentro dessas camadas, além de taxas de aprendizagem e momentum. São poucos os autores que trabalham com técnicas mais avançadas dentro do paradigma da retropropagação, como regularização, *early stopping*, função *softmax*, *Levenberg-Marquardt*, entre outras. Aqueles que o fazem normalmente constatam a melhora sensível dos resultados obtidos. Assim, no capítulo 6 algumas dessas técnicas serão introduzidas e aplicadas. Quanto aos *inputs*, a variação entre diferentes trabalhos é bastante pronunciada, indo desde oitenta até apenas um. De um modo geral, a maioria trabalha com combinações de preços passados e indicadores técnicos. Aqui há indicações menos claras sobre como proceder, logo a perspectiva adotada no capítulo 6 será proprietária, e explicada adiante.

5.3. Algoritmos Genéticos em Finanças

Embora menos aplicados que as redes neurais, os algoritmos genéticos também foram largamente estudados em aplicações à área de finanças. Como foi visto no capítulo 4, essa ferramenta é excelente para problemas onde há um espaço de busca muito grande a ser varrido, e tal busca deve ser realizada em curto período de tempo. Assim, não é surpresa que as três principais aplicações de AGs em finanças encontradas na literatura sejam justamente desse tipo. Uma classificação destes trabalhos pode ter a seguinte estrutura:

- *Seleção de Portfólios*: embora a mais importante no contexto do presente trabalho, essa é a aplicação menos comum de algoritmos genéticos encontrada na literatura. Foram levantados três trabalhos que aplicam Algoritmos Genéticos ao problema de alocação ótima de ativos. O primeiro e talvez, mais importante, é o de Lazo Lazo (2000), como já citado, é bastante semelhante à proposta do presente trabalho, usando redes neurais para prever retornos de ativos e AGs para realizar a alocação, tendo como base ações do índice Ibovespa. Embora os resultados sejam bons, o autor falha ao não reconhecer determinadas

características que fazem das carteiras que obteve impraticáveis no mercado real, notadamente a questão da liquidez[6]. Há ainda o trabalho de Freedman e DiGiorgio (1993) que comparam várias heurísticas aplicadas ao problema de otimização de carteiras, concluindo que AGs estão entre as melhores, e conseguem resolver o problema adequadamente. Por fim, Korczak e Lipinski (2001), também usam algoritmos genéticos para seleção de carteiras, realizando a previsão de retornos também com modelos baseados em AGs – os resultados se mostram positivos, a partir de dados de ações da Bolsa de Paris.

- *Geração de Regras para Compra/Venda de Ativos*: é a aplicação que agrupa o maior número de trabalhos de pesquisa, no campo de aplicações de algoritmos genéticos à finanças. A premissa por trás disso é que, como os AGs são heurísticas eficientes para buscar espaços de busca muito grandes, podem ser usados para buscar regras eficientes de ordens de Compra e Venda de ativos financeiros. De um modo geral, tais trabalhos partem de uma base inicial de regras simples, e tentam buscar parâmetros e conjuntos de regras que otimizem o resultado financeiro das operações. Nesse sentido, Armano et al (2002) trabalham com "experts" genéticos, gerando dinamicamente conjuntos de regras para operação na Bolsa de Milão; Becker e Seshadri (2002) geram regras para operação no S&P 500; Thawornwong (2002), Bhattacharyya et al. (2002), Li e Tsang (1999, 2000 e 2001), e Korczak e Roger (2001) também seguem pelo mesmo caminho, de um modo geral com resultados positivos. Seguem todos o caminho de Allen e Karjalanien (1999), a referência básica nesse tipo de aplicação de Algoritmos Genéticos.
- *Otimização da topologia de Redes Neurais*: um dos problemas levantados recorrentemente na literatura é o da dificuldade de se usar funções objetivo não-triviais em redes neurais, porque, de um modo geral, os algoritmos de treinamento requerem funções diferenciáveis contínuas para funcionar adequadamente. Assim, vários pesquisadores optam por utliizar algoritmos genéticos para encontrar um conjunto quase-ótimos de pesos para a rede, ou em outras palavras, para otimizar a topologia da rede neural. Essa perspectiva híbrida mostra ter bons resultados, como evidenciado por Skabar e Cloete (2001), que a

[6] Em outras palavras: as ações que o modelo de Lazo Lazo aloca nas carteiras simuladas não poderiam ser compradas no mercado naquele momento, basicamente por ausência completa de liquidez dos papéis.

aplicam à previsão do S&P500 e do Dow Jones, e Harland (2000), que a aplicou a dados de futuros de taxa de juros.

De um modo geral, as aplicações de algoritmos genéticos a finanças se mostram eficientes, por fazer uso da característica fundamental dos algoritmos genéticos: a capacidade de buscar com rapidez e eficiência espaços de estado bastante amplos. Contudo, justamente por serem menos comuns, há menos indicações de como projetar um modelo de AGs eficiente, e portanto o cuidado na modelagem do capítulo 6 terá de ser redobrado.

5.4. Pesquisa Prévia: Que Caminhos Seguir?

Após descrever sucintamente a pesquisa prévia lidando com aplicações de redes neurais e algoritmos genéticos à finanças, algumas conclusões podem ser tiradas, na tentativa de encontrar guias úteis para seguir e armadilhas para evitar.

5.4.1. Pontos Fortes

Os principais benefícios da análise da pesquisa prévia são relativos ao campo de estudo e à metodologia usada. De um modo geral, é possível concluir que as Redes Neurais são capazes de lidar com dados incertos e com ruído, de uma forma robusta e ao mesmo tempo flexível. Portanto, são ferramentas adequadas para a previsão dos retornos de ativos no mercado.

Uma análise comparativa dos trabalhos mostra que o algoritmo Backpropagation é o mais utilizado para aplicações em finanças. Embora isto possa ser atribuído à facilidade de se trabalhar com ele, e sua operação ser razoavelmente intuitiva, não necessariamente se poderia afirmar que é, *a priori*, o melhor tipo de rede para o problema. Contudo, o trabalho que comparou diversos tipos de redes de maneira controlada e cuidadosa (Zeki-Susac, 1999), conclui ao fim que realmente, o algoritmo é o que apresenta melhores resultados. Embora longe de ser definitivo, este representa um guia claro sobre que caminho seguir na modelagem. Além disso, a existência de *papers* genéricos sobre a questão proporciona guias importantes para o

projeto de um modelo de previsão de ativos usando redes neurais (embora Boyd e Kaastra, 1997 e Yao e Tan, 2001 e 2001b, sejam as principais referências, Refenes, 1995 e Shadbolt e Taylor, 2003, pela sua amplitude, também podem ser considerados referências). Desde o pré-processamento dos dados até a análise e validação final dos resultados, há "dicas" específicas sobre como proceder. Boa parte delas vai ser aplicada no capítulo 6.

Quanto ao uso de AGs para seleção de ativos e alocação, o escopo dos trabalhos é mais limitado, mas a análise permite concluir que são sim adequados para resolver o problema de Markowitz, inclusive com vantagens sobre outras heurísticas (Freedman e DiGiorgio, 1993). Além disso, no mesmo trabalho, foi usado como função objetivo do Algoritmo Genético o índice de Sharpe (para avaliar a *utilidade*, no contexto definido no capítulo 4, dos cromossomos), o que proporciona uma importante guia para os modelos a serem desenvolvidos no capítulo 6.

5.4.2. Limitações

Algumas das limitações do uso de Redes Neurais artificiais na previsão do mercado de ações são: primeiro, elas requerem um elevado número de casos passados em seu treinamento (Zeki-Susac, 1999), o que pode ser um problema em mercados emergentes com histórico pequeno, como é o caso do Brasil; segundo, a melhor arquitetura de rede para a resolução do problema é desconhecida (o trabalho de Zeki-Susac, 1999 é uma tentativa de resolver a questão), embora talvez isso seja irrelevante (caso se adote a perspectiva de Shadbolt e Taylor (Eds.), 2003: usar comitês de redes, aplicando diversas arquiteturas diferentes); terceiro, o projeto de redes e dos dados tem de ser extremamente cuidadoso, sob pena de os resultados serem enganosos (Refenes (Ed.), 1995; Boyd e Kaastra, 1996).

A primeira limitação está ligada ao problema de disponibilidade de dados: como citado, dados diários de qualidade não são facilmente obtidos. Vale dizer que no Brasil, algumas das empresas de maior relevância no mercado de ações atual não existem há muitos anos (exemplos: Telemar, a ação de maior volume atualmente na Bovespa, existe apenas desde 1999; Globocabo idem). A segunda limitação, embora relevante, aparentemente pode ser contornada sem grandes problemas: talvez não

seja necessário encontrar a "melhor" rede neural – o esforço para tanto pode ser muito maior do que o ganho marginal obtido. No caso, adota-se uma perspectiva heurística para o problema. A terceira limitação talvez seja a mais importante de todas: a maior parte dos estudos anteriores pesquisados utiliza de um conjunto diferente de características para os dados e topologias de rede. Os trabalhos de Boyd e Kaastra (1996) e Yao e Tan (2001 e 2001b) fornecem guias genéricos para a aplicação de redes neurais à previsão de séries temporais em finanças, e permitem evitar muitas das "armadilhas" em que pesquisadores anteriores podem ter caído. Ainda assim, muito do trabalho depende da adequada experimentação pelo autor com o modelo.

Vale ainda dizer que as redes neurais padecem, desde sua criação lá nos idos dos anos 50, de um problema: seu aspecto de caixa-preta (Haykin, 2000). Essa é uma crítica constante na literatura de aplicações: embora muitas vezes proporcionem bons resultados, ninguém consegue apontar as razões inerentes para tanto[7]

No caso dos algoritmos genéticos, duas limitações principais podem ser levantadas: (i) a ausência de trabalhos amplos e detalhados sobre a aplicação de AGs ao problema de seleção de carteiras (embora Lazo Lazo, 2000, os tenha utilizado, os resultados que obteve não são confiáveis, basicamente por questões de liquidez do mercado); (ii) a ausência de plataformas comuns de *software* para uso de AGs, que permitam ao pesquisador gastar tempo no modelo e não no código do programa (diferentemente do caso das Redes Neurais, onde o *Toolbox* para *MATLAB* é claramente a plataforma mais utilizada pelos pesquisadores, sendo flexível o bastante para acomodar diversos tipos de modelos).

A primeira limitação é aparentemente incontornável *a priori*: espera-se que os resultados deste trabalho possam contribuir para a questão. A segunda idem: não é possível identficar uma plataforma, contudo aquela que será utlizada é a disponível. Não é escopo deste trabalho tratar deste segundo problema, embora este seja de relevância para o campo de inteligência artifical. De todo modo, ao levantar a limitação, busca-se, simplesmente, saber antecipadamente em que terreno se movem os resultados obtidos.

[7] Entre outros, citam o problema: Armano et al. (2002); Buscema e Sacco (2000); Harland (2000); Lawrence (1997). Alguns dos autores que usam AGs para otimizar topologia de redes neurais tentam dar respostas ao problema. É possível citar Korczak e Roger (2001) nesse sentido, embora os resultados sejam, no máximo, inconclusivos.

5.4.3. Conclusões

Da pesquisa prévia, é possível tirar as seguintes conclusões: (i) Redes Neurais são métodos eficientes na tarefa de prever o mercado de ações, mas não há uma "receita" que permita ligar determinados tipos de problemas a determinadas metodologias de previsão; (ii) o algoritmo mais aplicado para a previsão de retornos do mercado de ações é o Backpropagation, e coincidentemente, no trabalho mais detalhado e que analisou diversos tipos de redes, foi justamente ele o que apresentou melhores resultados (Zeki-Susac, 1999); (iii) a medida de avaliação mais comumente utilizada é a taxa de acurácia (ou acertos da previsão feita pelas redes); (iv) os principais benefícios das redes neurais vêm da sua capacidade de fazer previsões mesmo com dados não precisos e incertos, ou em dados com alta taxa de ruído; (v) a combinação de redes neurais com outros métodos de inteligência artificial é possível, e as aplicações conjuntas com algoritmos genéticos são muitas; (vi) o uso de algoritmos genéticos para alocação de carteiras não é um tópico tão comum na pesquisa prévia, mas seus benefícios também são claros, dada sua capacidade de varrer rápida e eficientemente espaços de busca muito grandes, inclusive com vantagens sobre outras heurísticas (Lazo Lazo, 2000; Korczak e Lipinski 2001; Freedman e DiGiorgio, 1993).

Muitos autores enfatizam a necessidade de trabalhar com muitos dados para conseguir resultados mais consistentes com as redes neurais (Shadbolt e Taylor (Eds.), 2003; Zeki-Susac, 1999). Contudo, não está claro se isto realmente melhoraria os resultados, dado que as dinâmicas econômicas (que supostamente condicionam os preços dos ativos financeiros) são mutáveis ao longo do tempo. Como enfatizam Armano et al. (2002), uma perspectiva melhor seria quebrar o problema de previsão em dois: um macro, na tentativa de descobrir a "tendência" geral do mercado, e outro micro, de trabalhar com os dados de alta frequência do cotidiano. Outra recomendação comum na pesquisa prévia trata de se evitar o chamado "*data snooping*", ou seja, adaptar os dados ao modelo, e não vice-versa. Buscema e Sacco (2000) enfatizam a necessidade de se testar o modelo com inúmeras combinações de conjuntos de dados de teste e validação, dado que o tamanho da janela de dados usada para validação pode influenciar decisivamente os

resultados obtidos. Vale dizer que determinadas características que podem ser usadas com o algoritmo Backpropagation, como o "*Early Stopping*" e a Regularização (Demuth e Beale, 2001) permitem contornar esse problema.

Além desses cuidados com o tratamento dos dados, é possível dizer que ainda há muito espaço no campo de aplicações de redes neurais e algoritmos genéticos aos problemas de previsão do mercado e alocação de carteiras. Está claro que diversos algoritmos de redes neurais podem, e devem, ser testados nesse campo: por exemplo, as aplicações de Mapas Auto-Organizáveis de Kohonen são raras (é possível citar Moisão e Pires, 2001 e Giles et al., 1997), redes probabílisticas (apenas Zeki-Susac, 1999 faz uso delas), além de tratamentos de redes em comitê, talvez a melhor perspectiva para o problema (apenas Shadbolt e Taylor (Eds.), 2003, citam esse tipo de ferramenta). No campo de algoritmos genéticos, ainda são raros os trabalhos que os aplicam ao problema de alocação de carteiras. Isso talvez reflita uma característica mais geral da área de pesquisa em finanças em geral: o problema de previsão tem bastante mais "apelo" intelectual que o problema de alocação, e por isso muito mais esforços, recursos, e consequentemente, resultados publicados, são dedicados ao primeiro. A importância do trabalho de Markowitz e Sharpe não pode ser relevada, contudo, e o que fica claro é que uma alocação adequada pode ser tão, ou mais, adequada, que uma previsão correta do mercado[8].

[8] O autor já leu estudos realizados pelo banco Merrill Lynch que atribuíam 70% dos retornos totais de um portfólio à correta alocação, 25% à correta escolha de papéis individuais, e apenas 5% ao correto *timing* de compra e venda desses papéis. Na falta de trabalhos acadêmicos aprofundados sobre a questão na bibliografia, fica apenas o registro informal.

6. MODELOS & EXPERIMENTOS

"Prever é muito difícil. Especialmente o futuro."
Niels Bohr[1]

6.1. Introdução

Após desenvolver as teorias necessárias para a construção de um modelo, baseado em redes neurais e algoritmos genéticos, de gestão de carteiras de ações, vai-se agora mostrar, passo a passo, os detalhes de tal modelo e sua aplicação a dados reais de ações do mercado brasileiro.

O capítulo está estruturado do seguinte modo: inicialmente vai-se discutir os dados, sua escolha e o tratamento preliminar dado a eles. A seguir, será apresentado o modelo básico de previsão de retornos, que utiliza redes neurais para, a partir do treinamento usando dados passados, realizar previsões. No centro deste modelo, está o algoritmo de treinamento usado pela rede. Dada a ausência de paradigmas bem-estabelecidos, conforme discutido no capítulo 5, serão explicadas e testadas cinco variações do algoritmo *Backpropagation*, buscando determinar qual algoritmo traz os melhores resultados (e se há variações relevantes entre as ações, por exemplo, para a ação A o algoritmo X é o melhor, enquanto para a ação B o algoritmo Y traz performance superior). Além disso, serão testadas diversas configurações de neurônios na camada escondida, de modo a determinar qual a influência disto na performance preditiva do modelo. Ao fim, será escolhido o algoritmo de treinamento que apresentar melhores resultados (se é que há diferenças) e as previsões geradas por tal algoritmo serão armazenadas, para serem alimentadas no modelo de alocação.

Além da previsão de retornos, a previsão de riscos também é parte integrante do trabalho de alocação e gestão de carteiras de ações. Assim, um modelo GARCH vai ser apresentado e implementado, para realizar a previsão de risco para as ações. Tal modelo será implementado usando uma janela deslizante, de modo a capturar o comportamento dinâmico e variante no tempo que o risco apresenta. Novamente, os resultados serão armazenados para serem alimentados ao modelo de alocação.

[1] Apud Stewart, I. *Does God Play Dice?*, Nova York: Penguin Books, 1997.

Por fim, o modelo de alocação será desenvolvido, fazendo uso de algoritmos genéticos para a busca, no espaço de combinações possíveis, das carteiras que otimizam a relação risco-retorno de um investidor hipotético. A função de utilidade aplicada será o índice de Sharpe (vide seção 2.7), uma maneira simples e eficiente de computar a relação risco-retorno. O modelo será atualizado dinamicamente, replicando o processo de gestão que ocorreria no dia-a-dia: ao fim de um dia, os modelos de previsão, tanto de risco quanto de retorno, são atualizados, e são buscadas novas carteiras ótimas (ou quasi-ótimas) para o dia seguinte. Ao fim, o resultado financeiro desta gestão será computado, na tentativa de determinar se efetivamente os modelos são capazes de gerar bons retornos.

6.2. Dados

A primeira questão que se coloca, antes mesmo da construção dos modelos propriamente ditos, é a dos dados que serão usados para testá-los. De nada adianta desenvolver modelos complexos e sofisticados se não há dados para o teste, validação e calibração dos modelos. Seria um esforço intelectual com pouco valor prático – justamente o contrário daquilo que se propõe o presente trabalho.

Dito isto, foram escolhidas 25 ações, negociadas na Bovespa (Bolsa de São Paulo). Dado o pequeno tamanho do mercado brasileiro, não é viável trabalhar com mais ações, visto que uma boa parte dos resultados seria irreal e inaplicáveis na prática – por exemplo, o modelo ia determinar a alocação de recursos em uma ação cuja liquidez é extremamente restrita[2]. De um modo geral, o tipo de previsão e modelagem de que trata este trabalho é aplicável apenas em mercados bastante líquidos, onde a informação flui livremente (ou seja, não há informação privilegiada por trás das cotações), onde é possível comprar e vender sem alterar as cotações de mercado, e onde há liquidez suficiente para que, dado uma oferta (um *bid*) e uma demanda (o correspondente *ask*), seja possível entrar e sair do mercado a qualquer momento.

[2] Lazo Lazo (2001) por exemplo, incorre neste erro: constrói um modelo que determina a compra de ações como Bemge ON e Ericsson PN – papéis cuja liquidez, à época analisada, era extremamente restrita, ou seja, não podiam ser compradas/vendidas em quantidades razoáveis, sem expressivas alterações nas cotações. Notadamente no caso destas duas ações, os papéis não são mais disponíveis – as empresas tiraram suas ações da Bolsa (no caso do Bemge, o banco foi privatizado e incorporado pelo Itaú).

6.2.1. Escolha

A escolha das ações a serem utilizadas para a experimentação dos modelos levou em conta alguns fatores fundamentais, quais sejam:

- *Liquidez*: talvez seja o aspecto mais importante dentre todos – só há sentido em prever o comportamento de uma ação se há a possibilidade de se comprar e vender livremente esta ação em quantidades razoáveis. Se o papel é ilíquido, qualquer previsão cai por terra, dado que o *spread* do *bid and ask* é tão largo que o retorno almejado é perdido no momento da execução da ordem de compra ou venda. Em termos simples, o *bid* representa o preço que o melhor comprador está disposto a pagar, e o *ask* é o preço que o vendedor pede para realizar a venda. O *spread* é a diferença entre estes dois preços. As ordens são executadas conforme compradores se dispõem a pagar o preço pedido, ou vendedores se dispõem a vender no preço que os compradores estão pagando. Esta relação dinâmica determina o movimento das cotações de uma ação ao longo do dia. Quando há liquidez, os *spreads* são estreitos, permitindo a negociação com facilidade. Os fatores determinantes da liquidez são vários, podendo ser de caráter fundamental (por exemplo, as ações da Petrobrás, tanto ON quanto PN, têm alta liquidez devido à importância da empresa para a economia brasileira), de caráter puramente técnico (a ação de maior liquidez da Bovespa atualmente é Telemar PN - TNLP4 -, devido ao fato de haver um ativo mercado de opções sobre esta ação[3]) ou ainda devido à "moda" (no ano de 2000, a ação de maior liquidez era Globocabo PN – PLIM4 – devido à bolha de Internet). Todas as 25 ações escolhidas como dados-base deste trabalho possuem boa liquidez.
- *Existência de Histórico*: este aspecto está diretamente ligado à utilização de modelos de redes neurais para a previsão de retornos. É necessário treinar a rede neural com padrões passados, de modo que ela consiga capturar estes padrões na sua topologia de pesos, e quando apresentada com novos dados, possa realizar boas previsões para o futuro. Assim, é absolutamente fundamental a existência de um longo histórico de cotações para cada uma das ações. Para os fins deste trabalho, estão sendo utilizados dados entre 10/05/1999 e 10/10/2003, formando um conjunto de 1.100 cotações para cada uma das 25 ações escolhidas. Explicitamente, optou-se

[3] No mercado de opções, são negociados os direitos de comprar (chamados de *calls*) e de vender (*puts*) uma determinada ação em uma data futura pré-determinada.

por não usar mais dados (o que talvez implicasse em melhores resultados de previsões), pelo fato de que seria necessário usar cotações anteriores a 1999. Sabidamente, no começo daquele ano, o Brasil passou por uma mega-desvalorização cambial, alterando estruturalmente as condições da economia brasileira, e conseqüentemente do mercado acionário – logo, padrões de preços anteriores a 1999 não agregariam informação de boa qualidade para a previsão do mercado nas condições atuais.

- *Setor da Economia*: além da importância da liquidez e de um histórico estabelecido de cotações, buscou-se escolher ações de vários setores econômicos diferentes, de modo a estabelecer uma base ampla de diferentes empresas para a constituição de portfólios.

Dados estes requisitos, foram escolhidas 25 ações para serem objeto de análise e teste dos modelos desenvolvidos. A seguir, são apresentados as ações, os respectivos códigos e setores econômicos em que atuam as empresas, além de um campo indicando se são empresas estatais ou privadas:

Código	Nome	Setor	Estatal
ARCZ6	Aracruz ON	Papel e Celulose	
BBAS3	Banco do Brasil ON	Bancos	X
BBDC4	Bradesco PN	Bancos	
BELG4	Belgo-Mineira PN	Siderurgia	
CMIG4	Cemig PN	Energia	X
CPLE6	Copel PNB	Energia	X
CRUZ3	Souza Cruz ON	Consumo	
CSNA3	CSN ON	Siderurgia	
CSTB4	Siderúrgica Tubarão PN	Siderurgia	
EBTP4	Embratel Part. PN	Telecomunicações	
ELET6	Eletrobrás PNB	Energia	X
EMBR4	Embraer PN	Aviação	
FFTL4	Fosfértil PN	Fertilizantes	
ITAU4	Itaubanco PN	Bancos	
ITSA4	Itaúsa PN	Holding	
LAME4	Lojas Americanas PN	Varejo	
PCAR4	Pão de Açúcar PN	Varejo	
PETR3	Petrobrás ON	Petróleo	X
PETR4	Petrobrás PN	Petróleo	X
SBSP3	Sabesp ON	Saneamento	X
TLPP4	Telesp PN	Telecomunicações	
TNLP4	Telemar PN	Telecomunicações	
TSPP4	Telesp Celular PN	Telecomunicações	
USIM5	Usiminas PNA	Siderurgia	
VALE5	Vale do Rio Doce PNA	Mineração	

Tabela 2: Ações Escolhidas para Experimentos com Modelos

Para realizar as previsões de preços futuros das ações, não apenas os preços passados serão utilizados. Partindo do princípio explicitado teoricamente no modelo *APT* (vide seção 2.5.3), foram escolhidos alguns fatores que podem auxiliar na definição da dinâmica de comportamento dos preços das ações, e portanto poderiam conter informação útil para a realização de previsões. Assim, foram levantadas algumas outras séries temporais, que supostamente poderiam auxiliar a tarefa de previsão dos preços futuros. São elas:

- *Cotação máxima e mínima*: além da cotação média, que representa o preço efetivo da ação *X* na data *Y*, também serão usadas as cotações máxima e mínima, que agregam a informação de quais os máximos e mínimos que o mercado esteve disposto a pagar por um determinado papel.
- *Volume Financeiro negociado*: representa o volume total financeiro negociado da ação *X* naquele dia. Pode trazer informação sobre tendência de movimento da ação: quando uma alta (ou baixa) relevante está prestes a acontecer, normalmente existe um aumento do volume financeiro negociado.
- *Cotação do dólar*: após 1999, a cotação do dólar (ou seja, a relação entre o real e o dólar) flutua livremente no mercado, e dá indicações fortes dos rumos econômicos do país e portanto dos mercados, podendo indicar também as tendências para a Bolsa de Valores. Neste trabalho, será utilizada a cotação diária da PTAX, que é a oficial do Banco Central do Brasil para transações financeiras[4].
- *Taxas de Juros de curto e longo prazo*: além da cotação do dólar, outro preço fundamental da economia é dado pela taxa de juros. Assim, é útil saber a evolução das taxas de juros da economia, que podem indicar crescimento ou recessão, afetando diretamente as possibilidades de crescimento dos lucros das empresas e portanto da valorização de suas ações. Para os fins deste trabalho, serão utilizadas duas *proxies* da taxa de juros da economia: uma para indicar a taxa de juros de curtíssimo prazo, outra para indicar a taxa de juros de longo prazo. São dadas pelas cotações de *swap pré x DI* para 30 dias (curto prazo) e 360 dias (longo prazo)[5]. Essas cotações indicam efetivamente as taxas em que as empresas de grande porte conseguem tomar empréstimos para financiar seus investimentos.

[4] Em termos técnicos, será utilizada a cotação de venda do dólar comercial, apurada através da transação PTAX800 do Sisbacen, o Sistema de Informações do Banco Central.

[5] Ambas taxas são apuradas diariamente no mercado e divulgadas pela BM&F – Bolsa de Mercadorias & Futuros.

➢ *Índices de Bolsa de Valores nacionais e internacionais*: além de dados específicos de cada ação, é útil ter informações sobre o mercado em geral, tanto o brasileiro quanto o americano, o maior mercado de ações do mundo. Para tanto, séries de dados do índice Bovespa (ou Ibovespa), do Dow Jones Industrials[6], do S&P 500[7] e do Nasdaq[8] serão utilizadas para agregar informação sobre a performance das bolsas de valores, que se supõe podem afetar os preços das ações escolhidas.

Assim, para cada uma das 25 ações, ao todo há 11 séries temporais que se espera, produzam informação que ajude o modelo de redes neurais a prever as cotações futuras. A tabela a seguir apresenta cada uma das séries usadas:

Dado	Comentário
Preço Médio da Ação	Indica o valor da ação numa data específica.
Preço Máximo da Ação	Indica o valor mais alto da ação, naquela data - banda superior que o preço atingiu.
Preço Mínimo da Ação	Indica o valor mais baixo da ação, naquela data - banda inferior que o preço atingiu.
Volume Negociado	Indica o volume financeiro movimentado pela ação - denota interesse de outros investidores nela.
Cotação do Dólar	Preço fundamental da economia brasileira, indica interesse de investidores estrangeiros no país (e vice-versa).
Taxa de Juros de 30 dias	Indica perspectivas econômicas (capacidade de crescimento) de curto prazo.
Taxa de Juros de 360 dias	Indica perspectivas econômicas de longo prazo.
Índice Bovespa	Indica tendência geral da Bolsa de Valores de São Paulo.
Índice Dow Jones Industrials	Indica tendência das principais ações da Bolsa de Nova York.
Índice S&P 500	Indica tendência do mercado acionário americano como um todo.
Índice Nasdaq 100	Indica tendência das empresas de tecnologia americanas.

Tabela 3: Tipos de Dados usados como *Inputs*

Na seção 6.3.2, vai ser discutido o pré-processamento realizado antes de tais dados serem introduzidos no modelo, e ficará claro como apenas a informação que tem relevância na explicação das cotações de cada ação será efetivamente utilizada. De todo modo, como não há como saber *a priori* qual informação é mais relevante,

[6] Índice das 30 principais empresas negociadas na Bolsa de Nova York (NYSE).

[7] Índice calculado pela Standard & Poors, com as 500 principais empresas do mercado acionário americano. É geralmente tomado como a melhor *proxy* da performance do mercado.

[8] Índice das 100 principais empresas de tecnologia.

buscou-se levantar a maior quantidade de dados possíveis, através de uma análise qualitativa. O refinamento quantitativo de tal análise será realizado posteriormente.

6.2.2. Pré-Processamento

Escolhidos os dados a serem utilizados para a aplicação dos modelos, estes foram buscados em um banco de dados de séries históricas de ativos financeiros[9]. Contudo, nem todas as séries estavam perfeitamente completas, ou havia determinadas inconsistências entre elas, o que determinou a necessidade de realização de um pré-processamento dos dados, antes de estes poderem ser usados nos modelos que serão apresentados a seguir.

Os dois principais problemas eram: ausência de cotações em determinadas datas, e datas em que apenas um dos mercados opera. No primeiro caso, a causa é a ausência de negócios em determinados dias. A solução é adotar o preço médio do dia anterior como o preço daquele dia em que não houve negócios – ou seja, a melhor aproximação do valor daquela ação é o último preço médio dela. Afinal, não é possível saber a razão pela qual não houve negócios, e deve-se trabalhar com a informação que está disponível. Para o segundo problema, a solução adotada foi ignorar as cotações dos mercados nessas datas – por exemplo, no dia 31 de dezembro de 2002, o Dow Jones, S&P e Nasdaq estavam abertos, assim como o dólar, enquanto a Bovespa estava fechada, logo não há cotações nessa data para as ações. Assim, ignorou-se os dados desta data, e supôs-se que os mercados não estivessem abertos.

Finalizado este trabalho preliminar de seleção, coleta e tratamento dos dados, nas seções a seguir serão apresentados os modelos e os principais resultados obtidos.

6.3. Modelo de Previsão de Retornos com Redes Neurais

O primeiro passo do modelo de gestão de carteiras é a obtenção de previsões de retornos futuros. Dentro do quadro teórico delineado por Markowitz e Sharpe, a previsão dos retornos futuros é a etapa inicial da construção da fronteira eficiente.

[9] O autor obteve todos os dados através de uma ferramenta chamada *BDS*, gentilmente disponibilizada pela Hedging-Griffo.

Toda a construção teórica subseqüente parte da premissa de que se conhece, com boa dose de confiança, os retornos futuros dos ativos.

6.3.1. Introdução

Conforme discutido no capítulo 5, redes neurais são ferramentas ideais para a construção de modelos de previsão de ativos financeiros. Suas propriedades se adequam bem às características das séries temporais de preços de ações: flexibilidade (as redes lidam bem com situações de mudança de regime, tipicamente o que acontece com os mercados de ação, que vão da euforia ao pânico em poucos dias), robustez (há uma enormidade de situações possíveis no mercado, e o reconhecimento de padrões é tarefa absolutamente não-trivial) e capacidade de capturar padrões não-lineares complexos.

Para obter bons resultados de previsão, é fundamental realizar um bom treinamento da rede. Tal tarefa tem dois ângulos distintos, que devem ser cobertos com atenção, de modo a que os resultados sejam satisfatórios: (i) abundância de exemplos de treinamento – para que a rede possa generalizar a partir dos dados passados, deve conhecer diversos padrões diferentes; (ii) algoritmo de treinamento robusto – como são, em essência, matrizes cujos valores capturam padrões dos exemplos de treinamento, a maneira de calcular tais matrizes deve ser suficientemente poderosa para que os resultados obtidos quando são apresentados padrões desconhecidos sejam adequados. No presente trabalho, o primeiro fator já foi discutido na seção 6.2, onde se buscou levantar dados variados e em grande quantidade, para a realização do treinamento da rede.

O segundo ponto é mais sensível e de solução menos simples. Conforme discutido nos capítulos 3 e 5, a variedade de topologias de rede e algoritmos de treinamento disponíveis é enorme. Embora a conclusão, ao fim do capítulo 5, fosse pelo uso de uma estrutura de rede “alimentada adiante” (*feedforward*) com uma camada escondida e algoritmo de treinamento por retropropagação (*Backpropagation*), ainda assim há uma grande variedade de subtipos deste algoritmo que podem ser escolhidos para a realização do treinamento. Além disso, há

a questão da definição dos parâmetros de cada algoritmo, e também da topologia da rede.

Dada a ausência de paradigmas estabelecidos na literatura para a escolha da variante do treinamento por retropropagação (Zeki-Susac (1999) compara diferentes "famílias" de algoritmos – *Backpropagation*, Função de Base Radial, Probabilística, etc. – e não diferentes "subtipos" dentro de cada família), a opção será por comparar vários algoritmos diferentes, de um modo controlado, de maneira a ser possível concluir qual algoritmo proporciona melhores resultados. Os algoritmos foram escolhidos de modo a representar as diferentes perspectivas adotadas dentro do paradigma da retropropagação. Vão desde modelos simples até outros razoavelmente mais sofisticados. São eles:

- *Backpropagation com Momentum*
- *Backpropagation Resiliente*
- *Powell-Beale*
- *Quasi-Newton BFGS*
- *Levenberg-Marquardt*

O primeiro representa o desenvolvimento inicial do paradigma de retropropagação, com a adição do termo de momentum para melhoria da performance. O segundo é uma modificação do anterior, a partir da constatação de que a performance seria mais robusta com uma mudança no tipo de cálculo das derivadas parciais (vide seção 3.5 para o papel das derivadas parciais no treinamento da rede). Os três outros tipos utilizam conceitos de otimização numérica para obter melhor performance com menos recursos computacionais e de memória. Têm sofisticação crescente, e espera-se, performance idem. Cada um dos cinco será discutido em mais detalhe nas seções a seguir, quando serão apresentados os resultados obtidos quanto utilizados para previsão das séries de ações.

Quanto à questão dos parâmetros próprios de cada algoritmo, a solução é mais simples, mas por isso mesmo, menos robusta. Serão utilizados os parâmetros padrão (*default*) do *software*. Essa opção se deve à enormidade de estudos que deveriam ser realizados para a definição de parâmetros ótimos para cada algoritmo, para cada ação diferente, para cada topologia. Assim, fixando estes parâmetros, o foco fica naquilo

que é realmente importante na definição de um bom modelo de previsão por redes neurais: o algoritmo de treinamento e a topologia da rede.

Por fim, a escolha da topologia de rede: talvez o parâmetro mais difícil de controlar e definir. Na literatura de previsão de séries temporais financeiras usando redes neurais, a maioria dos trabalhos que usa o algoritmo de retropropagação traz topologias com uma camada escondida e número de neurônios nesta camada variando entre cinco e 20. Chen et al. (2001) propõe a seguinte regra: número de neurônios na camada escondida = (número de *inputs* + número de *outputs*)/2. Caso fosse aqui aplicada, o resultado seria seis neurônios na camada escondida (11 entradas, uma saída). Como esta regra não tem base teórica, e dada a variedade de possibilidades discutidas na literatura, a opção novamente foi pela experimentação controlada. Assim, para cada um dos algoritmos de treinamento e cada uma das 25 ações, serão testadas diferentes configurações de topologia de rede, variando entre **quatro e quinze neurônios na camada escondida**. Embora haja, na literatura, trabalhos que usam mais de 15 neurônios escondidos (Buscema e Sacco (2000) e Thawornwong et al. (2002) são exemplos), optou-se por não ir além pela razão de que, com um número maior de neurônios na camada escondida, a possibilidade de sofrer com o fenômeno de *excesso de ajuste* cresceria excessivamente. Como séries temporais de preços de ativos financeiros são inerentemente não-parametrizadas *a priori*, e é justamente esta característica que se busca explorar com o uso de redes neurais, existe a preocupação recorrente de não ajustar demais o modelo aos dados, sob pena de não ser possível generalizar adequadamente, para a realização de previsões futuras.

Um aspecto importante que deve ser enfatizado é o caráter estatístico dos resultados a serem obtidos. Cada vez que uma rede neural é criada, os pesos são inicializados de maneira aleatória (com média zero e variância unitária). Este processo tem influência não desprezível na performance da rede após o treinamento. Assim, é necessário repetir o processo de treinamento diversas vezes, de modo a coletar estatísticas confiáveis sobre a performance, e assim conhecer de fato as possibilidades do modelo. Portanto, cada um dos experimentos será repetido 50 vezes, de modo que médias e desvios dos resultados possam ser calculados, e um bom nível de confiança possa ser estabelecido.

Um resumo das características do modelo está dado na Tabela 4.

Parâmetro	Valor
Arquitetura	*Feedforward.*
Número de Camadas	3 - 1 camada escondida
Número de *Inputs*	11 (vide tabela 3)
Número de *Outputs*	1 - preço da ação
Algoritmo de Treinamento	*Backpropagation* com momentum
	Backpropagation resiliente
	Powell-Beale
	BFGS (quasi-Newton)
	Levenberg-Marquardt
Função de Transferência	Camada Escondida: Tangente Hiperbólica
	Camada de Saída: Linear
Neurônios na Camada Escondida	entre **4** e **15** - definição experimental

Tabela 4: Parâmetros do Modelo de Previsão de Retornos

6.3.2. Entradas, Saídas e *Targets*: Pré e Pós-Processamento

Definidas as características das redes neurais a serem utilizadas, o foco passa aos dados que serão introduzidos no modelo, qual pré-processamento necessário, e como será realizado o treinamento – ou seja, que tipo de padrão se espera que o modelo aprenda a capturar, e como deverão ser analisados os resultados obtidos.

Conforme discutido na seção 6.2.1, para cada uma das 25 ações, tem-se 1100 cotações – de 10/05/1999 até 10/10/2003. Além disso, tem-se as séries temporais dos outros dados que, supõe-se, tenham algum poder explicativo sobre os preços das ações. O paradigma de treinamento a ser utilizado está ilustrado na Figura 7.

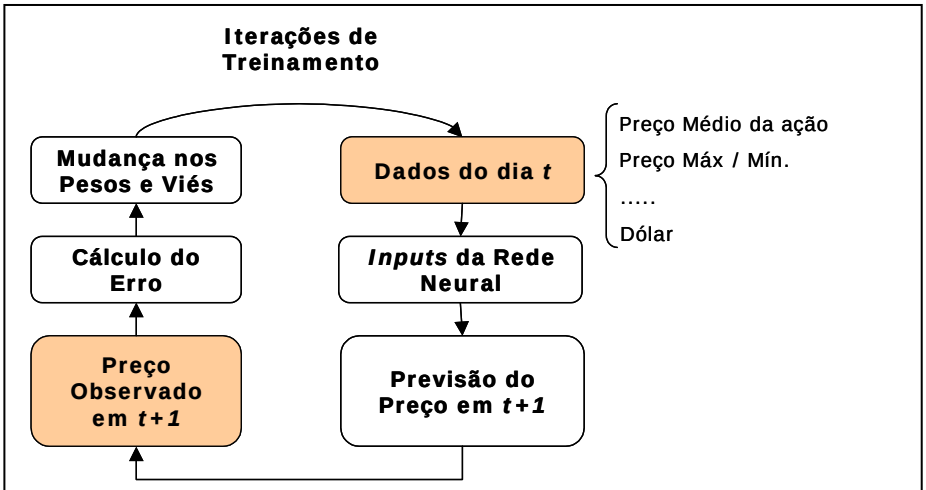


Figura 7: Fluxograma do Processo de Treinamento da Rede Neural

A partir desta formação, tem-se 1099 preços diários que serão utilizados como *target*, ou seja, como padrão que a rede neural deve aprender, sempre adiantados em um dia. E como *inputs*, entradas da rede, tem-se as 11 séries temporais discutidas na seção 6.2.1, cada uma com 1099 observações. Portanto, espera-se que, dados os valores destes onze dados na data *x*, a rede neural (já treinada) seja capaz de prever, com boa precisão, o valor do preço da ação na data *x+1*. Em essência, é isto que aqui se busca.

Antes de realizar a entrada dos dados na rede neural, é necessário realizar alguns pré-processamentos, de modo a obter uma boa performance preditiva do modelo (Demuth e Beale (2001), DeBoeck (2001), Refenes (1995) são referências no assunto). Todas as vezes que uma rede neural for criada, todos as etapas de pré-processamento serão repetidas. São dois procedimentos básicos:

- *Normalização*: o processo de consiste na normalização de todos os dados de entrada (*inputs*) de modo a terem média zero e variância unitária. Isto é necessária devido a uma peculiaridade das redes neurais: elas não trabalham bem com valores elevados – já que as funções de transferência (a função sigmóide, por exemplo) realizam o chamado *squashing*, ou seja, comprimem os dados (Haykin, 2000). Assim, a normalização dos dados facilita o trabalho de cálculos das matrizes de pesos e viés, proporcionando melhor performance. Após o treinamento, todas as saídas da rede tem de ser renormalizadas, ou seja, colocadas na média e variância originais.
- *Análise de Componentes Principais (PCA)*: em várias situações (caso deste trabalho), a dimensão do vetor de entrada é razoavelmente grande, e há alguma correlação entre os componentes do vetor (ou seja, redundância) (Demuth e Beale, 2000). Assim, é útil nestas situações reduzir o tamanho do vetor, e uma maneira eficiente de realizar este processo é através da Análise de Componentes Principais (ou *Principal Component Analysis*, *PCA*). Esta técnica traz três efeitos: ortogonaliza os componentes do vetor de entrada (de maneira a serem não-correlacionados uns com os outros), ordena os componentes ortogonais resultantes (componentes principais) de maneira que os com maior variação venham antes, e elimina aqueles que contribuem menos para a variância total do conjunto de dados. Neste trabalho, serão eliminados os componentes que explicam menos de **1%** da variação total do

conjunto de *inputs* (não se deseja reduzir excessivamente o conjunto de dados de entrada, por isso um limite baixo foi determinado). Após o treinamento da rede, todos os novos dados que são introduzidos devem ser transformados usando a matriz de componentes principais, de maneira a garantir homogeneidade de resultados.

Assim, a normalização e a análise de componentes principais auxiliam no processo de obtenção de boa performance preditiva das redes neurais a serem construídas.

Após o treinamento se faz necessário analisar os resultados. Após renormalizados para média e variância apropriados, será realizada uma Análise de Regressão entre os dados simulados pela rede e os *targets*, ou seja, vão ser comparados os preços previstos pelo modelo de rede neural e os preços reais observados no mercado, de maneira a aferir a performance das redes testadas. O resultado desta análise, em termos de **coeficiente de correlação**, será de fundamental importância na avaliação da performance de cada um dos modelos de rede neural utilizados daqui por diante. Na Figura 8, é ilustrada graficamente esta análise, com a comparação dos dados de ARCZ6 com os preços previstos por uma rede neural com 12 neurônios na camada escondida, treinada com algoritmo Levenberg-Marquardt.

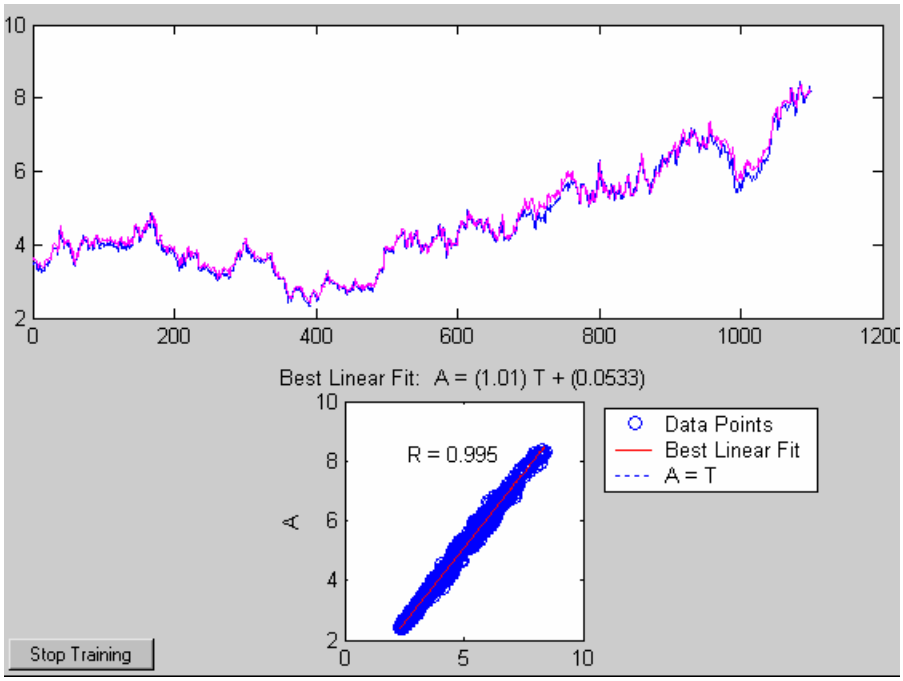


Figura 8: Previsão de Preços de ARCZ6 por uma rede Levenberg-Marquardt com 12 neurônios na camada escondida

Note-se a comparação entre os *targets* e as previsões da rede já treinada: no gráfico superior, tem-se a série de dados real (em azul) e a série de preços previstos (em magenta) – a rede consegue ajustar muito bem, a primeira vista, aos dados históricos. Esta impressão inicial é corroborada pelo segundo gráfico, que mostra a regressão linear entre os dados reais e as previsões, retornando um coeficiente de correlação de 0,9954, ou 99,54%, que é excelente.

6.3.3. Modelos de Rede Neural

Uma vez determinados os dados a serem usados como *inputs* das redes neurais, o pré-processamento a ser realizado, e como serão analisados os resultados obtidos, cabe passar aos experimentos propriamente ditos. Conforme discutido na seção 6.3.1, serão testados cinco tipos diferentes de algoritmos, e 12 quantidades diferentes de neurônios na camada escondida (entre quatro e 15). Além disso, todos os experimentos serão repetidos 50 vezes, de maneira a estabelecer o caráter estatístico efetivo dos resultados. Grosso modo, isto resulta na seguinte quantidade de redes neurais a serem criadas:

25 ações x 5 algoritmos x 12 topologias x 50 repetições = 75.000 redes[10]

De maneira a obter uma otimização da performance de generalização das redes, será adotado, em todos os experimentos, o método de *parada antecipada*[11]. Esta técnica consiste na divisão dos dados de entrada em três subconjuntos: treinamento, validação e teste. O primeiro subconjunto é usado para computar os gradientes de treinamento e os pesos e viés da rede. Durante o processo de treinamento, o erro no conjunto de validação é monitorado: inicialmente, deve decrescer, conforme melhora a performance preditiva da rede. Contudo, se a rede começa a ser excessivamente treinada e sofrer de *excesso de ajuste*[12], o erro de validação vai tipicamente começar a crescer, e quando isto ocorre repetidamente o processo de treinamento é interrompido. O erro do subconjunto de teste é usado para comparar diferentes

[10] [10] Rodando em um Pentium III de 800 MHz e 256 MB de RAM, o *script* de *MATLAB* escrito para realizar a determinação do número ideal de neurônios escondidos levou quase 3 dias ininterruptos de cálculo (aprox. 65 horas) para estimar todas as redes. Detalhes deste código podem ser encontrados no Apêndice 1.

[11] Ou *early stopping*.

[12] Também chamado de *overfitting*, vide seção 3.6.1 para detalhes.

modelos, e não é monitorado no processo de treinamento. No presente trabalho, não será utilizado. Em todos os experimentos, os dados foram divididos da seguinte maneira:

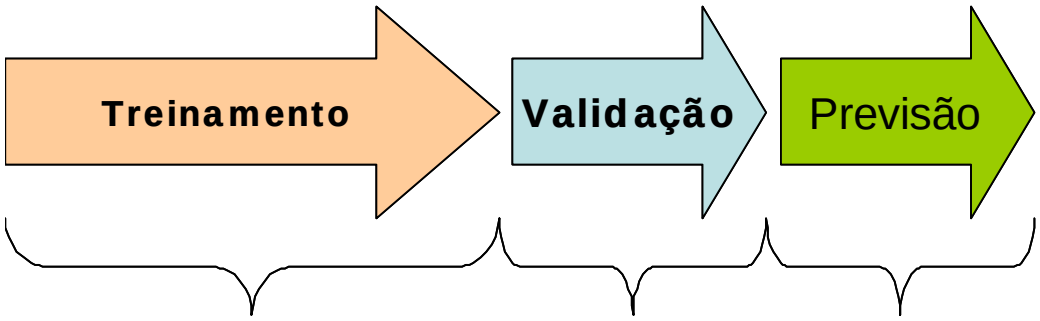


Figura 9: Divisão de Dados – Treinamento/Validação/Previsão

A subdivisão dos dados é sempre realizada após o pré-processamento. Já a performance é aferida sobre o conjunto completo de entradas.

Determinadas todas as características dos modelos a serem apresentados, a seguir serão discutidos cada um dos cinco algoritmos. Em cada subseção seguinte, serão apresentados os fundamentos do algoritmo, os resultados em termos de número ótimo de neurônios na camada escondida para cada uma das 25 ações, e a performance preditiva das redes, aferida em termos de coeficiente de correlação, também determinada para cada uma das 25 ações. Ao fim, na seção 6.3.4, serão sumarizados e discutidos os resultados e a performance preditiva das redes neurais.

6.3.3.1. Rede *Backpropagation* com *Momentum*

O primeiro tipo de rede a ser testado é também o mais simples: a *Backpropagation* com momentum. No algoritmo de treinamento por retropropagação mais simples, que faz uso da técnica de descida pelo gradiente, os pesos e viés são movidos na direção do negativo do gradiente da função erro da rede. O parâmetro básico que controla o processo de aprendizagem é a taxa de aprendizagem (vide seção 3.5.1 para detalhes). Uma maneira de melhorar a convergência do treinamento é utilizar um outro parâmetro, chamado de *momentum*, para controlar o treinamento em conjunto com a taxa de aprendizagem. Este termo permite à rede não apenas

responder ao gradiente local (na iteração *t*), como também às tendências recentes na superfície de erro.

O termo pode ser adicionado ao algoritmo de treinamento básico fazendo-se as variações dos pesos iguais à soma de uma fração da última variação e à nova mudança sugerida pela regra de retropropagação. A magnitude desta fração é definida pela constante de *momentum*. De um modo geral, a adição deste termo à equação de atualização dos pesos proporciona uma melhor performance ao processo de treinamento, tanto em velocidade quanto em erro médio (ou seja, performance da rede).

Os resultados das previsões realizadas com o algoritmo não se mostraram particularmente bons[13]. A Figura 10 traz uma a dispersão dos resultados, em termos de correlação com os *targets*:

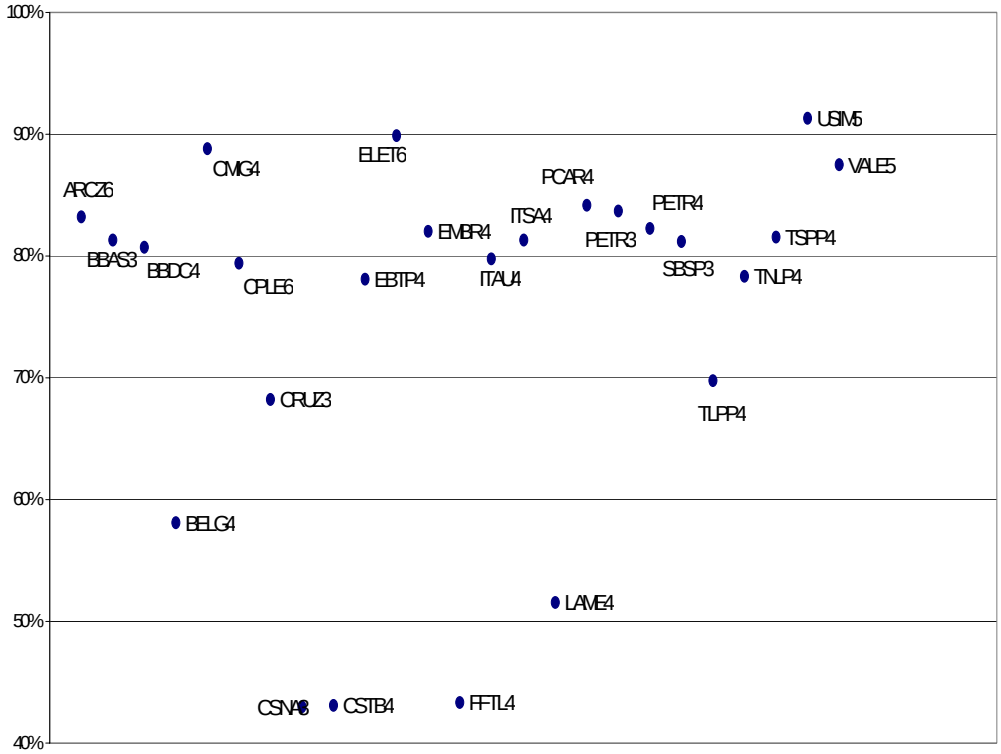


Figura 10: Dispersão Resultados para Algoritmo *Backpropagation* com Momentum

Tais resultados estão sumarizados na Tabela 5.

[13] Uma constatação que já pode ser vista na literatura: Refenes (1995) e Kalyvas (2001) estão entre os autores que testam este algoritmo para previsão de preços de ações, e afirmaram não ter conseguido resultados conclusivos.

Código	Backpropagation com Momentum	Código	Backpropagation com Momentum	Código	Backpropagation com Momentum
ARCZ6	83,25%	EBTP4	78,11%	PETR4	82,24%
BBAS3	81,27%	ELET6	89,86%	SBSP3	81,20%
BBDC4	80,73%	EMBR4	81,97%	TLPP4	69,78%
BELG4	58,04%	FFTL4	43,34%	TNLP4	78,31%
CMIG4	88,77%	ITAU4	79,78%	TSPP4	81,57%
CPLE6	79,40%	ITSA4	81,36%	USIM5	91,30%
CRUZ3	68,24%	LAME4	51,53%	VALE5	87,46%
CSNA3	42,98%	PCAR4	84,11%		
CSTB4	43,13%	PETR3	83,75%		
média	**74,86%**				

Tabela 5: Resultados Algoritmo *Backpropagation* com *Momentum*

Obviamente, a qualidade dos resultados obtidos só vai ficar clara após a comparação com todos os outros algoritmos. Ainda assim, já é possível dizer que o algoritmo de treinamento usando *Backpropagation* com *Momentum* não tem uma performance preditiva particularmente notável, com uma correlação média de 74% com os preços-alvo, a tendência é que estas previsões não tragam bons resultados, caso resultassem em decisões de alocação de recursos. Embora sejam melhores que um previsor ingênuo (por exemplo, alguém que joga uma moeda para cima e se o resultado for cara, afirma que a ação X vai subir hoje, e se coroa, que ela vai cair), ainda assim não se pode afirmar que temos aqui uma metodologia brilhante de previsão de mercado.

Quanto ao número de neurônios na camada escondida, os resultados são bastante dispersos novamente (variando entre cinco e quinze), contudo há uma tendência de concentração entre seis e sete neurônios na camada escondida. Uma média simples resultaria em 8 neurônios na camada escondida, mas nesse caso este número não faz sentido, deve se olhar a concentração dos resultados. O histograma da Figura 11 explicita esses resultados, e a Tabela 6 traz os valores correspondentes.

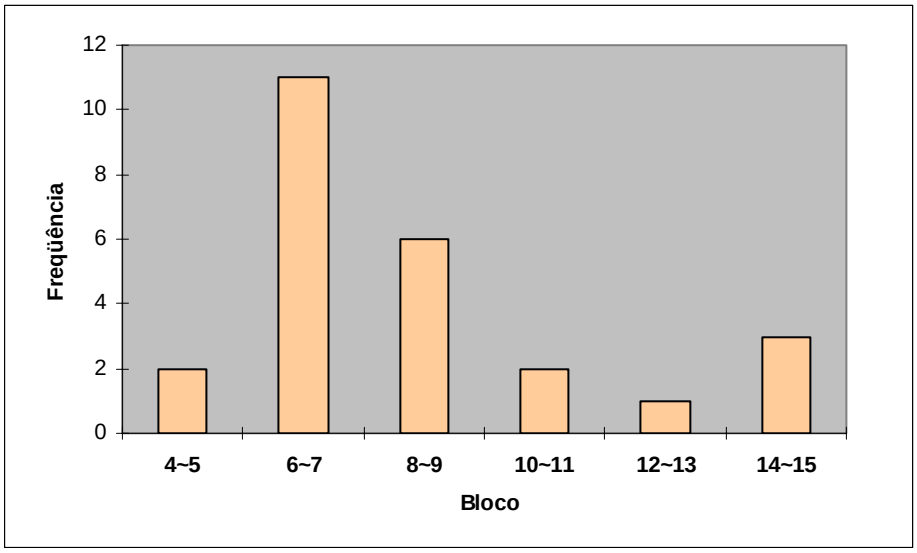


Figura 11: Histograma de Distribuição dos Neurônios na Camada Escondida para o algoritmo *Backpropagation* com *Momentum*

Código	Backpropagation com Momentum
ARCZ6	15
BBAS3	8
BBDC4	7
BELG4	6
CMIG4	6
CPLE6	15
CRUZ3	9
CSNA3	12
CSTB4	8

Código	Backpropagation com Momentum
EBTP4	14
ELET6	8
EMBR4	5
FFTL4	7
ITAU4	6
ITSA4	6
LAME4	6
PCAR4	6
PETR3	10

Código	Backpropagation com Momentum
PETR4	7
SBSP3	6
TLPP4	11
TNLP4	5
TSPP4	7
USIM5	9
VALE5	8

Tabela 6: Nº de Neurônios na Camada Escondida para *Backpropagation*

Um resultado típico de previsão realizado pelo algoritmo de *Backpropagation* pode ser visto na Figura 12 a seguir. Aqui, tem-se a ação Usiminas PNA (USIM5), com previsão realizada para o período entre 22/05/2003 e 10/10/2003. A rede foi treinada seguindo o paradigma definido anteriormente, na seção 6.3.3, e foram utilizados os dados para treinamento padronizado. Dado o fator aleatório na inicialização dos pesos da rede neural, o resultado mostrado a seguir tende a não ser semelhante à performance média exibida na Tabela 5 – ali se tem a agregação de muitos processos de treinamento consecutivos, e a seguir se tem apenas uma seção típica, que demonstra as qualidades do algoritmo.

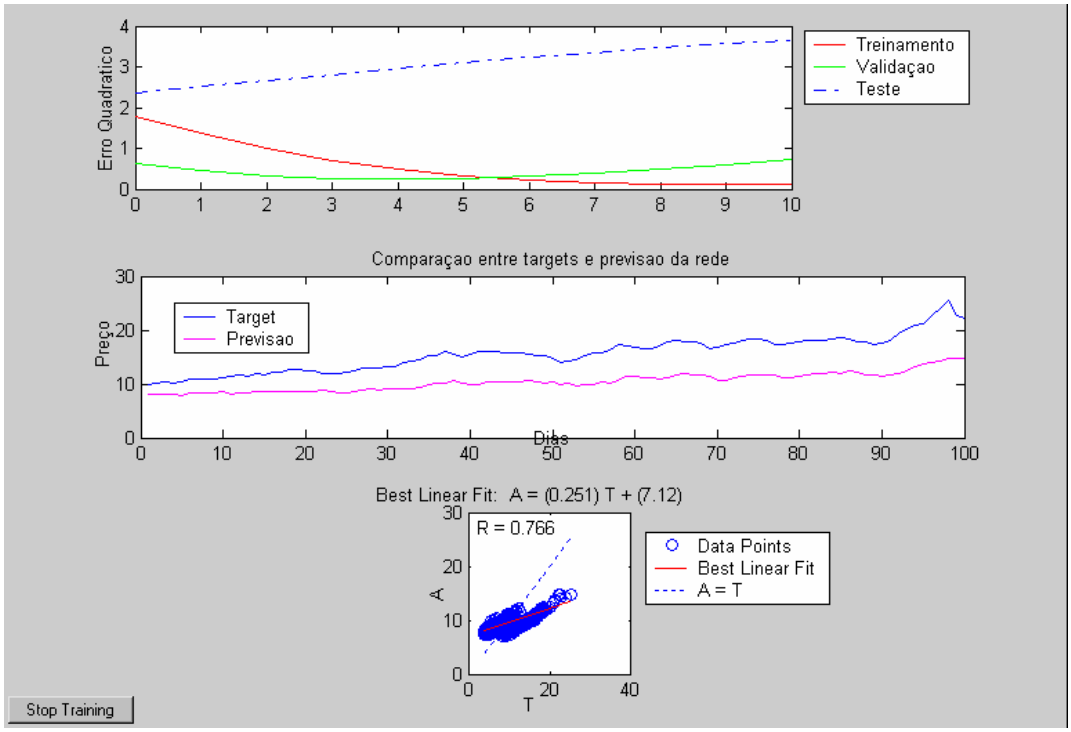


Figura 12: Resultado de Previsão com algoritmo *Backpropagation* com *Momentum*

É possível observar no gráfico do meio, uma comparação entre as previsões realizadas pela rede (série em magenta) e os preços reais observados no mercado. Fica claro que a rede neural gerada por este algoritmo de treinamento não obtém boa performance preditiva, o que é confirmado pelo valor do *R* calculado (0,766) e pelo primeiro gráfico da figura. Ali, aparece a evolução dos três erros calculados pela rede neural: o erro de treinamento (primeiros 900 dados, usados para treinar a rede), o erro de validação (os 100 dados subseqüentes) e o erro de teste (os 100 restantes, que vão de 22/05/03 até 10/10/03, conforme já mencionado). Fica claro que a rede neural gerada não é capaz de generalizar adequadamente, porque enquanto os erros de treinamento e validação diminuem, o erro calculado sobre o conjunto teste não só não diminui, como cresce ao longo do treinamento da rede, mostrando que o algoritmo não gera redes com boa capacidade de generalização.

6.3.3.2. Rede *Backpropagation* Resiliente

Redes neurais com múltiplas camadas tipicamente fazem uso de funções de transferência sigmóides (é o caso dos modelos aqui apresentados, que usam tangente hiperbólica). Estas funções são chamadas de *squashing* (fato já mencionado na seção

6.3.2) pois comprimem uma faixa de *inputs* infinitamente larga em uma faixa de *outputs* limitada (tipicamente entre 0 e 1 ou entre –1 e 1). Assim, estas funções são caracterizadas pelo fato de que sua derivada tende a zero conforme o valor da entrada cresce. Este fato causa um problema ao se usar descida pelo gradiente para realizar o treinamento, pois o valor do gradiente pode ter uma magnitude muito pequena, e portanto causar mudanças pequenas demais nos valores dos pesos e viés, mesmo que estes estejam longe de seus valores ótimos.

A idéia básica do algoritmo de *Backpropagation* resiliente está em eliminar estes efeitos nocivos dos valores das derivadas parciais (gradiente). Assim, apenas o sinal da derivada é utilizado para determinar a direção em que ocorrerá a mudança do peso da rede – a magnitude da derivada não é utilizada. O tamanho da variação do peso é determinada por um parâmetro distinto, controlado independentemente. Embora o uso de normalização no pré-processamento dos dados evite o problema de valores extremos nos *inputs*, ainda assim o algoritmo de retropropagação resiliente proporciona uma alternativa interessante aos algoritmos básicos, para previsão de preços em séries temporais de ativos financeiros.

Sendo assim, os resultados obtidos se mostraram consideravelmente melhores que os do algoritmo *Backpropagation* mais simples. De forma até inesperada, dado que a idéia básica por trás da retropropagação resiliente é um tanto simples, a performance do algoritmo se mostrou bastante boa. Os resultados podem ser observados na Tabela 7.

Código	Backpropagation Resiliente	Código	Backpropagation Resiliente	Código	Backpropagation Resiliente
ARCZ6	**99,70%**	EBTP4	98,36%	PETR4	98,42%
BBAS3	98,33%	ELET6	98,76%	SBSP3	**99,15%**
BBDC4	97,63%	EMBR4	98,89%	TLPP4	97,68%
BELG4	97,19%	FFTL4	89,28%	TNLP4	98,00%
CMIG4	97,87%	ITAU4	97,71%	TSPP4	**99,05%**
CPLE6	98,46%	ITSA4	98,37%	USIM5	91,28%
CRUZ3	96,46%	LAME4	96,68%	VALE5	98,73%
CSNA3	87,89%	PCAR4	98,34%		
CSTB4	96,45%	PETR3	98,74%		

Tabela 7: Resultados Algoritmo *Backpropagation* Resiliente

Quanto ao número de neurônios na camada escondida, para cada ação, e a distribuição de freqüências destas topologias, podem ser vistos na Tabela 8 e na Figura 13 a seguir:

Código	Backpropagation Resiliente	Código	Backpropagation Resiliente	Código	Backpropagation Resiliente
ARCZ6	11	EBTP4	9	PETR4	7
BBAS3	7	ELET6	5	SBSP3	12
BBDC4	9	EMBR4	7	TLPP4	15
BELG4	11	FFTL4	8	TNLP4	5
CMIG4	5	ITAU4	6	TSPP4	10
CPLE6	11	ITSA4	8	USIM5	9
CRUZ3	13	LAME4	11	VALE5	9
CSNA3	6	PCAR4	8		
CSTB4	5	PETR3	8		

Tabela 8: Nº de Neurônios na Camada Escondida para *Backpropagation* Resiliente

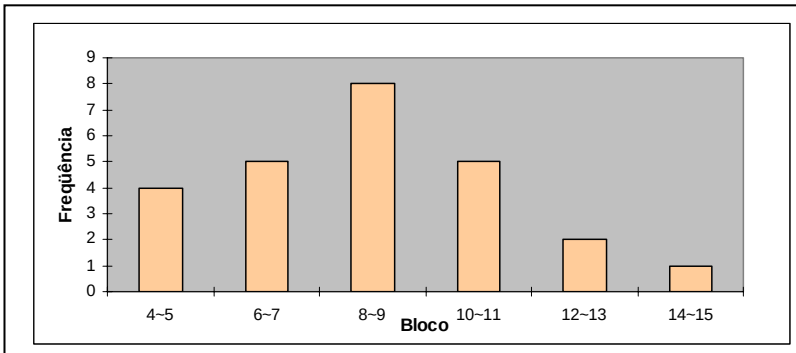


Figura 13: Histograma de Distribuição dos Neurônios na Camada Escondida para o algoritmo *Backpropagation* Resiliente

É possível constatar que há uma distribuição em torno do número 9 de neurônios na camada escondida, com preferência por redes menores. Contudo, não é possível concluir que este algoritmo tende a performar melhor com este algoritmo. De fato, as redes com melhor potencial preditivo, para as ações ARCZ6, SBSP3 e TSPP4, têm, respectivamente, 11, 12 e 10 neurônios em suas configurações de melhor desempenho. É interessante notar que, embora a performance do algoritmo tenha se mostrado boa em treinar redes para prever preços de ações, não há constância de performance: em algumas ações (as citadas acima), os resultados são bastante bons, enquanto em outras a performance é consideravelmente mais fraca (CSNA3 e FFTL4 são exemplos).

O que é possível concluir com clareza é que este algoritmo de *Backpropation* Resiliente proporciona performance consideravelmente melhor que a retropropagação simples, e que em alguns casos sua performance é tão boa ou até melhor que a de algoritmos mais complexos e (supostamente) mais avançados.

6.3.3.3. Rede Powell-Beale

O algoritmo de retropropagação básico ajusta os pesos, durante o treinamento, sempre na direção do gradiente de máxima declividade. Esta é a direção em que a função de erro (que mede a performance da rede) está decaindo mais rapidamente. Contudo, embora o erro diminua mais rapidamente na direção da negativa do gradiente, nem sempre é esta a direção que proporciona mais rápida convergência global. Esta é a idéia básica por trás dos algoritmos chamados de "Gradiente Conjugado": realizar uma busca numérica em direções conjugadas do gradiente, que geralmente produzem convergência mais rapidamente do que meramente seguir na direção de máxima declividade. Há várias maneiras de realizar esta busca pelas direções conjugadas, o que dá origem a uma família de diferentes algoritmos: Fletcher-Reeves, Polak-Ribiére, Powell-Beale e Gradiente Escalonado[14]. Para testar a capacidade dessa família de algoritmos em prever séries temporais de ativos financeiros, optou-se por usar a variante Powell-Beale do algoritmo.

Em todos os algoritmos desta família, periodicamente a direção de busca é reiniciada para o negativo do gradiente. Há várias maneiras de realizar essa reinicialização, uma das quais foi proposta por Powell em 1977, baseado num trabalho anterior de Beale, em 1972. Nesta técnica, a reinicialização ocorre quando há muito pouca ortogonalidade restante entre o gradiente na iteração *t* e o gradiente na iteração *t-1*. Há várias metodologias disponíveis para realizar a busca nas direções conjugadas (*Golden Section Seach, Brent's Search, Charambolous*, entre outras[15] - em todos os experimentos aqui realizados, será utilizada a busca pelo método de Charambolous).

Os resultados obtidos com este algoritmo na previsão de preços de ações se mostraram bastante satisfatórios. Na Tabela 9 está apresentada a performance do algoritmo para cada uma das 25 ações, em termos de coeficiente de correlação médio obtido em 50 repetições do processo completo de treinamento da rede:

[14] Para detalhes de cada um, vide Demuth e Beale (2001), capítulo 5.

[15] Vide Demuth e Beale (2001) para mais detalhes.

Código	Powell-Beale	Código	Powell-Beale	Código	Powell-Beale
ARCZ6	95,68%	EBTP4	**99,21%**	PETR4	98,86%
BBAS3	98,66%	ELET6	98,08%	SBSP3	98,82%
BBDC4	98,11%	EMBR4	**99,09%**	TLPP4	98,13%
BELG4	98,32%	FFTL4	96,44%	TNLP4	98,75%
CMIG4	97,58%	ITAU4	98,68%	TSPP4	98,68%
CPLE6	98,45%	ITSA4	98,81%	USIM5	94,88%
CRUZ3	97,27%	LAME4	97,69%	VALE5	97,71%
CSNA3	97,10%	PCAR4	98,60%		
CSTB4	97,30%	PETR3	98,33%		

Tabela 9: Resultados Algoritmo Powell-Beale

É possível constatar na Tabela 9 como a performance do algoritmo é bastante satisfatória, melhor que a apresentada pelos outros dois algoritmos já avaliados: há muito menor dispersão de performance entre as redes treinadas para cada uma das ações (a melhor é a de EBTP4, com 99,21% e a pior a de USIM5, com 94,88%). Assim, o aumento da sofisticação do algoritmo se traduz em ganho de performance em termos de melhores previsões realizadas.

Quanto à estrutura de neurônios escondidos, novamente há uma ampla distribuição, mas desta vez com alguma preferência por redes maiores, acima de 8 neurônios na camada escondida (a mais freqüente é a com 11 neurônios na camada escondida). Os resultados completos estão detalhados na Tabela 10 e na Figura 14 a seguir:

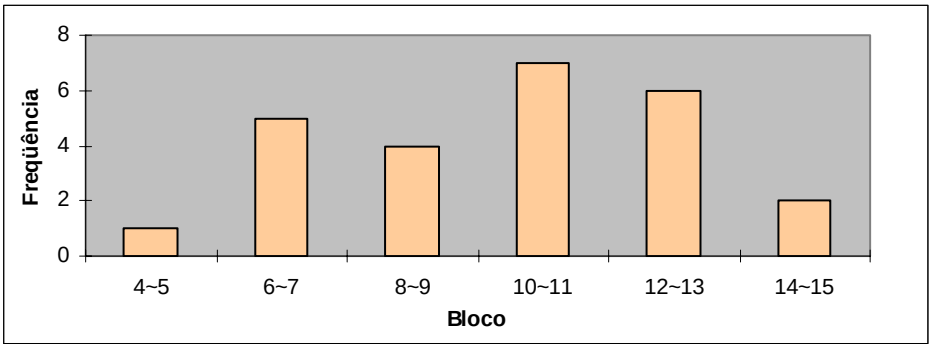


Figura 14: Histograma de Distribuição dos Neurônios na Camada Escondida para o algoritmo Powell-Beale

Código	Powell-Beale	Código	Powell-Beale	Código	Powell-Beale
ARCZ6	5	EBTP4	13	PETR4	10
BBAS3	6	ELET6	7	SBSP3	8
BBDC4	9	EMBR4	14	TLPP4	11
BELG4	13	FFTL4	11	TNLP4	11
CMIG4	7	ITAU4	13	TSPP4	12
CPLE6	7	ITSA4	11	USIM5	12
CRUZ3	10	LAME4	14	VALE5	8
CSNA3	13	PCAR4	10		
CSTB4	9	PETR3	7		

Tabela 10: Nº de Neurônios na Camada Escondida para Powell-Beale

6.3.3.4. Rede Quasi-Newton BFGS

Uma alternativa aos métodos de gradiente conjugado (como o Powell-Beale), para otimização rápida, é o método de Newton. O passo básico do método de Newton envolve a computação do Hessiano (derivadas segundas) da função de erro para os valores, na iteração *t*, dos pesos e viés da rede neural. Infelizmente, o cálculo do Hessiano da função é computacionalmente complexo (Demuth e Beale, 2000), e disso derivou uma família de algoritmos baseados no método de Newton, que não requerem o cálculo das derivadas segundas da função. Esta família é chamada de "Quasi-Newton". Dentre os métodos desta família, o mais bem sucedido é o *BFGS* (sigla para Broyden, Fletcher, Goldfarb e Shanno), que computa a cada iteração uma aproximação do Hessiano, baseado no valor do gradiente.

O algoritmo BFGS, embora possua os mesmos princípios básicos dos algoritmos Powell-Beale e Levenberg-Marquardt, surpreendentemente apresentou uma performance bastante ruim, bastante abaixo até do algoritmo *Backpropagation* com *Momentum*, que era, numa análise *a priori*, aquele do qual se esperava menos. A Tabela 11 a seguir deixa claro os resultados obtidos:

Código	BFGS	Código	BFGS	Código	BFGS
ARCZ6	97,78%	EBTP4	50,88%	PETR4	91,22%
BBAS3	84,55%	ELET6	87,14%	SBSP3	80,96%
BBDC4	90,20%	EMBR4	94,09%	TLPP4	85,53%
BELG4	62,56%	FFTL4	81,07%	TNLP4	84,19%
CMIG4	94,25%	ITAU4	88,35%	TSPP4	51,60%
CPLE6	90,33%	ITSA4	81,06%	USIM5	95,52%
CRUZ3	91,29%	LAME4	91,27%	VALE5	93,38%
CSNA3	97,62%	PCAR4	94,80%		
CSTB4	80,23%	PETR3	87,26%		

Tabela 11: Resultados Algoritmo BFGS

Além da performance ruim, a consistência também deixa a desejar: a rede treinada com BFGS de melhor performance – aquela para previsão de ARCZ6 – alcança 97,78% de correlação, enquanto a pior – TSPP4 – chega a apenas 51,60%, um desempenho muito fraco de previsão. Além disso, a topologia também varia consideravelmente, com agrupamentos em torno de 9 e 13 neurônios na camada escondida. Os resultados estão na Tabela 12 e Figura 15.

Código	BFGS
ARCZ6	15
BBAS3	9
BBDC4	7
BELG4	15
CMIG4	9
CPLE6	5
CRUZ3	13
CSNA3	12
CSTB4	8

Código	BFGS
EBTP4	11
ELET6	13
EMBR4	9
FFTL4	6
ITAU4	13
ITSA4	10
LAME4	13
PCAR4	10
PETR3	9

Código	BFGS
PETR4	13
SBSP3	14
TLPP4	14
TNLP4	8
TSPP4	5
USIM5	11
VALE5	9

Tabela 12: Nº de Neurônios na Camada Escondida para BFGS

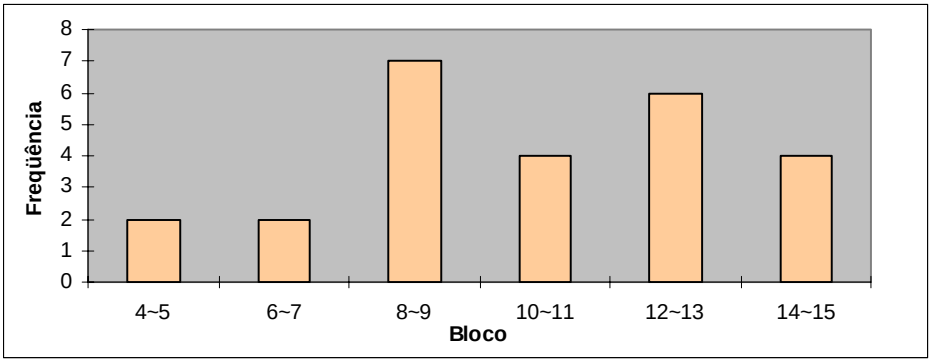


Figura 15: Histograma de Distribuição dos Neurônios na Camada Escondida para o algoritmo BFGS

6.3.3.5. Rede Levenberg-Marquardt

Assim como as técnicas quasi-Newton, o algoritmo de Levenberg-Marquardt foi desenhado para realizar uma aproximação do Hessiano da função de erro de treinamento da rede neural a cada iteração. Quando a função erro tem a forma de uma soma de quadrados (como é tipicamente o caso com redes neurais – e especificamente o caso deste trabalho, onde a função de erro padrão é o erro quadrático médio), é possível aproximar o Hessiano através da seguinte equação:

$$H = J^T J \tag{6.1}$$

e o gradiente é computado pela seguinte relação:

$$g = J^T e \tag{6.2}$$

onde J é o Jacobiano da função erro, que depende apenas das derivadas de primeira ordem da função erro com respeito aos pesos e viés da rede, e e é um vetor dos erros. O processo de computação do Jacobiano é bastante mais simples que o processo de computação do Hessiano (Demuth e Beale, 2000), o que faz do algoritmo Levenberg-Marquardt um poderoso algoritmo de treinamento para redes neurais.

Dentre todos os algoritmos de treinamento testados ao longo deste trabalho, o Levenberg-Marquardt foi o que apresentou melhor performance. Tanto em termos de velocidade quanto em termos de qualidade da regressão. A tabela a seguir mostra os resultados obtidos. Note-se que a performance sempre se situa acima da faixa de 97%, mostrando tanto consistência entre todas as 25 ações – ao contrário dos algoritmos anteriores, que tinham performance muito boa em uma ação e bastante fraca em outras – quanto constância, ou seja, mesmo reinicializando a rede aleatoriamente várias vezes, o resultado era bastante constante a cada iteração.

Código	Levenberg-Marquardt
ARCZ6	**99,10%**
BBAS3	**99,40%**
BBDC4	98,65%
BELG4	**99,35%**
CMIG4	98,45%
CPLE6	**99,35%**
CRUZ3	98,58%
CSNA3	97,67%
CSTB4	98,63%

Código	Levenberg-Marquardt
EBTP4	**99,83%**
ELET6	**99,17%**
EMBR4	**99,66%**
FFTL4	97,89%
ITAU4	**99,17%**
ITSA4	**99,54%**
LAME4	97,98%
PCAR4	**99,28%**
PETR3	**99,57%**

Código	Levenberg-Marquardt
PETR4	**99,29%**
SBSP3	**99,57%**
TLPP4	**99,21%**
TNLP4	**99,17%**
TSPP4	**99,77%**
USIM5	96,81%
VALE5	**99,12%**

Tabela 13: Resultados Algoritmo Levenberg-Marquardt

Além da performance elevada e consistente, o algoritmo apresenta uma clara preferência por uma topologia de rede específica, com 4 ou 5 neurônios na camada escondida. O histograma na Figura 16 mostra este fato claramente, corroborado pela Tabela 14, com os dados para cada uma das 25 ações. Curiosamente, tal comportamento não foi apresentado por nenhum outro dos algoritmos nem foi constatado em nenhuma das 25 ações particularmente. Aparentemente, a capacidade do algoritmo Levenberg-Marquardt de treinar rapidamente a rede permite a ele trabalhar com menos pesos e *bias*, mantendo o erro de validação baixo, dado que é capaz de evitar o problema de *excesso de ajuste* ou *excesso de treinamento.*

Código	Levenberg-Marquardt	Código	Levenberg-Marquardt	Código	Levenberg-Marquardt
ARCZ6	14	EBTP4	7	PETR4	4
BBAS3	4	ELET6	4	SBSP3	4
BBDC4	6	EMBR4	8	TLPP4	5
BELG4	5	FFTL4	5	TNLP4	4
CMIG4	4	ITAU4	6	TSPP4	11
CPLE6	7	ITSA4	4	USIM5	4
CRUZ3	9	LAME4	6	VALE5	6
CSNA3	15	PCAR4	5		
CSTB4	6	PETR3	8		

Tabela 14: Nº de Neurônios na Camada Escondida para Levenberg-Marquardt

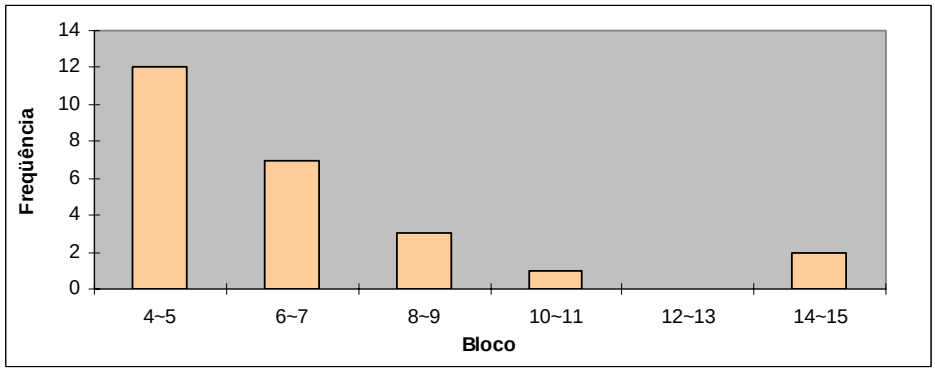


Figura 16: Histograma de Distribuição dos Neurônios na Camada Escondida para o algoritmo Levenberg-Marquardt

Dada a qualidade do potencial preditivo das redes neurais treinadas com o algoritmo Levenberg-Marquardt, estas se tornam a opção óbvia para a realização das previsões que serão utilizadas pelo modelo de algoritmos genéticos para otimização das carteiras. Além disso, dada a preferência pela topologia com poucos neurônios na camada escondida, optou-se por uma rede com 5 neurônios na camada intermediária como a *proxy* do modelo de previsão de retornos por redes neurais artificiais.

6.3.4. Resultados

Após testar cinco tipos de algoritmos de treinamento de redes neurais diferentes, onze topologias possíveis em cada algoritmo, e rodar os modelos centenas de vezes, é possível concluir, com elevado grau de confiança, que o algoritmo Levenberg-Marquardt é o melhor algoritmo de treinamento (dentre os pesquisados) para redes neurais cujo intuito é realizar previsões de retornos futuros de séries temporais de ativos financeiros. Mais ainda, redes Levenberg-Marquardt apresentam melhor

performance com um número reduzido de neurônios na camada escondida, o que é de certa maneira contra-intuitivo, visto que há uma tendência a se relacionar capacidade de absorver conhecimento (em outras palavras, redes topologicamente maiores) com capacidade de previsão. Os resultados obtidos vão contra esta crença. Além disso, dados os elevados graus de correlação obtidos entre as previsões e os dados historicamente observados, também é possível inferir que redes neurais artificiais são plenamente capazes de prever preços futuros de ações. Até onde é possível levar esta previsão, contudo, é uma discussão diferente, e no capítulo 7 esta questão será explorada.

De modo a consolidar e explicitar as conclusões dos resultados obtidos, convém analisar brevemente as Tabelas 15 e 16. A Tabela 15 traz uma consolidação de todas as topologias definidas como tendo melhor performance para cada uma das ações e cada um dos algoritmos. A Tabela 16, a mais importante, traz os resultados de coeficiente de correlação para cada uma das 25 ações com previsões a partir do treinamento com cada um dos 5 algoritmos. Na Tabela 16, para cada ação, está ressaltado o algoritmo que obteve melhor performance. A superioridade do Levenberg-Marquardt fica evidente:

Número de Neurônios na Camada Escondida com Melhor Performance

	Algoritmo				
Código	Backpropagation com Momentum	Backpropagation Resiliente	Powell-Beale	BFGS	Levenberg-Marquardt
ARCZ6	15	11	5	15	14
BBAS3	8	7	6	9	4
BBDC4	7	9	9	7	6
BELG4	6	11	13	15	5
CMIG4	6	5	7	9	4
CPLE6	15	11	7	5	7
CRUZ3	9	13	10	13	9
CSNA3	12	6	13	12	15
CSTB4	8	5	9	8	6
EBTP4	14	9	13	11	7
ELET6	8	5	7	13	4
EMBR4	5	7	14	9	8
FFTL4	7	8	11	6	5
ITAU4	6	6	13	13	6
ITSA4	6	8	11	10	4
LAME4	6	11	14	13	6
PCAR4	6	8	10	10	5
PETR3	10	8	7	9	8
PETR4	7	7	10	13	4
SBSP3	6	12	8	14	4
TLPP4	11	15	11	14	5
TNLP4	5	5	11	8	4
TSPP4	7	10	12	5	11
USIM5	9	9	12	11	4
VALE5	8	9	8	9	6

Tabela 15: Topologias de Melhor Performance

	Melhor Resultado de Regressão (R)				
	Algoritmo				
Código	Backpropagation com Momentum	Backpropagation Resiliente	Powell-Beale	BFGS	Levenberg-Marquardt
ARCZ6	83,25%	98,72%	99,30%	97,78%	99,10%
BBAS3	81,27%	98,33%	98,66%	84,55%	99,40%
BBDC4	80,73%	97,63%	98,11%	90,20%	98,65%
BELG4	58,04%	97,19%	98,32%	62,56%	99,35%
CMIG4	88,77%	97,87%	97,58%	94,25%	98,45%
CPLE6	79,40%	98,46%	98,45%	90,33%	99,35%
CRUZ3	68,24%	96,46%	97,27%	91,29%	98,58%
CSNA3	42,98%	87,89%	97,10%	97,62%	97,67%
CSTB4	43,13%	96,45%	97,30%	80,23%	98,63%
EBTP4	78,11%	98,36%	99,21%	50,88%	99,83%
ELET6	89,86%	98,76%	98,08%	87,14%	99,17%
EMBR4	81,97%	98,89%	99,09%	94,09%	99,66%
FFTL4	43,34%	89,28%	96,44%	81,07%	97,89%
ITAU4	79,78%	97,71%	98,68%	88,35%	99,17%
ITSA4	81,36%	98,37%	98,81%	81,06%	99,54%
LAME4	51,53%	96,68%	97,69%	91,27%	97,98%
PCAR4	84,11%	98,34%	98,60%	94,80%	99,28%
PETR3	83,75%	98,74%	98,33%	87,26%	99,57%
PETR4	82,24%	98,42%	98,86%	91,22%	99,29%
SBSP3	81,20%	99,15%	98,82%	80,96%	99,57%
TLPP4	69,78%	97,68%	98,13%	85,53%	99,21%
TNLP4	78,31%	98,00%	98,75%	84,19%	99,17%
TSPP4	81,57%	99,05%	98,68%	51,60%	99,77%
USIM5	91,30%	91,28%	94,88%	95,52%	96,81%
VALE5	87,46%	98,74%	98,95%	93,38%	99,12%
média	74,86%	97,06%	98,16%	85,09%	98,97%

Tabela 16: Resumo da Performance Preditiva dos Algoritmos

6.4. Modelo de Previsão de Risco *GARCH*

A segunda parte fundamental do modelo de gestão de carteiras de ações trata da previsão dos riscos. O papel do risco no contexto da teoria de portfólios é absolutamente fundamental, sendo a grande inovação do trabalho de Markowitz a conceituação do risco como a variância dos retornos – até então risco não era quantificado explicitamente. Assim, dado que aqui se busca desenvolver um modelo consistente para gerir carteiras de investimento em ações, uma parte essencial do trabalho tem de ser devotada à busca de previsões consistentes dos riscos futuros.

Conforme mencionado no item 2.6, as metodologias modernas de previsão de risco fazem uso de ferramentas estatísticas para dar um significado quantitativo ao fato de que, quando se trata de risco, o passado recente tem uma importância significativamente maior que eventos transcorridos num passado mais remoto. Um dos modelos mais utilizados para o estudo e a previsão de séries temporais de ativos

financeiros é o modelo GARCH, ou heterocedástico autoregressivo condicional, desenvolvido por Engle e Bollerslev[16]. Em uma definição pouco rigorosa, é possível dizer que heterocedástico se refere à variância que muda no tempo (ou seja, variância não estática ou constante). O termo condicional se refere ao fato de haver uma dependência das informações do passado imediato, e o autoregressivo descreve um mecanismo de *feedback* em que as observações passadas são incorporadas ao presente.

O uso de modelagem GARCH para previsão de risco (ou volatilidade) em séries temporais financeiras é altamente difundido, tanto na academia quanto no mercado, e traz algumas vantagens importantes: leva em conta excesso de curtose (responsável pelas "caudas largas" observadas em séries temporais financeiras) e o fenômeno de *clustering* de volatilidade (ou seja, observações de alta volatilidade tendem a gerar alta volatilidade, e vice-versa – a volatilidade se agrega, de um certo modo). Contudo, são modelos paramétricos, e por isso tendem a operar melhor quando não há flutuações de grande magnitude na série (por exemplo, quando o mercado cai 20% num dia, como ocorreu pós-atentado de 11 de Setembro).

Dentro da classe de modelos GARCH, o mais simples deles é o chamado GARCH (1,1). Esse modelo tem subjacente uma representação da média e da variância condicional da série temporal, que seguem as seguintes equações:

$$y_t = C + \varepsilon_t \tag{6.3}$$

$$\sigma_t^2 = \kappa + G_1\sigma_{t-1}^2 + A_1\varepsilon_{t-1}^2 \tag{6.4}$$

onde o retorno y_t consiste de uma constante, mais um ruído aleatório, e a variância consiste de três termos: uma constante, uma ponderação da variância anterior, e o quadrado do ruído observado no período anterior. Embora simples, este modelo requer a estimação de apenas 4 parâmetros (C, κ, G_1, A_1), e consegue capturar a maior parte da variabilidade de séries temporais de retornos de ações (Demuth e Beale, 2001).

Assim, será usado um modelo GARCH (1,1) para a realização de previsões de risco futuro para as 25 ações escolhidas. Dado o caráter autoregressivo do modelo, a

[16] Em setembro de 2003, o americano Robert Engle recebeu o Prêmio Nobel de Economia pelo desenvolvimento dessa classe de modelos de previsão de risco. No anúncio do prêmio, o comitê do Nobel ressalta a importância da previsão de riscos em séries temporais de ativos financeiros. Para detalhes, vide http://www.nobel.se.

previsão de risco depende apenas dos preços passados das ações, não sendo necessários outros *inputs*. Além disso, a previsão será realizada em caráter dinâmico com janela deslizante: define-se o tamanho da janela (por exemplo, 10 dias), os preços da ação X durante este período são coletados, os parâmetros do modelo GARCH (1,1) são estimados, via otimização, e então a previsão do risco para o dia seguinte é realizada. Feito isto, a janela desliza um período, saindo de *t* e indo para *t+1*, e todo o processo é reiniciado.

No caso da aplicação deste trabalho, a janela foi definida como sendo de 21 dias úteis. Tal escolha deve-se aos seguintes fatos: (i) um mês (21 dias úteis em média) é período suficiente para a captura da dinâmica do risco dos retornos de qualquer ação; (ii) é possível estimar os parâmetros com algum grau de confiança (21 dias úteis geram 20 observações de retornos, ou seja, há 20 dados para estimar-se os quatro parâmetros do modelo); e finalmente (iii) o esforço computacional é reduzido, se comparado àquele com janelas mais curtas. Ainda assim, o esforço computacional é relevante: para 25 ações, com 1100 observações cada uma, e janela deslizante de 21 dias, tem-se o seguinte:

25 ações x (1100-21) observações = 26.975 modelos = 107.900 parâmetros[17]

Assim, o modelo desenvolvido para a previsão de risco das séries temporais dos ativos financeiros é composto pelas etapas da Figura 17.

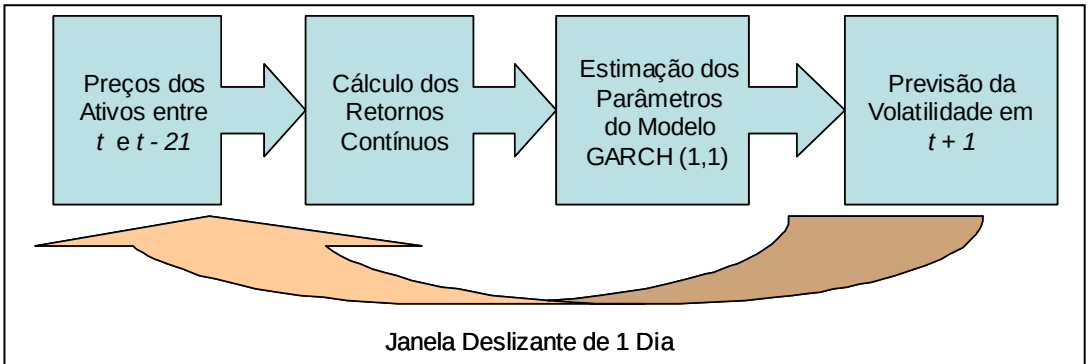


Figura 17: Passos do Modelo GARCH de Previsão de Risco

O modelo foi aplicado às séries das 25 ações, gerando previsões de risco para 1079 períodos, entre as datas de 09/06/1999 e 10/10/2003. Por exemplo, para a ação CSN ON (CSNA3), as previsões de risco têm a evolução vista na Figura 18.

[17] Rodando em um Pentium III de 800 MHz e 256 MB de RAM, o *script* de *MATLAB* escrito para realizar a previsão de riscos levou 8 horas ininterruptas de cálculo para realizar todas as previsões. Detalhes deste código podem ser encontrados no Apêndice 1.

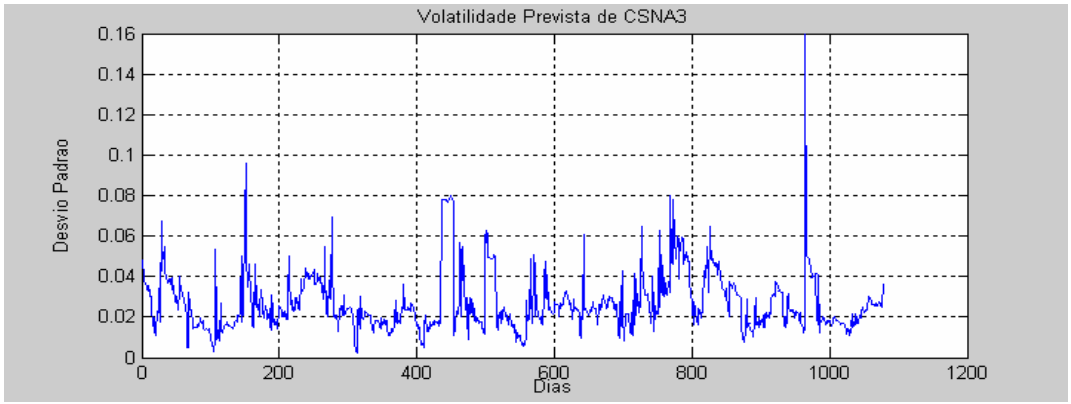


Figura 18: Volatilidade Prevista de CSNA3

Na Figura 18 já é possível observar que, ocasionalmente, ocorrem saltos no risco previsto pelo modelo, embora, no geral, as previsões de risco tendam a se concentrar em faixas razoavelmente estreitas (no caso de CSNA3, por exemplo, entre 0 e 4%).

Para constatar melhor estes fatos, convém observar mais séries temporais de previsões de risco resultantes do modelo. Assim, tem-se para Petrobrás PN (PETR4) e Usiminas PNA (USIM5), as previsões de risco vistas nas Figuras 19 e 20.

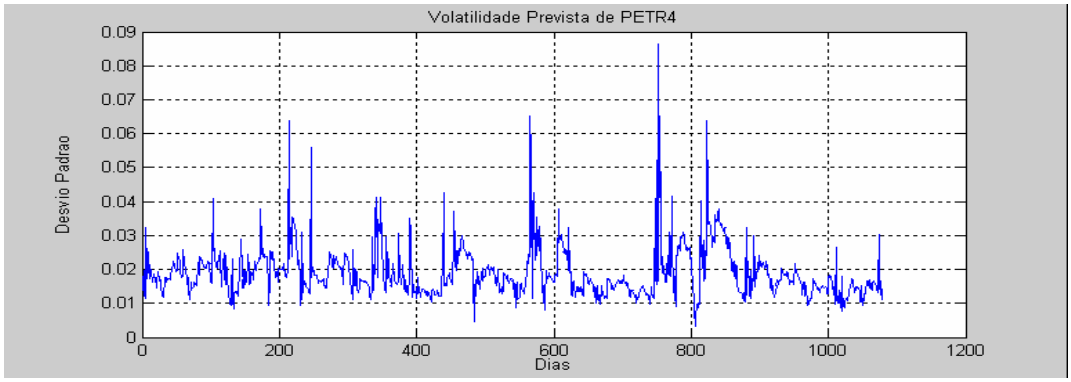


Figura 19: Volatilidade Prevista de PETR4

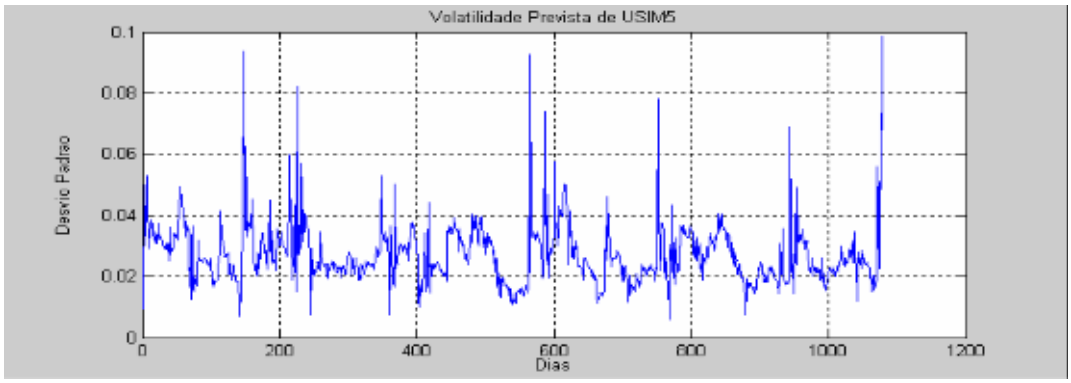


Figura 20: Volatilidade Prevista de USIM5

No Apêndice 2, são apresentados os resultados completos (curvas de previsão de risco histórica) para todas as 25 ações.

É interessante notar que a previsão de risco serve para fundamentar o risco futuro que se está correndo com determinada ação. Ou seja, ela dá uma idéia da variabilidade futura dos retornos desta ação. Assim, dada uma previsão de retorno (como as realizadas ao longo da seção 6.3 anterior), a previsão de risco permite ter uma idéia do grau de confiança que se pode ter em relação a esta previsão de retorno. Assim, quando se menciona que a previsão de risco para o dia 10/10/03 da ação Aracruz ON (ARCZ6) é 0,014619, ou 1,4619%, esta informação apenas faz sentido quando analisada em conjunto com a previsão de retornos.

6.5. Modelo de Gestão de Carteiras com Algoritmos Genéticos

Após analisar detidamente tanto os modelos de previsão de retornos usando redes neurais, quanto o modelo GARCH para previsão de riscos, dentro da proposta deste trabalho, falta agora agregar os resultados obtidos anteriormente, e colocá-los para otimizar uma carteira de investimentos em ações. Dessa maneira, será possível aferir a verdadeira performance dos modelos, e testar o uso de algoritmos genéticos como uma ferramenta heurística para otimização das carteiras.

6.5.1. *Inputs* do Modelo

Dentro do paradigma teórico delineado por Markowitz e Sharpe, descrito em detalhes no capítulo 2 (notadamente na seção 2.5), são necessárias duas informações para a construção de uma carteira de investimentos que seja ótima: os retornos futuros dos ativos, e os riscos. Dadas as duas, é possível construir a chamada fronteira eficiente (vide Figuras 1 e 2), e obter a carteira ótima para aquele investidor.

Assim, ao construir e testar os modelos de previsão de retornos e riscos, o que se buscou fundamentalmente foi obter, com alguma confiança, estas informações que a Moderna Teoria de Portfólios preconiza serem necessárias para a construção da carteira eficiente.

Conforme descrito na seção 6.3.3, o período utilizado para as previsões pelos modelos de rede neural foi escolhido entre 22/05/2003 e 10/10/2003, o que resulta em 100 previsões de preços. Conforme descrito na seção 6.3.4, estas previsões foram calculadas utilizando-se o algoritmo Levenberg-Marquardt com 5 neurônios na camada escondida, para todas as 25 ações pré-selecionadas. Dados os 100 preços previstos, são então calculados 99 retornos previstos, da seguinte maneira:

$$R_{previsto} = \ln\left(\frac{\Pr evisão_t}{\Pr eço \quad Médio_{t-1}}\right) - 1 \tag{6.5}$$

visto que a janela de previsão de preços utilizada foi justamente essa, a dos preços do dia seguinte. Ou seja, dado o preço observado em mercado em *t*, e os dados necessários para realizar a previsão (dólar, Ibovespa, etc., conforme seção 6.2), prevê-se o preço para *t+1*, e então se calcula o retorno esperado entre *t* e *t+1*.

Assim, tem-se uma série de 99 retornos esperados para cada uma das 25 ações, e a estes dados são agregados os riscos previstos pelo modelo GARCH, para o mesmo período (na seção 6.4, foram previstos os riscos das ações para o período completo entre 10/05/1999 e 10/10/2003, aqui se vai utilizar apenas uma pequena porção destes dados).

Além disso, conforme as equações 2.11 e 2.12, a correlação (ou a covariância) entre os ativos é um componente fundamental da determinação do risco agregado de um portfólio. Embora já tenham sido realizadas previsões dos riscos de cada ação, estas dão medida do risco individual, e não do risco sistemático da carteira. Assim, é necessário também determinar o relacionamento entre os riscos de cada ação. Para tanto, vai-se calcular a correlação entre cada uma das 25 ações, com uma janela de 21 dias (mesmo período usado para determinação do risco) – contudo, tal janela não será deslizante, se está considerando, simplificadamente, que a correlação futura dos ativos vai seguir a correlação do passado recente. Dado que as carteiras a serem otimizadas serão em um curto período (100 dias) é possível considerar que a correlação não varia tanto de um período para outro. Em suma, por exemplo, a correlação entre ARCZ6 e VALE5 utilizada para a otimização da carteira do vai ser calculada com base nos preços dos ativos entre 22/05/2003 e 23/04/2003, ou seja, a janela de 21 dias antes do início dos experimentos.

Portanto, serão dados como *inputs* do modelo de algoritmos genéticos duas séries temporais (previsão de risco e previsão de retorno) e uma matriz de correlações 25x25, indicando as correlações entre cada uma das ações da carteira.

6.5.2. Parâmetros do Algoritmo Genético

Definidos os dados que serão introduzidos no modelo, resta escolher os parâmetros dos algoritmos, de maneira a poder implementar o modelo e analisar a performance global das carteiras.

O parâmetro mais importante a ser definido é a função objetivo – conforme visto na seção 4.5.2, esta definição afeta decisivamente a performance do algoritmo. Além disso, é o parâmetro que vai definir a performance das carteiras, em última instância. De um modo geral, dentro da moderna teoria de portfólios, costuma-se otimizar um dos parâmetros, ou risco ou retorno, para um nível pré-fixado do outro. Em outras palavras, busca-se a carteira de melhor retorno para um dado risco, ou busca-se a carteira de menor risco para um dado retorno desejado. Contudo, tal escolha é extremamente sensível ao nível exato que se pré-define. Por exemplo, tem-se dez ativos, e deseja-se a carteira de menor risco que dê um retorno de 4,5%. O resultado da otimização (definição dos pesos de cada um dos dez ativos da carteira) pode ser extremamente sensível ao nível definido de retorno desejado, ou seja, se fosse escolhido um retorno esperado de 4,6%, os pesos dos ativos poderiam ser radicalmente diferentes. Em suma: a otimização comumente realizada se pré-fixando um dos parâmetros é muito pouco robusta. E como se está tratando de buscar uma boa performance futura, é desejável ter alguma robustez nos resultados obtidos.

Sendo assim, a opção deste trabalho foi por trabalhar com uma função objetivo que unisse os dois critérios fundamentais do paradigma teórico de Markowitz, risco e retorno. Assim, vai-se buscar otimizar, via algoritmos genéticos, a seguinte função, que vai se chamar aqui de “Índice de Sharpe modificado”:

$$IS_{\mathrm{mod}\,ificado} = \frac{r_{esperado}}{\sigma_{esperado}} \tag{6.6}$$

Optou-se por não considerar a taxa livre de risco como um parâmetro do índice (compare-se a equação 6.4 com a equação 2.30) dado o foco em testar a qualidade

dos modelos em gerar retornos absolutos, e não em gerar retornos "em excesso" (ou seja, retornos acima da taxa livre de risco).

Assim, o índice definido na equação 6.6 será a função objetivo (ou de utilidade, na terminologia utilizada no capítulo 4) do algoritmo genético. Todas as soluções candidatas serão avaliadas em termos deste índice. O $r_{esperado}$ será calculado conforme a equação 6.7 a seguir (repetindo a equação 2.7):

$$r_{esperado} = w \cdot r^T \tag{6.7}$$

onde *w* é o vetor de pesos de cada ativo na carteira, e *r* é o vetor de retornos previstos usando o modelo de redes neurais. Já o $\sigma_{esperado}$ será calculado com base na equação 6.8 a seguir (semelhante às equações 2.11 e 2.12):

$$\sigma_{esperado} = \sqrt{w \cdot Co\,\mathrm{var} \cdot w^T} \tag{6.8}$$

Além da função objetivo, devem ser definidas também as restrições que serão aplicadas ao modelo de otimização. Primeiramente, optou-se por não se permitir alavancagem da carteira, ou seja, nenhum ativo pode ter mais de 100% da carteira, nem serão permitidas vendas a descoberto de quaisquer ações[18]. Esta restrição é implementada no modelo através das seguintes equações:

$$\begin{aligned} &\sum_i w_i = 1 \\ &0 \le w_i \le 1 \end{aligned} \tag{6.9}$$

Outra restrição implementada no modelo de gestão diz respeito à quantidade de ações diferentes na carteira. Dentre as 25 ações para escolher para a carteira, há a preferência por ter uma quantidade média de papéis – visto que isto facilita a gestão cotidiana, e reduz a necessidade de controlar uma grande quantidade de diferentes ações. Assim, optou-se por impor uma restrição "leve": há uma preferência por soluções que tenham ao máximo 8 ações em carteira. O termo "leve" se refere à imposição de uma penalidade à função objetivo daquela determinada solução candidata que represente uma carteira com mais de 8 ações . Isto é diferente de impor uma restrição "dura", como a representada pela equação 6.9, que obrigatoriamente

[18] "Vender a descoberto" é um jargão do mercado de ações que significa apostar na queda de uma determinada ação, alugando-a de alguém que a possui (pagando por isso um pequeno juro) e vendendo a mesma no mercado. O lucro vem se o preço da ação cair, pois ficará mais barato para comprá-la no mercado para entregar para o dono original.

não deve ser quebrada. Nesta, de quantidade de papéis, a penalidade segue a seguinte função:

$$penalidade = 100 \cdot \left(e^{desvio/100} - 1 \right) \tag{6.10}$$

Portanto, definidas a função objetivo e as restrições sobre ela impostas, falta definir os parâmetros próprios do algoritmo genético. A Tabela 17 mostra as escolhas realizadas:

Parâmetro	Valor
Função Objetivo	*Índice de Sharpe modificado*
Restrições	*Pesos >= 0*
	Preferência por 8 ações ou menos
População	*1000 soluções candidatas*
Taxa de *Crossover*	*50%*
Taxa de Mutação	*5%*
Iterações	*100.000*

Tabela 17: Parâmetros do Algoritmo Genético

Optou-se por trabalhar com populações grandes (de 500 indivíduos), para conseguir varrer uma parte relevante do espaço de busca, que é bastante grande para 25 ações. A taxa de mutação foi definida em 5% para não introduzir excessiva aleatoriedade na varredura do espaço de busca, e a taxa de crossover em 50% para permitir que as boas soluções tenham alta chance de se propagarem pelas populações descendentes[19]. Todos os modelos serão rodados em 1000 iterações, e interrompidos se a variação entre a utilidade da melhor solução não variar em mais de 1% por 100 iterações consecutivas. Com estas definições, garante-se que o algoritmo vai varrer uma região considerável do espaço de busca. Definidos todos os parâmetros relevantes, é possível implementar o modelo e analisar as carteiras geradas e o resultado por elas alcançado.

6.5.3. Resultados

Os *inputs* do modelo de gestão de carteiras estão definidos para o período entre 23/05/2003 e 10/10/2003. Assim, para cada uma destas datas, será rodado um modelo de algoritmo genético, o que corresponde a gerar, para cada dia, uma carteira ótima, de maneira a poder analisar se as carteiras geradas pelo modelo trariam bons retornos caso fossem aplicadas na realidade. Uma hipótese que se faz daqui por

[19] Vide seções 4.5 e 4.6 para detalhamento de todos os termos.

diante é que todas as ações podem ser compradas em quaisquer quantidades, sem alterar o preço do mercado[20].

É interessante acompanhar como o algoritmo genético vai progressivamente alcançando soluções melhores, através dos operadores de *crossover* e mutação. Nas figuras a seguir, é possível enxergar como, progressivamente, as soluções candidatas com maior utilidade (neste caso, maior Índice de Sharpe *modificado*) vão dominando a população total, numa evidência clara de "darwinismo computacional". Na Figura 21, os organismos mal-adaptados ainda dominam a população (figura da direita, são o grosso, abaixo da linha preta), por terem aparecido antes (figura da esquerda, o ramo de baixo). Já na Figura 22, os organismos bem-adaptados dominam a população, e o ramo inferior passou a tender ao superior.

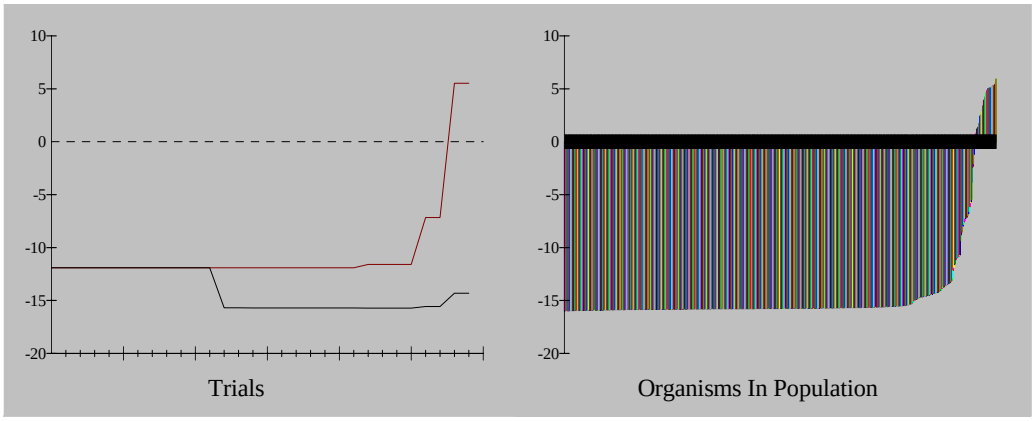


Figura 21: Evolução da Utilidade das Soluções (esq.) e Utilidade dos Organismos numa dada população dentro do Algoritmo Genético (dir.), ao início da otimização

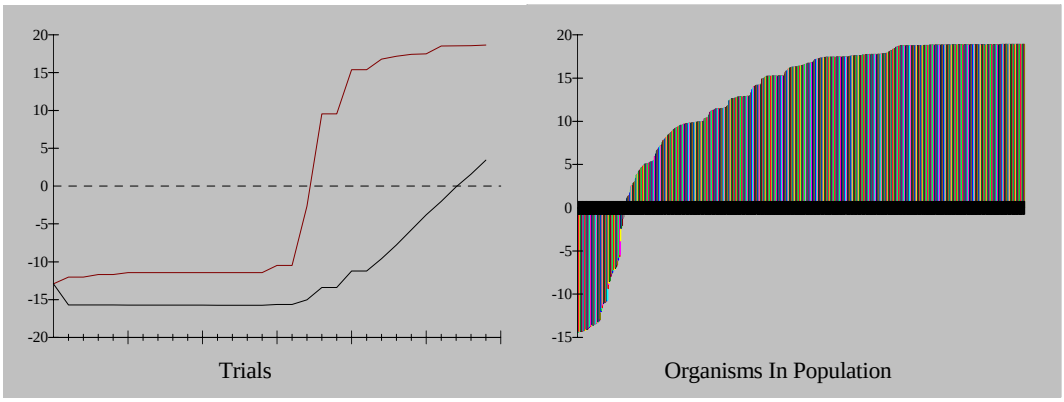


Figura 22: Evolução da Utilidade das Soluções (esq.) e Utilidade dos Organismos numa dada população dentro do Algoritmo Genético (dir.), próximo ao fim da otimização

[20] Para uma discussão de hipóteses deste tipo, e seu papel na teoria de finanças, vide Campbell et al. (1997), cap. 3, pp.83-144, que versa sobre um tópico conhecido como "Microestrutura de Mercado".

Algumas constatações foram feitas acerca do comportamento global do modelo de otimização de carteira usando AGs: (i) embora fosse definido *a priori* 100.000 iterações, para permitir uma boa varredura do espaço de busca, de um modo geral, para todas as carteiras calculadas, o algoritmo convergia para a região da solução em torno de 3.000 iterações – a partir de um certo momento, foi reduzido este parâmetro do modelo, de 100.000 para 10.000, sem perda perceptível de performance e com ganho considerável em termos de tempo de máquina, processamento e memória (em problemas maiores, a definição adequada deste parâmetro se torna fundamental portanto); (ii) de um modo geral, todas as implementações do AG começavam com Índice de Sharpe negativo, ou seja, buscando uma região inadequada do espaço de busca – mas a mutação proporcionava, em torno de 500 iterações, uma saída desta região (por isso a taxa de mutação aparentemente alta – 5% - quando comparado com as definições normalmente dadas em teoria – vide capítulo 4 – de taxas em torno de 1%); (iii) todos os modelos foram inicializados com os mesmos valores para os pesos dos ativos na carteira: 4% para cada uma das 25% ações, totalizando 100%. Desse modo, evita-se a introdução de algum tipo de viés na heurística modelada pelo algoritmo genético.

Para cada uma das 99 datas (de 23/05/2003 até 10/10/2003) foi otimizada uma carteira, tomando como entrada os retornos previstos para aquela data, os riscos idem, e a correlação com janela de 21 dias antes do início do período. Assim, a carteira que o algoritmo genético retornar para uma data *t* representa a carteira que o modelo recomenda que um investidor possua naquela data. Se estas 99 carteiras forem agregadas, tem-se então o modelo de gestão via algoritmos genéticos, com base em previsões de retornos por redes neurais e previsões de risco por *GARCH*. A performance deste portfólio dinamicamente calculado a cada dia pode então ser aferida, de modo a determinar se proporciona bons retornos sem incorrer em grandes riscos. Na Figura 23 e Figura 24, a seguir, tem-se os resultados obtidos. A Figura 23 traz a evolução diária dos retornos das carteiras e a Figura 24 mostra uma evolução do portfólio, começando com base 100 no dia 23/05/2003. Na Figura 24 estão ainda indicadas as duas grandes quedas que o portfólio sofre ao longo do período analisado.

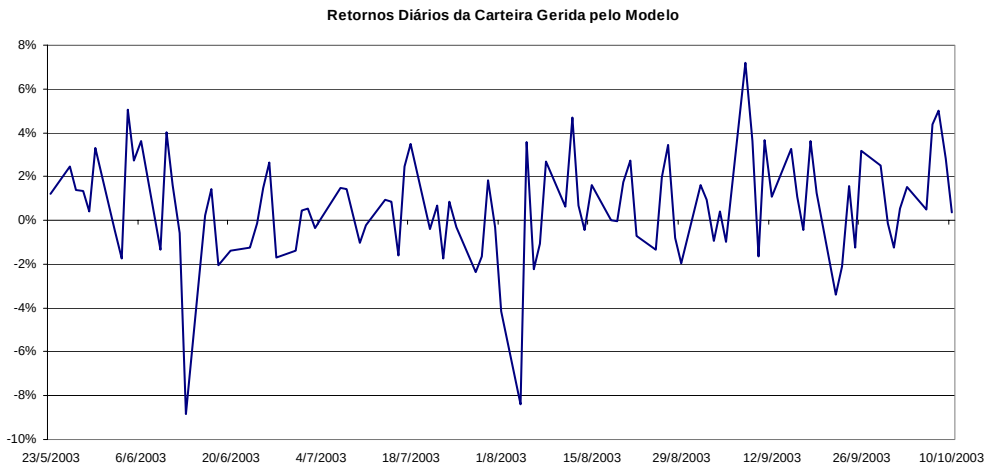


Figura 23: Retornos contínuos diários das carteiras otimizadas por Algoritmo Genético, entre 23/05/2003 e 10/10/2003.

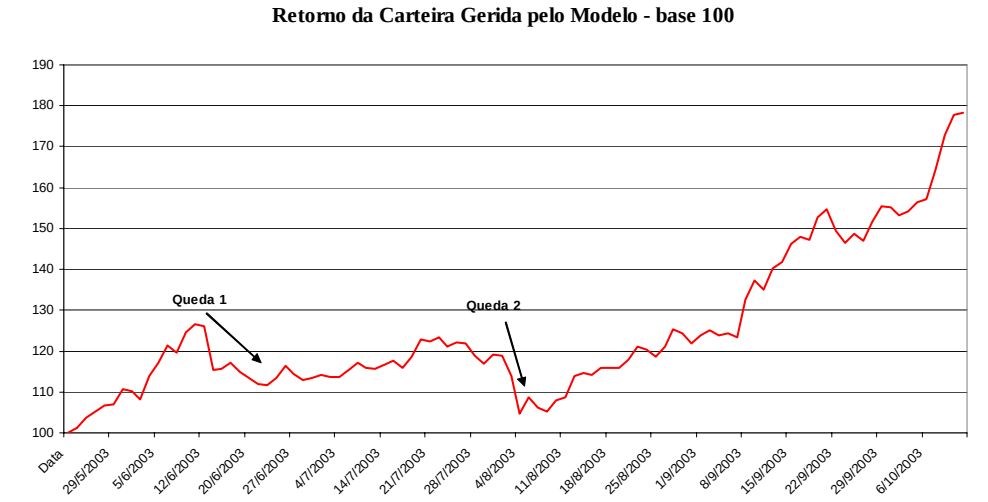


Figura 24: Evolução do Portfólio montado a partir das carteiras diárias

Os resultados se mostram bastante satisfatórios, mas com algumas ressalvas. O portfólio teria gerado um retorno de 78,4% nos 100 dias de gestão. Contudo, este resultado não leva em conta nenhum custo de operação, o que poderia reduzir substancialmente a rentabilidade, ainda mais para carteiras reotimizadas diariamente. Além disso, há uma volatilidade talvez excessiva na carteira, fruto da concentração em alguns poucos papéis. Uma maneira de resolver isso seria através da imposição de novas restrições ao algoritmo genético, proibindo por exemplo, a alocação de mais de 50% dos recursos em uma só ação. Assim, seria possível capturar ainda boa parte da rentabilidade sem incorrer em tanta volatilidade.

6.6. Conclusões

Como é possível observar, especialmente nas seções 6.3.4 e 6.5.3, os resultados obtidos pelos modelos aqui descritos e implementados são bastante significativos. No primeiro caso, a obtenção de uma estrutura (*feedforward* com uma camada escondida, e 5 neurônios nesta camada) e um algoritmo (Levenberg-Marquardt) de rede neural artificial que proporciona excelentes resultados em termos de previsões de preços de ações. No segundo, um modelo que integra previsões de retornos por redes neurais – realizadas pelo modelo anterior, previsões de risco com um modelo GARCH (1,1) e um algoritmo genético para otimizar dinamicamente a alocação do portfólio, cuja performance em termos de rentabilidade, no período analisado, se mostrou bastante boa, mesmo sofrendo algumas quedas que podem ser consideradas expressivas ao longo do período.

7. CONCLUSÕES

"O melhor meio de prever o futuro é inventá-lo."
Alan Key[1]

7.1. Conclusões

As evidências apresentadas no capítulo anterior mostraram a possibilidade de, através de um processo cuidadoso de modelagem e construção experimental, usar redes neurais artificiais para a previsão de preços de ativos financeiros. Embora a discussão sobre a previsibilidade inerente dos mercados ainda seja um tópico em aberto (vide seção 2.3.3), aparentemente a capacidade de redes, especialmente aquelas treinadas com algoritmo Levenberg-Marquardt, preverem preços futuros de ações, pode ser considerada como bastante real. De fato, os resultados obtidos subseqüentemente ao desenvolvimento do modelo de previsão de retornos, para alocação de carteiras, mostram que a capacidade de prever adequadamente os preços futuros proporciona benefícios claros em termos de ganho de rentabilidade.

Além da constatação da capacidade de redes neurais em prever o mercado, é também bastante clara a conclusão de que não é qualquer rede que é capaz de fazê-lo. Foram testadas, de maneira controlada, várias redes, envolvendo muitas combinações de topologia e algoritmo de treinamento, e apenas uma combinação se mostrou clara e indubitavelmente capaz de prever, consistentemente, os preços futuros de ações. Esta combinação, do algoritmo Levenberg-Marquardt com uma topologia de poucos neurônios na camada escondida mostrou resultados consistentes, e superou, com grande distância, aqueles obtidos por outros algoritmos, seja os simples, como o *Backpropagation*, quanto os mais complexos, como o BFGS.

Além das redes neurais, a outra ferramenta de inteligência artificial aqui discutida, os algoritmos genéticos, atingiu plenamente os objetivos que se colocaram de antemão, quais sejam: (i) obter de maneira rápida e eficiente carteiras que otimizassem a relação risco-retorno, a partir de *inputs* de previsões de retornos e risco; (ii) alcançar tal objetivo dentro de um espaço de busca amplo, composto de

[1] Apud Bass, 2000.

todas as combinações possíveis de 25 ações diferentes. Vale mencionar a fácil escalabilidade do modelo, ou seja, realizar o mesmo trabalho com 100 ações é apenas uma questão de força bruta computacional, daqui por diante.

7.2. Trabalho Futuros

O trabalho científico é um edifício em permanente construção, e este trabalho não se pretende isolado do resto. Se do levantamento da literatura ficou patente a ausência de trabalhos na mesma área (inteligência artificial aplicada a finanças) no Brasil, mais evidente ficou a distância separando o estado-da-arte sendo pesquisado lá fora. A partir daí, algumas avenidas futuras a serem percorridas podem ser levantadas:

- *Incorporar mais dados aos modelos:* aqui se esbarra no problema do tamanho do mercado acionário brasileiro – não há muitas ações com liquidez suficiente para proporcionar dados confiáveis, além daquelas já consideradas neste trabalho. Uma alternativa seria trabalhar com papéis no exterior.
- *Trabalhar com funções objetivo complexas dentro das redes neurais*: como visto no capítulo 5, muitos autores trabalham com uma junção de algoritmos genéticos e redes neurais, de modo a ter funções objetivo (para determinar o erro ao longo do treinamento) mais complexas. Para melhorar a performance já muito boa do algoritmo Levenberg-Marquardt, esta pode ser uma boa opção.
- *Trabalhar com "Redes em Comitê"*: gerar várias redes simultaneamente e trabalhar com uma forma de comitê de redes, com competição entre elas, e determinação de quais componentes afetam positiva ou negativamente sua performance. Este caminho é seguido por Shadbolt e Taylor (Eds.) (2003), com resultados bastante promissores.
- *Incorporar outros paradigmas de inteligência artificial*: lógica *fuzzy*, outros tipos de redes neurais, teoria do caos, dinâmica não-linear, são todas ferramentas avançadas sendo aplicadas por pesquisadores ao redor do mundo na tarefa de tentar prever os mercados. Avançar nestas direções pode trazer resultados consideráveis, também em mercados emergentes como o brasileiro.

8. BIBLIOGRAFIA

ALLEN, F. e KARJALAINEN, R. “Using genetic algorithms to find technical trading rules” *in Journal of Financial Economics 51*, pp. 245–271, 1999.

ARMANO, G. et al. “Stock Market Prediction by a mixture of Genetic-Neural Experts” *in International Journal of Pattern Recognition and Artificial Intelligence Vol.16 No.5*, pp. 501-526, 2002.

BECKER, L.A. e SESHADRI, M. *GP-evolved Technical Trading Rules can outperform Buy and Hold*, Worcester: Dept. of Computer Science, Worcester Polytechnic Institute, 2002.

BERNSTEIN, P. L. *Against the Gods: The Remarkable Story of Risk*, Nova York: John Wiley & Sons, 1998.

BERNSTEIN, P. L. *Capital Ideas: The improbable origins of modern Wall Street*, Nova York: Touchstone Books, 1993.

BHATTACHARYYA, S. et al. “Knowledge-Intensive Genetic Discovery in Foreign Exchange Markets” *in IEEE Transactions On Evolutionary Computation*, vol. 6, no. 2, 169-181, 2002.

BOYD, M. e KAASTRA, I. “Designing a neural network for forecasting financial and economic time series” *in Neurocomputing 10*, pp. 215-236, 1996.

BRUNI, A. L. e FAMÁ, R. “Moderna Teoria de Portfólios: é possível captar, na prática, os benefícios decorrentes de sua utilização?” in *Resenha BM&F 128*, pp. 19-32, 1998.

BUSCEMA, M. e SACCO, L. “Feedforward Networks in financial predictions: the future that modifies the present” *in Expert Systems, Vol. 17, No. 3*, 2000.

CAMPBELL, J. Y. et al. *The Econometrics of Financial Markets*, Princeton: Princeton University Press, 1997.

CHAN, M.C. et al. *Financial Time Series Forecasting by Neural Network using conjugate gradient learning algorithm and multiple linear regression weight initialization*, Hong Kong: Dept. of Computing, The Hong Kong Polytechnic University, 2002.

CHEN, A.P. et al. "Establishing an Index Arbitrage Model by applying Neural Networks method – a case study of Nikkei 225 Index" *in International Journal of Neural Systems Vol.11 No.5*, pp. 489-496, 2001.

CHEN, S.H. (Ed.) *Genetic Algorithms and Genetic Programming in Computational Finance*, Nova York: Kluwer Academic Publishers, 2002.

CHEN, S.H. *Can We Believe that Genetic Algorithms would help without actually seeing them work in Financial Data Mining? Part 1, Theoretical Foundations*, Taipei: National Chengchi University, 2000.

CHENOWETH, T. et al. "Embedding Technical Analysis into Neural Network based trading systems" *in Journal of Applied Artificial Intelligence 10*, pp. 523-541, 1996.

DEBOECK, G.J. (Ed.) *Trading on the Edge – Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets*, Nova York: John Wiley & Sons, 1994.

DEMUTH, H. e BEALE, M. *Neural Network Toolbox – for use with MATLAB – User's Guide Version 4*, Natick: The Mathworks, Inc., 2001.

EAKINS, S.G. e STANSELL, S.R. "Can value-based stock selection criteria yield superior risk-adjusted returns: an application of neural networks" *in International Review of Financial Analysis 12*, pp. 83-97, 2003.

FERNÁNDEZ-RODRÍGUEZ, F. et al. “On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid stock market” *in Economic Letters 69*, pp. 89-94, 2000.

FIGUEIREDO, A. C. et al. “A Utilização da Teoria de Carteiras de Markowitz e do Modelo de Índice Único de Sharpe no Mercado de Ações Brasileiro em 1999” in *Resenha BM&F 141*, pp. 51-59, 2000.

FORTUNA, E. *Mercado Financeiro – Produtos e Serviços*, São Paulo: Qualitymark Editora, 2002.

FREEDMAN, R.S. e DIGIORGIO, R. “A Comparison of Stochastic Search Heuristics for Portfolio Optimization” *in Proc. 2nd International Conference on Artificial Intelligence Applications on Wall Street*, pp.149-151, 1993.

FREITAS, S.O. e SOUZA, A.A. “Utilização de redes neurais na precificação de opções” *in Resenha BM&F 150*, pp. 63-73, 2002.

GILES, C.L. et al. “Rule Inference for Financial Prediction using Recurrent Neural Networks” *in Proc. IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, pp. 253-259, 1997.

GRUBER, E. et al. *Modern Portfolio Theory and Investment Analysis*, Nova York: John Wiley & Sons, 2003.

HAMID, S.A. e IQBAL, Z. “Using neural networks for forecasting volatility of S&P 500 Index futures prices” *in Journal of Business Research 5881*, pp. 1-10, 2003.

HARLAND, Z. *Using Nonlinear Neurogenetic models with profit related objective functions to trade the US TBond future*, Nova York: Krueger Research, 2000.

HAYKIN, S. *Redes Neurais – Princípios e Fundamentos*, São Paulo: Bookman, 2001.

JORION, P. *Value at Risk: The New Benchmark for Controlling Market Risk*, Nova York: McGraw-Hill, 1997.

KALYVAS, E. *Using Neural Networks and Genetic Algorithms to Predict Stock Market Returns*, Manchester: University of Manchester – Department of Computer Science, 2001.

KIM, S.H. e CHUN, S.H. "Graded forecasting using an array of bipolar predictions: application of probabilistic neural networks to a stock market index" *in International Journal of Forecasting 14*, pp. 323-337, 1998.

KORCZAK, J. e LIPINSKI, P. *Evolutionary Approach to Portfolio Optimization*, Research Report 2001/05, University Louis Pasteur, Illkirch, 2001.

KORCZAK, J. e ROGER, P. *Stock Timing using genetic algorithms*, Research Report 2001/08, University Louis Pasteur, Illkirch, 2001.

LAWRENCE, R. *Using Neural Networks to Forecast Stock Market Prices*, Manitoba: University of Manitoba – Department of Computer Science, 1997.

LAZO LAZO, J. G. *Sistema Híbrido Genético-Neural para Montagem e Gerenciamento de Carteiras de Ações*, Rio de Janeiro: PUC-RJ, 2000.

LEUNG, M.T. et al. "Forecasting stock indices: a comparison of classification and level estimation models" *in International Journal of Forecasting 16*, pp. 179-190, 2000.

LI, J. e TSANG, E.P.K. “Reducing Failures in Investment Recommendations using Genetic Programming” *in Proc. Sixth International Conference on Computing in Economics and Finance*, Barcelona, pp.1-14, 2000.

LI, J. e TSANG, E.P.K. *Combining Ordinal Financial Predictions with Genetic Programming*, Essex: Department of Computer Science, University of Essex, 1999.

LI, J. e TSANG, E.P.K. *Improving Technical Analysis Predictions: An Application of Genetic Programming*, Essex: Department of Computer Science, University of Essex, 1999.

MALKIEL, B.G. *A Random Walk Down Wall Street, Completely Revised and Updated 8th Edition*, Nova York: W.W. Norton & Company, 2003.

McINTYRE-BHATTY, Y.T. “Neural network analysis and the characteristics of market sentiment in the financial markets” *in Expert Systems Vol. 17 No. 4,* 2000.

McNELIS, P. “A Neural Network Analysis of Brazilian Stock Prices: Tequila Effects vs. Pisco Sour Effects” *in Journal of Emerging Markets Vol.1 No.2,* 1996.

McNELIS, P. *Computational Finance with Neural Networks*, Washington: Department of Economics, Georgetown University, 2003.

MITCHELL, M. *An Introduction to Genetic Algorithms*, Cambridge: The MIT Press, 1996.

MOISÃO, R.L.M. e PIRES, F.M. *Prediction Model, based on Neural Networks, for Time Series with Origin in Chaotic Systems*, Working Paper, Universidade de Évora, 2001.

OLSON, D. e MOSSMAN, C. “Neural Network forecast of Canadian stock returns using accounting ratios” *in International Journal of Forecasting 19,* pp. 453-465, 2003.

RAPOSO, R. C. T. e CRUZ, A. J. O. *Stock Market prediction based on fundamentalist analysis with fuzzy-neural networks*, Rio de Janeiro: Instituto de Matemática-UFRJ, 1999.

REFENES, A.P. (Ed.) *Neural Networks in the Capital Markets,* Londres: John Wiley & Sons, 1995.

REFENES, A.P. e ZAPRANIS, A.D. *Principles of Neural Model Identification, Selection and Adequacy: With Applications in Financial Econometrics*, Londres: Springer-Verlag, 1999.

SHADBOLT, J. e TAYLOR, J. G. (Eds.) *Neural Networks and the Financial Markets – Predicting, Combining and Portfolio Optimization*, Londres: Springer-Verlag, 2003.

SKABAR, A. e CLOETE, I. “Neural Networks, Financial Trading and the Efficient Markets Hypothesis” *in Proc. 25th Australasian Computer Science Conference*, Melbourne, Australia, 2001.

SORNETTE, D. *Why Stock Markets Crash – Critical Events in Complex Financial Systems*, Princeton: Princeton University Press, 2003.

SWALES, G.S. e YOON, Y. “Applying artificial neural networks to investment analysis” *in Financial Analysts Journal* 48, pp. 78-80, 1992.

THAWORNWONG, S. et al. “Genetic Algorithms and Neural Networks for Stock Trading Prediction and Technical Signal Optimization” *in Proc. Decision Sciences Institute 2002 Annual Meetings*, pp. 776-781, 2002.

THE MATHWORKS, INC. *GARCH Toolbox User's Guide, version 1*, 1999.

TRIPPI, R.R. e LEE, J.K. *Artificial Intelligence in Finance & Investing – State-of-the-Art Technologies for Securities Selection and Portfolio Management*, Chicago: Irwin Professional Publishing, 1996.

VERLEYSEN, M. e BODT, E. e LENDASSE, A. "Forecasting financial time series through intrinsic dimension estimation and non-linear data projection" *in Proc. IWANN'99 – International Work Conference on Artificial and Natural Neural Networks*, pp. II-596-II–605, 1999.

WATADA, J. e ODA, K. "Formulation of a Two-Layered Boltzmann Machine for Portfolio Selection" *in International Journal of Fuzzy Systems Vol.2 No.1*, pp 39-44, 1999.

WONG, B.K. et al. "Neural Network Applications in Business: A Review and Analysis of the literature (1988-1995)" *in Decision Support Systems,* vol. 19, pp. 301-320, 1997.

XU, L. e CHEUNG, Y.M "Adaptative Supervised Learning Decision Networks for Trading and Portfolio Management" *in Journal of Computational Intelligence in Finance November/December 1997*, pp. 11-16, 1997.

YAO, J. e TAN, C.L. *A Study on Training Criteria for Financial Time Series Forecasting*, Cingapura: Dept. of Computer Science, National University of Singapore, 2001.

YAO, J. e TAN, C.L. *Guidelines for Financial Forecasting with Neural Networks*, Cingapura: Dept. of Computer Science, National University of Singapore, 2001.

ZAHEDI, F. *Intelligent Systems for Business, Expert Systems with Neural Networks*, Londres: Wodsworth Publishing Inc., 1993.

ZEKI-SUSAC, M. *Neural Networks in Investment Profitability Decisions*, Zagreb: University of Zagreb - Department of Organization and Informatics, 1999.

ZHANG, G. et al. “Forecasting with Neural Networks: The state of the art” *in International Journal of Forecasting 14*¸ pp. 35-62, 1998.

APÊNDICE 1 - DETALHAMENTO DOS MODELOS & CÓDIGOS

A1.1. Modelo de Redes Neurais

Ao longo da seção 6.3, foram discutidos os detalhes dos modelos de redes neurais utilizados para a realização de previsões de retornos de 25 ações diferentes. Neste apêndice, vai-se expandir e detalhar o funcionamento deste modelo.

O primeiro passo era agrupar e tratar os dados. Para cada uma das 25 ações, foi construída uma matriz contendo, nas colunas, cada um dos 11 tipos de dados a serem usados nas previsões, e nas linhas, cada observação destes dados, entre as datas 10/05/1999 e 10/10/2003. As matrizes têm o formato exemplificado na Figura A1.1, para a ação Vale do Rio Doce PNA (VALE5). Os códigos usados nesta matriz têm a correspondência com tais dados de acordo com o exposto na Tabela A1.1.

VALE5	med	max	min	vol	dolar	pre30d	pre360d	ibov	dow	s&p	nasdaq
10/5/1999	38,71	39,30	37,60	10.210.534,00	1,71	26,74	24,02	10.675,00	10.863,20	1.347,76	2.233,51
11/5/1999	35,95	38,05	35,10	18.927.128,00	1,73	26,43	23,87	10.734,00	10.979,00	1.362,84	2.308,78
12/5/1999	34,37	35,60	34,00	14.867.629,00	1,72	26,02	23,77	10.691,00	10.972,10	1.365,40	2.333,78
13/5/1999	36,02	36,49	35,00	26.421.234,00	1,73	22,11	22,39	10.568,00	10.791,30	1.341,03	2.263,06
14/5/1999	36,11	36,90	35,49	14.124.289,00	1,75	21,05	22,68	10.441,00	10.655,20	1.328,72	2.270,93
17/5/1999	35,15	36,50	34,60	3.806.988,00	1,74	23,12	22,74	10.211,00	10.646,00	1.328,05	2.264,81
18/5/1999	34,92	35,75	34,10	7.599.070,00	1,74	22,63	22,51	10.264,00	10.677,30	1.322,18	2.239,18
19/5/1999	35,01	35,50	34,50	8.170.938,00	1,75	22,41	22,77	10.172,00	10.674,80	1.305,33	2.204,33
⋮	⋮										⋮
3/10/2003	116,32	117,70	113,51	27.068.785,00	2,89	19,34	17,81	17.089,00	9.572,31	1.029,85	1.375,32
6/10/2003	116,66	117,80	115,60	17.301.407,00	2,88	19,25	17,84	17.273,00	9.594,98	1.034,35	1.381,70
7/10/2003	117,19	117,80	115,84	26.919.569,00	2,87	19,17	17,87	17.470,00	9.654,61	1.039,25	1.392,56
8/10/2003	115,76	116,70	113,20	27.585.973,00	2,84	19,08	17,76	17.804,00	9.630,90	1.033,78	1.382,40
9/10/2003	113,47	114,40	112,05	23.600.763,00	2,84	19,04	17,71	17.708,00	9.680,01	1.038,73	1.396,95
10/10/2003	111,75	113,99	111,31	17.835.810,00	2,84	19,01	17,70	17.676,00	9.674,68	1.038,06	1.404,87

Figura A1.1: Exemplo de Matriz de Dados de Entrada para VALE5

Dado	Código
Preço Médio da Ação	*med*
Preço Máximo da Ação	*max*
Preço Mínimo da Ação	*min*
Volume Negociado	*vol*
Cotação do Dólar	*dolar*
Taxa de Juros de 30 dias	*pre30d*
Taxa de Juros de 360 dias	*pre360d*
Índice Bovespa	*ibov*
Índice Dow Jones Industrials	*dow*
Índice S&P 500	*s&p*
Índice Nasdaq 100	*nasdaq*

Tabela A1.1: Tipos de Dados e seus códigos na Matriz de Dados

Cada uma das 25 ações tem sua matriz correspondente, onde as principais mudanças estão nos quatro primeiros dados, sendo que os sete restantes são dados do mercado em geral, não sendo específicos de cada ação. Uma vez construídas as

matrizes, estas eram introduzidas no *MATLAB*, onde seriam manipuladas de acordo com o modelo proposto: primeiramente pré-processadas, depois introduzidas no modelo de rede neural, para a realização dos processos de treinamento e validação das redes neurais.

O pré-processamento, conforme discutido na seção 6.3.2, consiste em duas etapas: normalização e *PCA*. Ambas são implementas diretamente via *MATLAB*. A normalização consiste em transformar a série histórica em uma série com média zero e variância unitária. Este processo está exemplificado na série de preços médios de ARCZ6 (Aracruz PNB) da Figura A1.2.

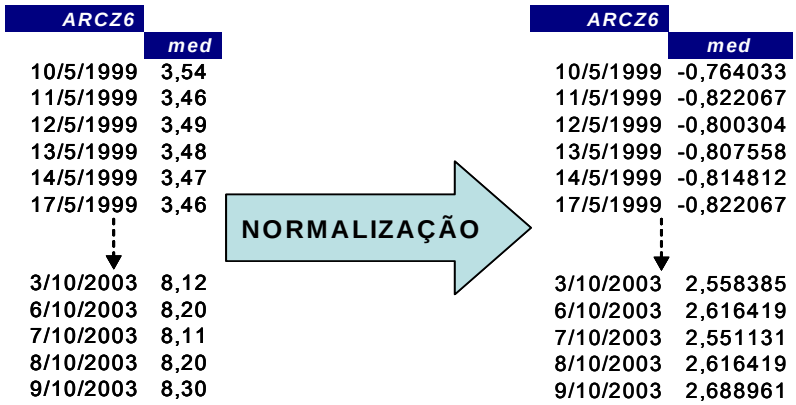


Figura A1.2: Normalização dos Dados

Todas as séries temporais passavam pelo processo de normalização. Após isto, eram submetidas à Análise de Componentes Principais, uma técnica mais complexa, e que por isto mesmo não será detalhada aqui. É suficiente dizer que tal técnica reduz a dimensão da matriz de entrada de dados, visto que supõe-se haver uma certa correlação entre os dados de entrada. Esta técnica limpa tal correlação, e mantém na matriz de entrada apenas dos dados relevantes para a realização das previsões. Para cada uma das 25 ações, o processo de PCA (*Principal Component Analysis*) traz resultados diferentes, e portanto não vai ser aqui generalizado. Apenas um exemplo de como funciona vai ser dado, no contexto de todo o processo de pré-processamento da matriz de entrada de dados. Na Figura A1.3, é mostrado todo o processo até aqui descrito, para a matriz de entrada de ARCZ6. Primeiro aparece a matriz inicial, depois a matriz com todas as séries já normalizadas, e a seguir a matriz após terem sido deixados apenas seus componentes principais.

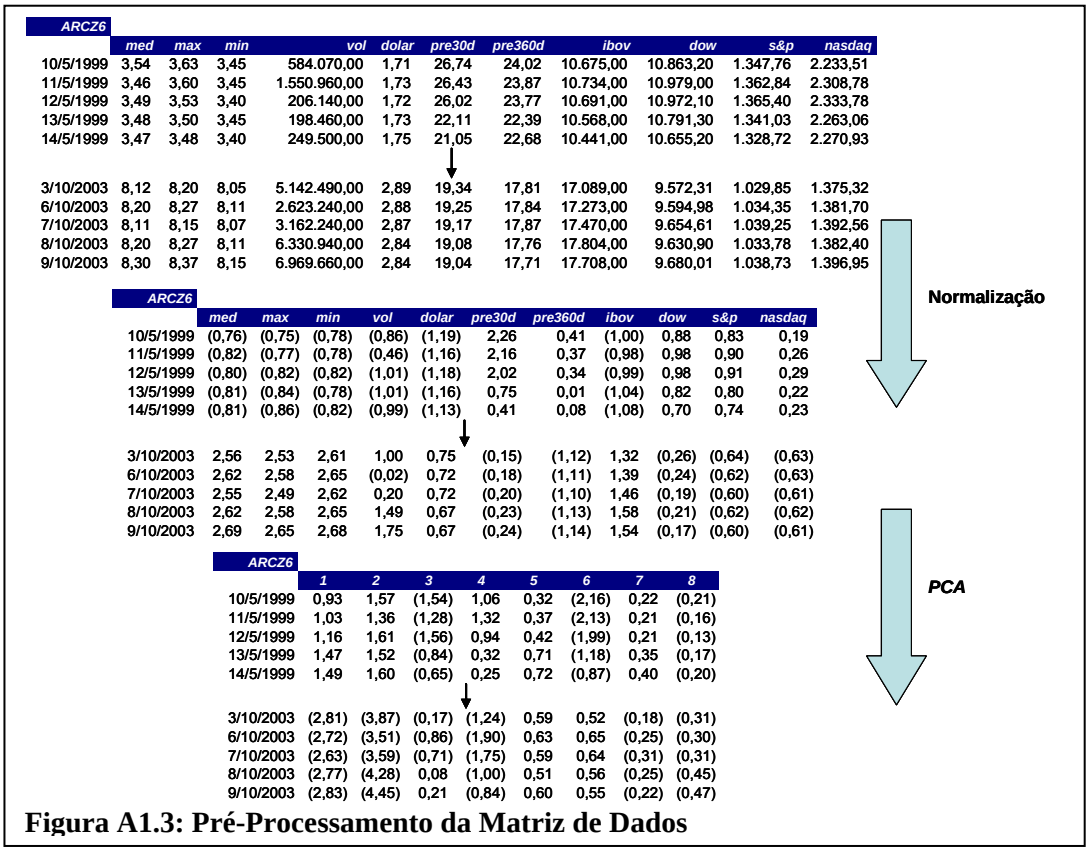


Figura A1.3: Pré-Processamento da Matriz de Dados

Na passagem após o *PCA*, não é mais possível identificar as séries temporais com suas respectivas origens. Contudo, o *software* armazena a matriz de transformação, para que seja possível realizar o pós-processamento dos resultados.

Tendo a matriz de dados a serem introduzidos na rede neural, falta apenas a série cujos padrões a rede neural deve aprender, ou seja, a série-alvo do processo de treinamento. Este pode ser resumido da seguinte maneira: a rede recebe uma série de dados de entrada (digamos, os oito dados pré-processados, para o dia 10/05/1999, da ação ARCZ6, conforme visto na Figura A1.3), e deve tentar prever o preço da mesma ação no dia 11/05/1999. Esta previsão vai então ser comparada com a previsão dita "perfeita", ou seja, o preço conhecido no dia 11/05/1999, e a partir daí um erro de previsão deve ser calculado. Este erro é então introduzido na rede, cujos pesos são adaptados na tentativa de minimizá-lo. Este processo iterativo acaba por gerar (espera-se) uma rede neural que seja capaz de boas previsões. Sendo assim, pode ser construída a série-alvo a ser introduzida no modelo de rede neural: ela é simplesmente a série de preços médios da ação *X*, deslocada em 1 dia. Tem-se assim

todos os dados a serem introduzidos na rede neural. Na Figura A1.4 estes estão exibidos.

ARCZ6	med	max	min	vol	dolar	pre30d	pre360d	ibov	dow	s&p	nasdaq	Target d+1
10/5/1999	3,54	3,63	3,45	584.070,00	1,71	26,74	24,02	10.675,00	10.863,20	1.347,76	2.233,51	3,46
11/5/1999	3,46	3,60	3,45	1.550.960,00	1,73	26,43	23,87	10.734,00	10.979,00	1.362,84	2.308,78	3,49
12/5/1999	3,49	3,53	3,40	206.140,00	1,72	26,02	23,77	10.691,00	10.972,10	1.365,40	2.333,78	3,48
13/5/1999	3,48	3,50	3,45	198.460,00	1,73	22,11	22,39	10.568,00	10.791,30	1.341,03	2.263,06	3,47
14/5/1999	3,47	3,48	3,40	249.500,00	1,75	21,05	22,68	10.441,00	10.655,20	1.328,72	2.270,93	3,46
3/10/2003	8,12	8,20	8,05	5.142.490,00	2,89	19,34	17,81	17.089,00	9.572,31	1.029,85	1.375,32	8,20
6/10/2003	8,20	8,27	8,11	2.623.240,00	2,88	19,25	17,84	17.273,00	9.594,98	1.034,35	1.381,70	8,11
7/10/2003	8,11	8,15	8,07	3.162.240,00	2,87	19,17	17,87	17.470,00	9.654,61	1.039,25	1.392,56	8,20
8/10/2003	8,20	8,27	8,11	6.330.940,00	2,84	19,08	17,76	17.804,00	9.630,90	1.033,78	1.382,40	8,30
9/10/2003	8,30	8,37	8,15	6.969.660,00	2,84	19,04	17,71	17.708,00	9.680,01	1.038,73	1.396,95	8,16

Figura A1.4: Matriz completa com Dados e Série-alvo

Note-se como a série-alvo nada mais é do que a série *"méd"*, ou seja, os preços médios da ação, deslocada em um dia. Note-se também que a série-alvo passa pelo mesmo pré-processamento já descrito, basicamente a normalização.

Uma vez preparados os dados, pode ser iniciado o processo de treinamento da rede neural. A matriz de dados é introduzida no modelo, cujos pesos iniciais são iniciados randomicamente. O modelo separa os dados em partes de treinamento, validação e teste, conforme explicado na seção 6.3.3 e resumido na Figura 9. A partir daí, segue o processo de treinamento, controlando os erros do modelo de maneira a gerar uma boa capacidade de previsão. Ao fim do processo de treinamento, o modelo gera previsões para 100 dias adiante, conforme a Figura 9. Todas as saídas da rede passam pelo pós-processamento, sendo renormalizadas, para que os resultados sejam comparáveis com os resultados desejados.

O código completo de MATLAB usado para definir as redes está na seção A1.4.

A1.2. Modelo *GARCH*

O funcionamento do modelo *GARCH* é ainda mais simples. Para ele é necessário apenas um *input*, os preços médios das ações. A partir dos preços, o modelo calcula os retornos diários contínuos para os últimos 21 dias (este é o tamanho da janela utilizada), estima os parâmetros da equação *GARCH*, e realiza a previsão para *t+1* conforme esta equação.

A previsão é armazenada, e a janela desliza um dia, reiniciando todo o processo. Este está resumido na Figura A1.5.

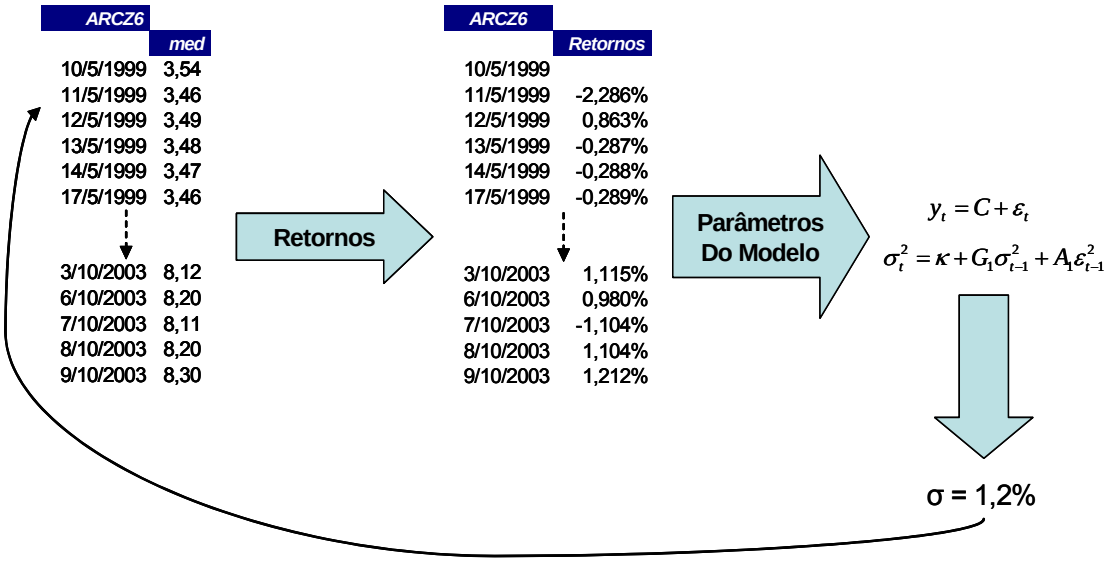


Figura A1.5: Funcionamento do Modelo *GARCH*

O código de MATLAB escrito para implementação deste modelo está na seção A1.4.2.

A1.3. Modelo de Algoritmo Genético

O modelo de redes neurais gera previsões de retornos para as datas entre 23/05/2003 e 10/10/2003. O modelo *GARCH* gera previsões de risco para estas mesmas datas. O modelo de algoritmo genético une estas duas previsões para, dentro do paradigma teórico da moderna teoria de portfólios, realizar a otimização da alocação das carteiras. O modelo é implementado via um suplemento para *Excel*, chamado *Evolver*. Este suplemento já tem implementado um conjunto de funções para criação e manipulação de algoritmos genéticos. Assim, basta definir a função utilidade, as restrições e os outros parâmetros do modelo (todos delineados na seção 6.5.2 e resumidos na Tabela 17), e colocar o modelo para rodar. Como se deseja montar carteiras para todas as datas entre 23/05/2003 e 10/10/2003, o modelo deve ser rodado 99 vezes, cada uma gerando uma alocação diferente entre as 25 ações para as quais se tem previsão de retornos e riscos.

A Figura A1.6 mostra a planilha em *Excel* com a janela do *Evolver* aberta, onde são definidos os parâmetros do modelo. Nesta janela está definida a função utilidade e as restrições. Os outros parâmetros do modelo são definidos em uma janela subseqüente.

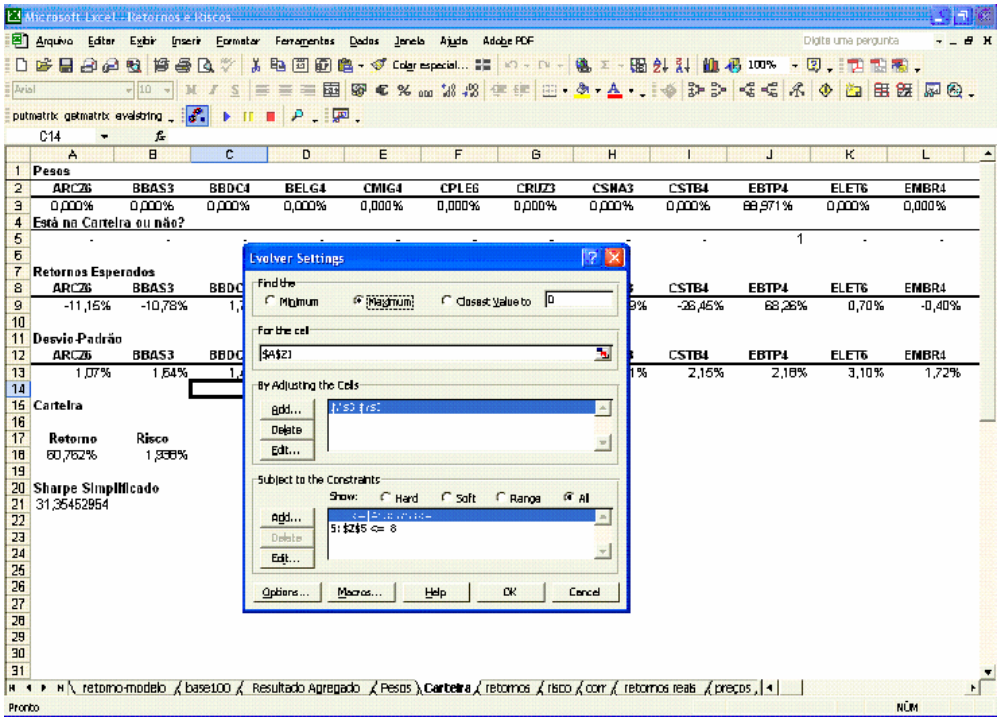


Figura A1.6: Janela do *Excel* com *Evolver*

Após estruturar o modelo, deve-se colocar o *Evolver* para rodar. A partir daí, os resultados podem ser acompanhados iterativamente, através de uma janela especial. Esta é mostrada na Figura A1.7.

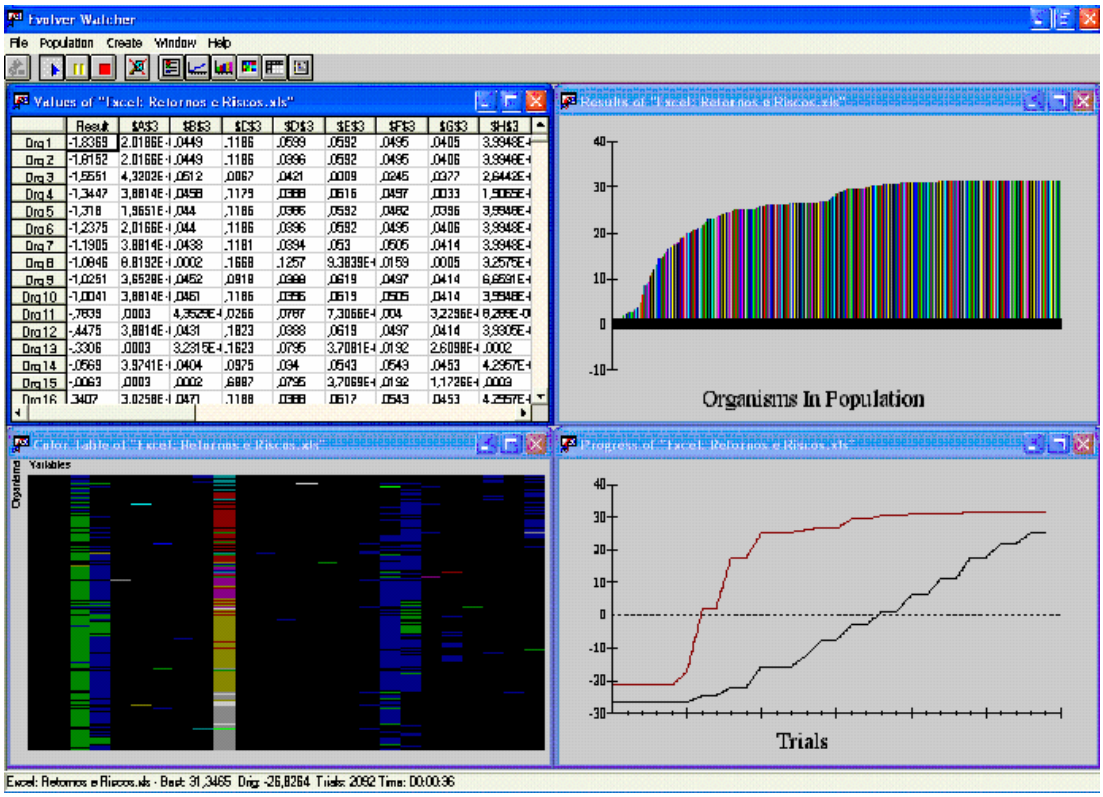


Figura A1.7: *Evolver* em operação

Nela pode ser observado, no canto esquerdo superior, os resultados obtidos (em termos de melhor organismo na população, ou seja, melhor carteira) numericamente. No canto superior direito aparece a distribuição de utilidade dos organismos na população da iteração atual. No canto inferior esquerdo aparece a distribuição de "genes" dentro do "genoma" completo da iteração atual, ou seja, mostra onde há diversidade genética, em uma tabela de cores. Por fim, no canto inferior direito aparece a evolução dos resultados ao longo das iterações já passadas.

O modelo pára de rodar após um número pré-determinado de iterações, e mostra como resultado as alocações de cada papel na carteira de ações. A partir daí, estas podem ser facilmente manipuladas e analisadas usando as funcionalidades inerentes do *Excel*.

A1.4. Principais Códigos

Nesta seção são mostrados e brevemente analisados os principais códigos usados para construção e implementação dos modelos descritos anteiormente. Note-se que todos os códigos estão à disposição dos interessados, bastando contatar o autor.

A1.4.1. Rede Neural

Os modelos de rede neural são implementados através de um conjunto de cinco funções. Cada uma usa um algoritmo de treinamento dentre os que serão analisados: *Backpropagation* com *momentum*, *Backpropagation* Resiliente, Powell-Beale, BFGS, Levenberg-Marquardt. Elas se aproveitam das capacidades do *MATLAB*, que tem dentro do *Neural Network Toolbox* uma série de funções que permitem criar e manipular redes neurais. O código aqui descrito se aproveita disto, e organiza tanto as entradas quanto os resultados da maneira definida pelo modelo.

Dada a semelhança entre as cinco funções, será exemplificado o código de apenas uma delas. A única diferença entre elas está na função de *MATLAB* que define o algoritmo de treinamento a ser utilizado. O código da função "*previsao_retornos_Levenberg_ea*" está a seguir, e implementa uma rede neural com treinamento Levenberg-Marquardt.

O código está em azul e os comentários do autor após o símbolo “%” em preto.

```
function resultado=previsao_retornos_Levenberg_ea(inputs, targets, hidden)
p = inputs;
t = targets;

%Normalização das entradas e targets
%para media zero e variância unitária
[pn, meanp, stdp,tn, meant, stdt] = prestd(p,t);

%Realizar PCA e deixar apenas componentes responsáveis por mais de 0.1% da variação
[ptrans, transMat] = prepca(pn,0.001);

%Dividir os dados em treinamento, validação e teste. A validação será feita com os 100
%preços antes da faixa de teste.O teste com os 100 últimos preços.
[R,Q] = size(ptrans);
iitr = 1:Q-200;
iival = Q-200:Q-100;
iitst = Q-100:Q;
ptr = ptrans(:,iitr);
ttr = tn(:,iitr);
validation.P = ptrans(:,iival);
validation.T = tn(:,iival);
testing.P = ptrans(:,iitst);
testing.T = tn(:,iitst);

%Definir a Rede Neural
%Algoritmo de Treinamento Levenberg-Marquardt 'trainlm'
h = hidden;
net = newff(minmax(ptr), [h 1], {'tansig' 'purelin'}, 'trainlm');
net.trainParam.show = 100;

%Treinar a Rede Neural
[net,tr] = train(net, ptr, ttr, [], [], validation, testing);

%Plotar os erros de treinamento, validacao e teste.
subplot(3,1,1)
plot(tr.epoch,tr.perf,'r',tr.epoch,tr.vperf,':g',tr.epoch,tr.tperf,'-.b')
legend('Treinamento','Validaçao','Teste',-1)
ylabel('Erro Quadratico')

%Simular a rede treinada. Converter os resultados para media e variancia originais.
an = sim(net, ptrans);
a = poststd(an, meant, stdt);
subplot(3,1,2)
plot(t(1000:1099))
hold on
plot(a(1000:1099), 'm')
title('Comparaçao entre targets e previsao da rede')
legend('Target','Previsao')
xlabel('Dias')
ylabel('Preço')

%Fazer analise de regressão outputs da rede treinada x targets.
subplot(3,1,3)
[m,b,r]=postreg(a,t);
resultado=r;
```

A1.4.2. Modelo *GARCH*

O modelo é implementado através de uma função e de um *script*. A função é chamada de "*previsão_risco*" e se utiliza de duas funções que o *MATLAB* já tem previamente implementada, chamadas "*garchfit"* e "*garchpred*". A primeira otimiza os parâmetros do modelo, e a segunda realiza previsões, dados os parâmetros. O script serve para passar os dados de cada uma das 25 ações para a função, e armazenar os resultados, para facilitar a tarefa de análise posterior.

O código da função "*previsão_risco*" está a seguir, com código em azul e comentários após o símbolo "%" em preto.

```
function [volatilidade]=previsao_risco(dados, janela)
%a função previsao_risco adapta um modelo GARCH (1,1) na serie
%temporal de preços de um ativo, fornecida via o argumento 'dados'
%alem disso, o processo é iterativo, através do parâmetro janela,
%onde se define o tamanho da janela deslizante a ser utilizada para
%a realização da previsão.

precos = dados;
j = janela;
size = length(precos)-j;
i = 1;
while (i <= size)
   %calcula os retornos contínuos
   ret = price2ret(precos(i:i+j));
   %estima os parâmetros do modelo GARCH
   [coeff, errors, LLF, innovations, sigma] = garchfit(ret);
   %realiza a previsão um dia a frente
   [sFcast] = garchpred(coeff, ret);
   %guarda a previsão
   vol(i) = [sFcast];
   i = i + 1;
end
%retorna o vetor completo de previsões de risco
[volatilidade]=vol;
end
```

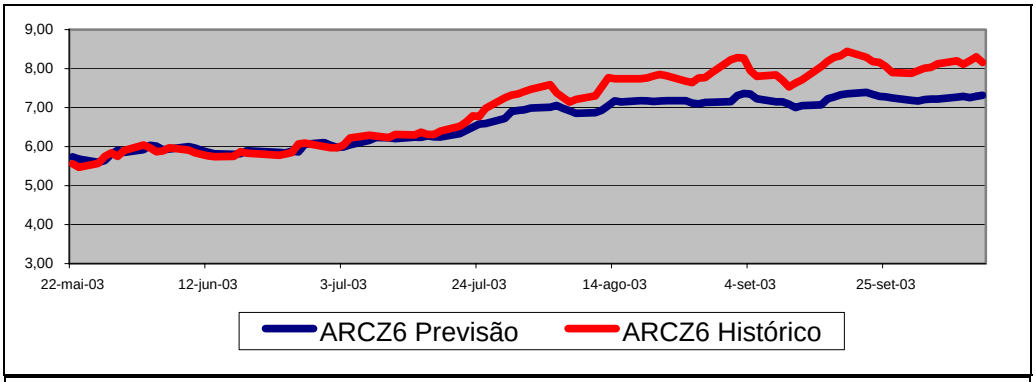
APÊNDICE 2 - RESULTADOS COMPLETOS

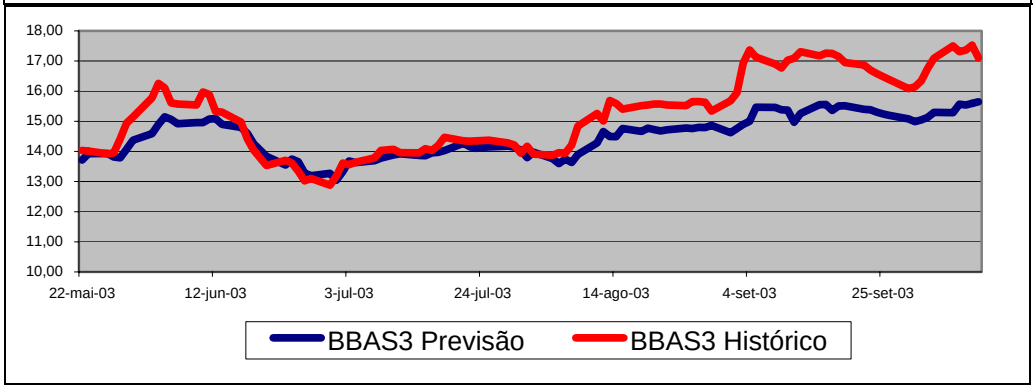
A2.1. Introdução

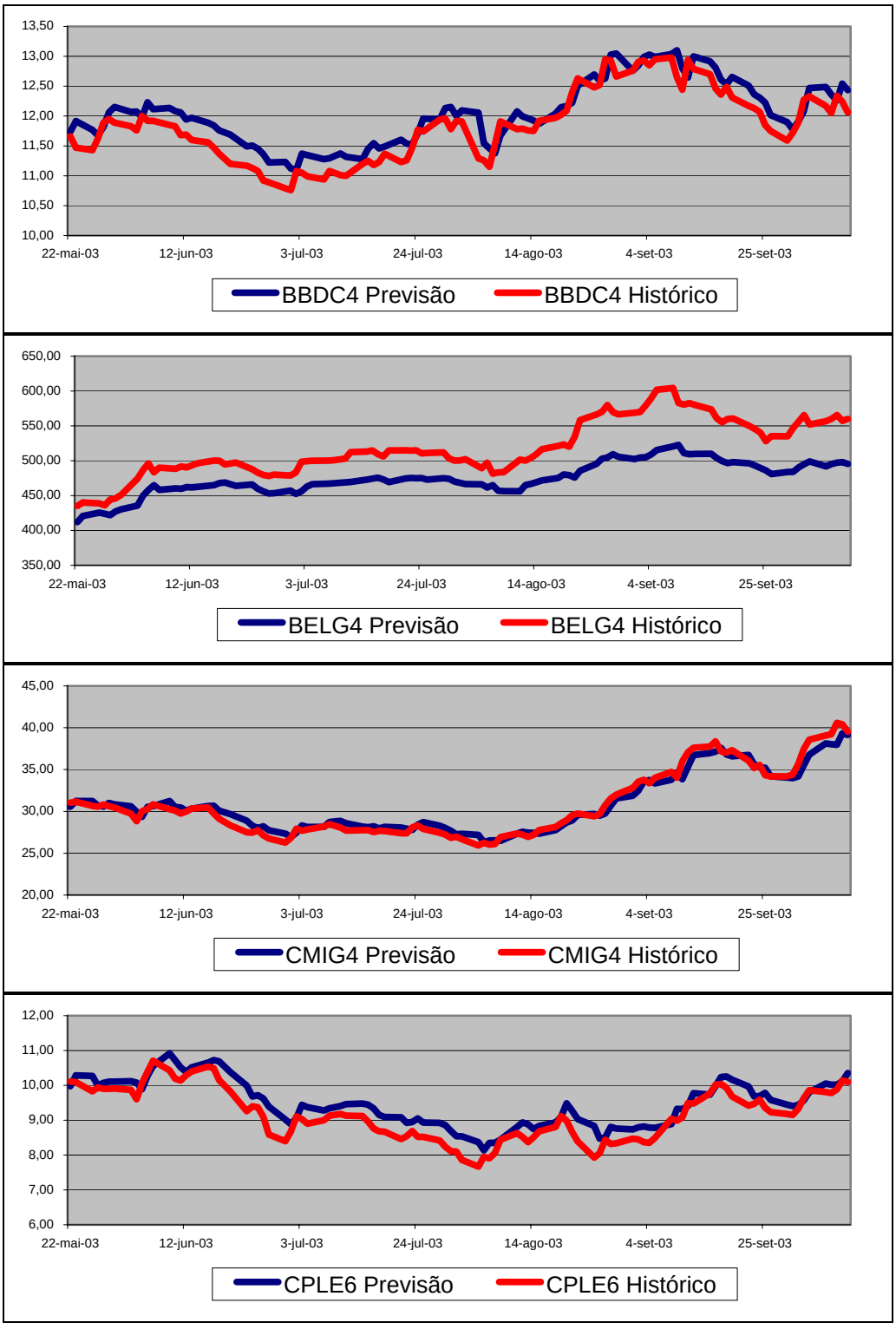
Nas seções 6.3.4 e 6.4 são descritos os principais resultados obtidos nos modelos de previsão analisados, tanto para previsão de retornos como para previsão de riscos. Estes são apresentados em forma de gráficos (os dados numéricos estão à disposição dos interessados).

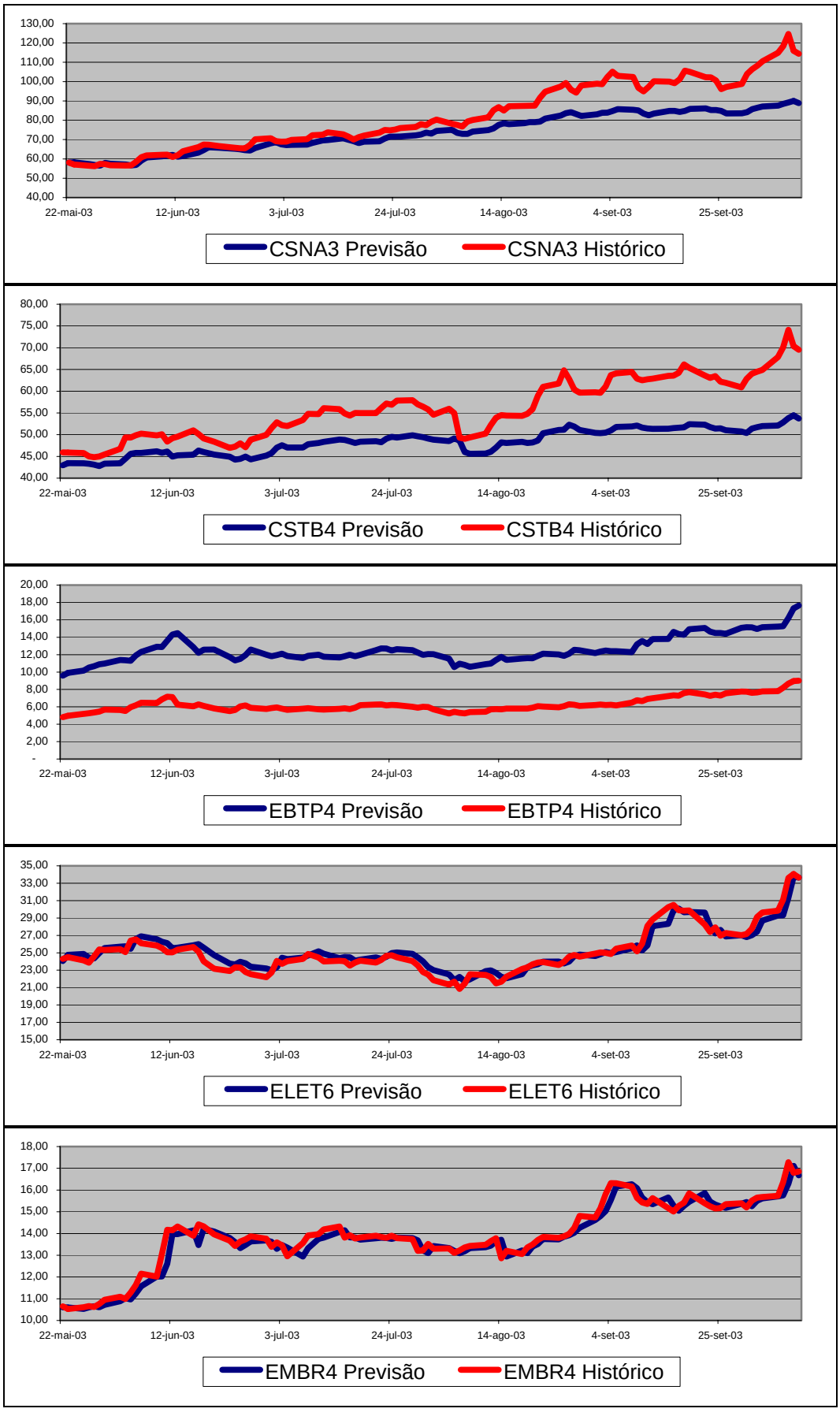
A2.2. Previsão de Retornos

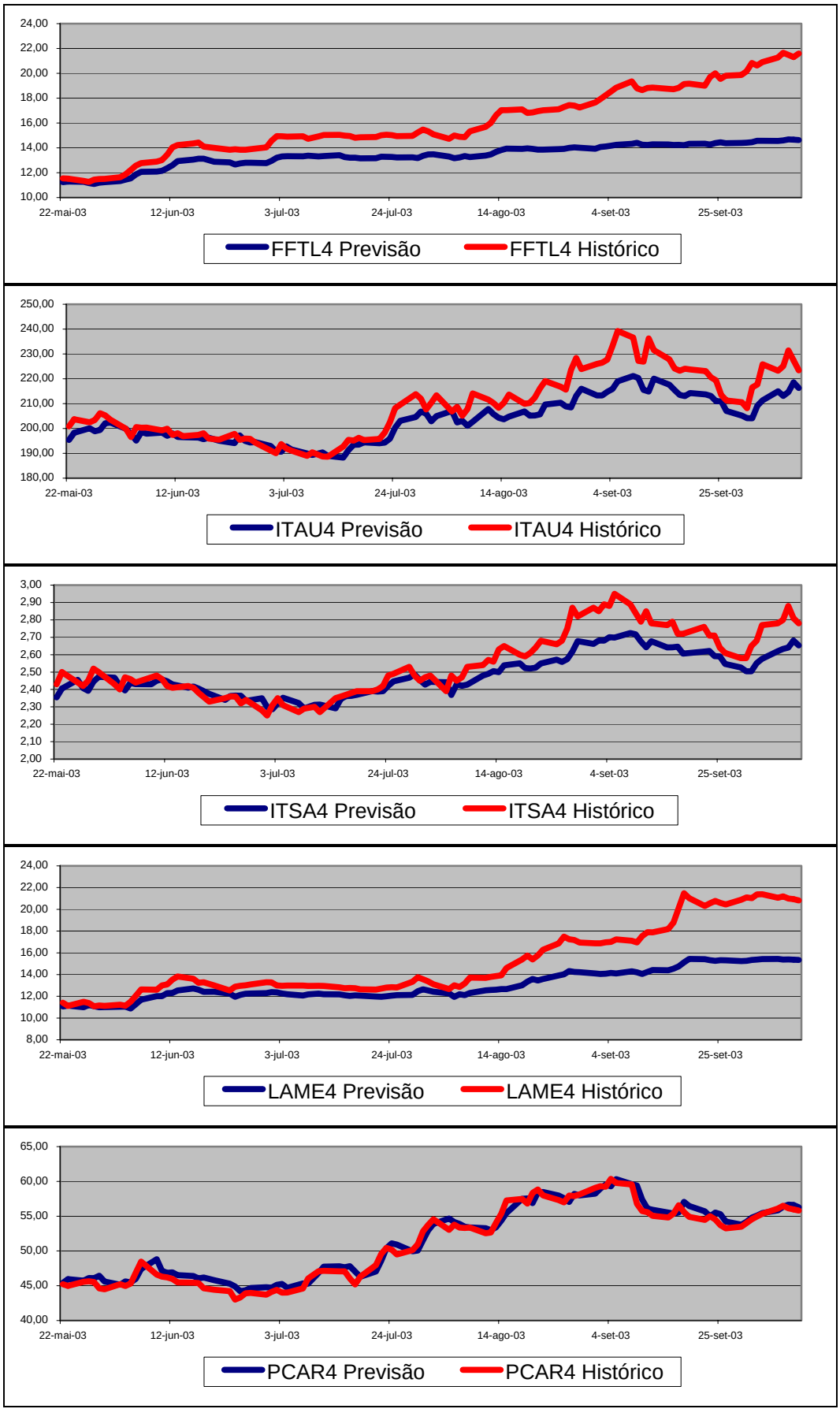
Conforme descrito no item 6.5.1 deste trabalho, foi escolhida uma rede neural com 5 neurônios na camada escondida e algoritmo de treinamento Levenberg-Marquardt para realização das previsões de retornos para cada uma das 25 ações escolhidas. As previsões foram realizadas para o período entre 22/05 e 10/10. Em todos os gráficos a seguir, é mostrada uma comparação entre o preço previsto pela rede neural (curvas em azul) e o preço observado no mercado (ou preço histórico, dado nas curvas em vermelho).

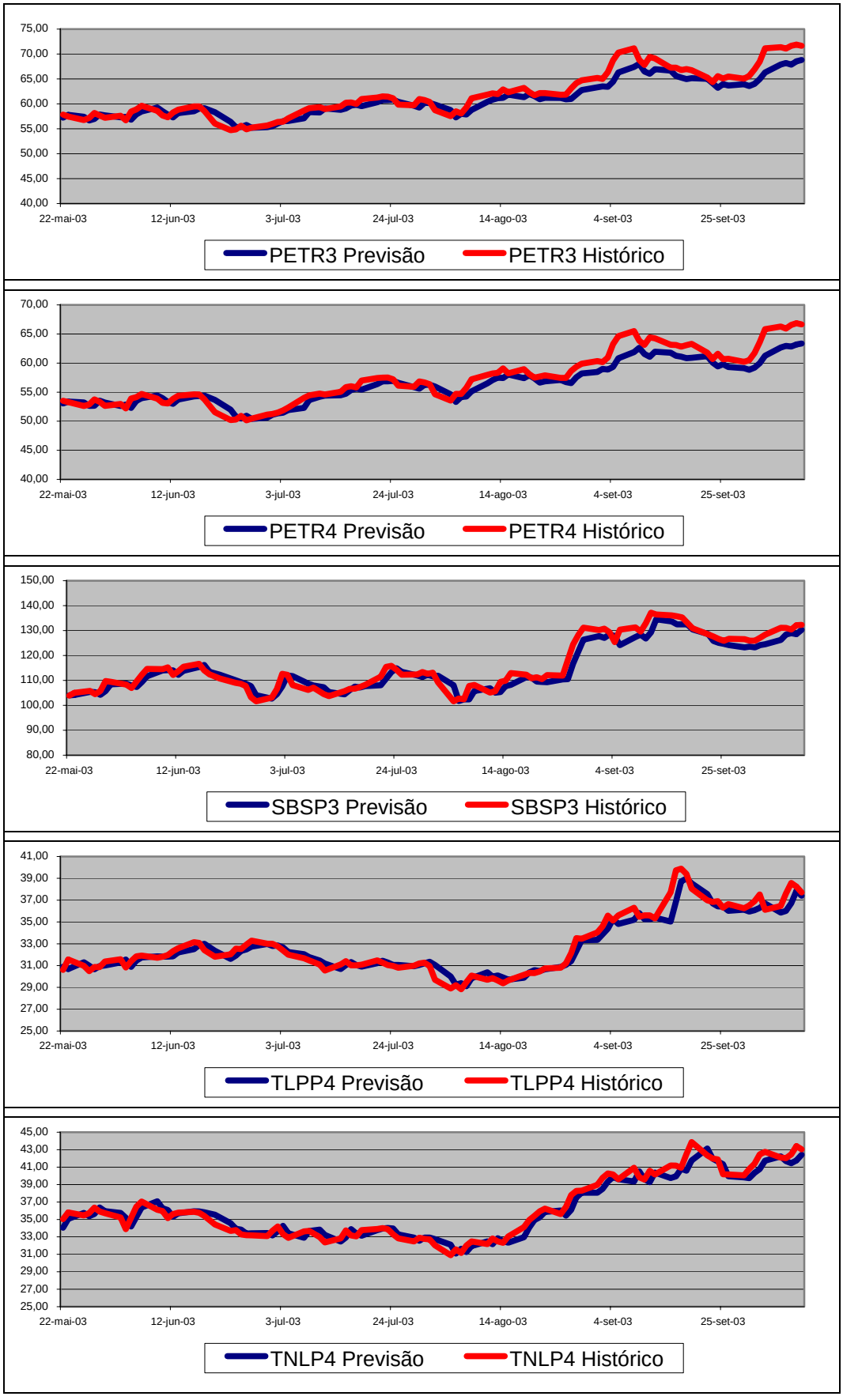


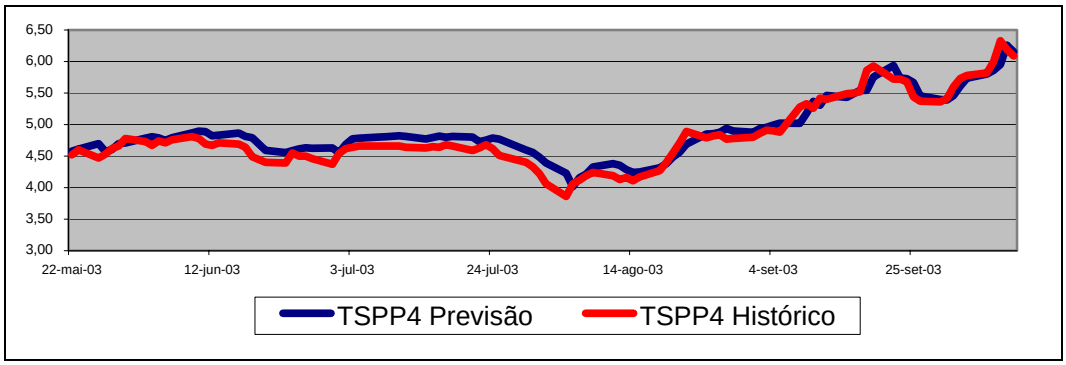


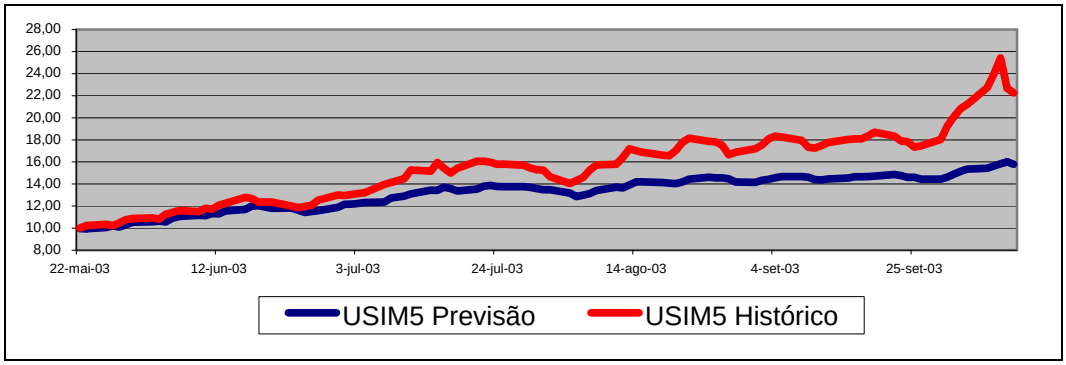

13,50
13,00
12,50
12,00
11,50
11,00
10,50
10,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
BBDC4 Previsão
BBDC4 Histórico
650,00
600,00
550,00
500,00
450,00
400,00
350,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
BELG4 Previsão
BELG4 Histórico
45,00
40,00
35,00
30,00
25,00
20,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
CMIG4 Previsão
CMIG4 Histórico
12,00
11,00
10,00
9,00
8,00
7,00
6,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
CPLE6 Previsão
CPLE6 Histórico

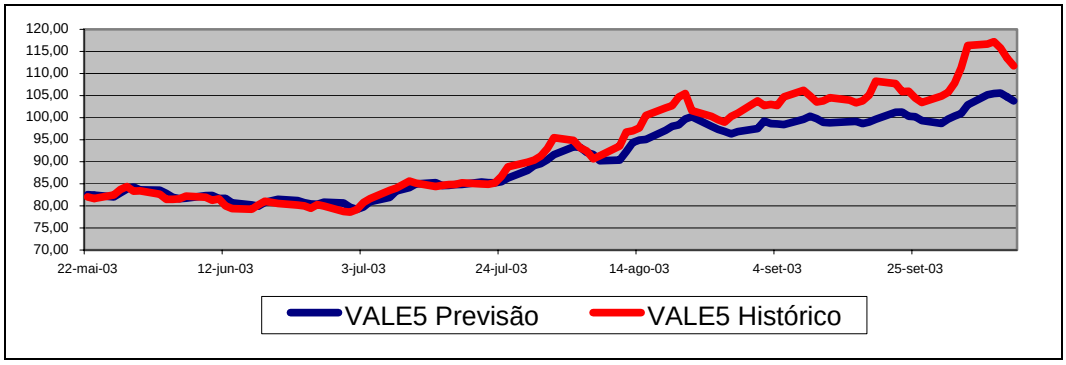

130,00
120,00
110,00
100,00
90,00
80,00
70,00
60,00
50,00
40,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
CSNA3 Previsão
CSNA3 Histórico
80,00
75,00
70,00
65,00
60,00
55,00
50,00
45,00
40,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
CSTB4 Previsão
CSTB4 Histórico
20,00
18,00
16,00
14,00
12,00
10,00
8,00
6,00
4,00
2,00
-
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
EBTP4 Previsão
EBTP4 Histórico
35,00
33,00
31,00
29,00
27,00
25,00
23,00
21,00
19,00
17,00
15,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
ELET6 Previsão
ELET6 Histórico
18,00
17,00
16,00
15,00
14,00
13,00
12,00
11,00
10,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
EMBR4 Previsão
EMBR4 Histórico


24,00
22,00
20,00
18,00
16,00
14,00
12,00
10,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
FFTL4 Previsão
FFTL4 Histórico
250,00
240,00
230,00
220,00
210,00
200,00
190,00
180,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
ITAU4 Previsão
ITAU4 Histórico
3,00
2,90
2,80
2,70
2,60
2,50
2,40
2,30
2,20
2,10
2,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
ITSA4 Previsão
ITSA4 Histórico
24,00
22,00
20,00
18,00
16,00
14,00
12,00
10,00
8,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
LAME4 Previsão
LAME4 Histórico
65,00
60,00
55,00
50,00
45,00
40,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
PCAR4 Previsão
PCAR4 Histórico


75,00
70,00
65,00
60,00
55,00
50,00
45,00
40,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
PETR3 Previsão
PETR3 Histórico
70,00
65,00
60,00
55,00
50,00
45,00
40,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
PETR4 Previsão
PETR4 Histórico
150,00
140,00
130,00
120,00
110,00
100,00
90,00
80,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
SBSP3 Previsão
SBSP3 Histórico
41,00
39,00
37,00
35,00
33,00
31,00
29,00
27,00
25,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
TLPP4 Previsão
TLPP4 Histórico
45,00
43,00
41,00
39,00
37,00
35,00
33,00
31,00
29,00
27,00
25,00
22-mai-03
12-jun-03
3-jul-03
24-jul-03
14-ago-03
4-set-03
25-set-03
TNLP4 Previsão
TNLP4 Histórico



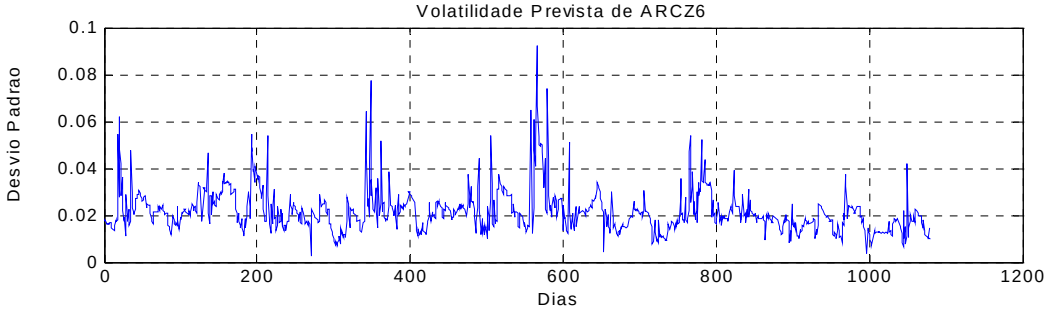


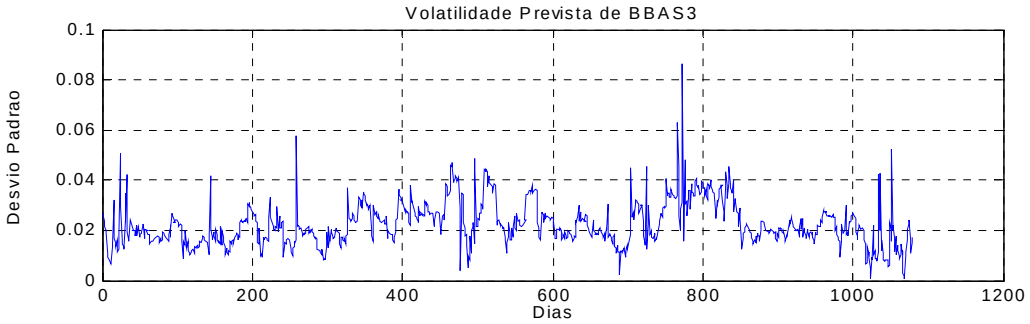


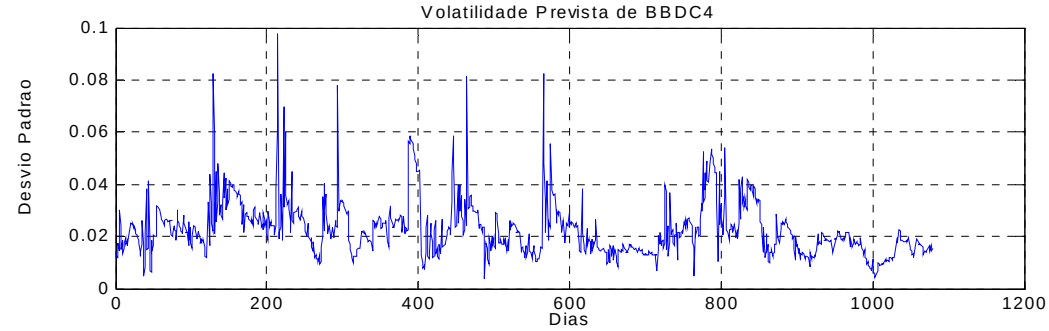
Como é possível constatar, algumas previsões são de melhor qualidade, outras de pior qualidade. No geral, contudo, a rede neural mostra sua capacidade de capturar padrões e realizar boas previsões a partir do aprendizado destes padrões.

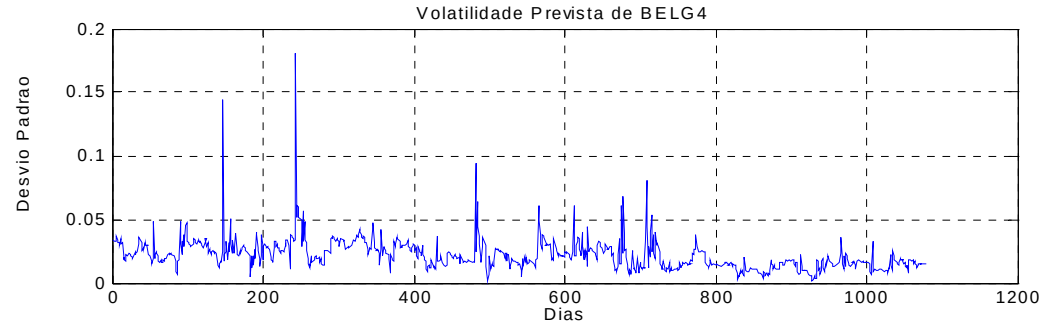
A2.3. Previsão de Riscos

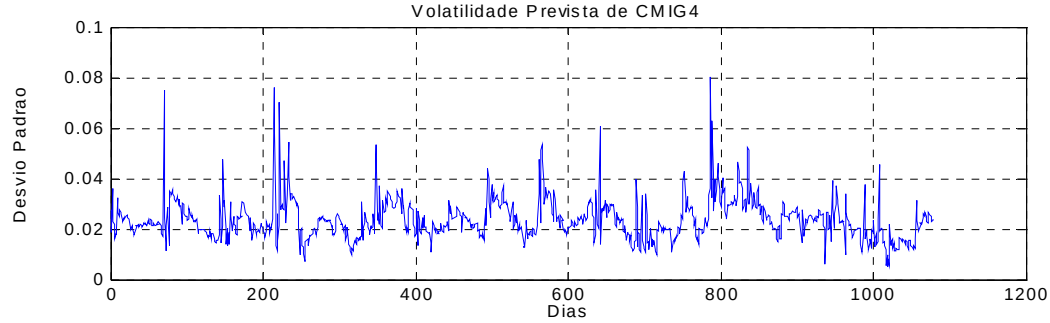
Além das previsões dos retornos futuros para cada uma das 25 ações, também foram realizadas previsões de riscos futuros, utilizando um modelo GARCH (1,1), com janela deslizante de 21 dias. As previsões foram realizadas para um período de 1079 dias, indo de 09/06/1999 até 10/10/2003. A seguir estão apresentadas todas as curvas de risco previsto para cada uma das 25 ações.

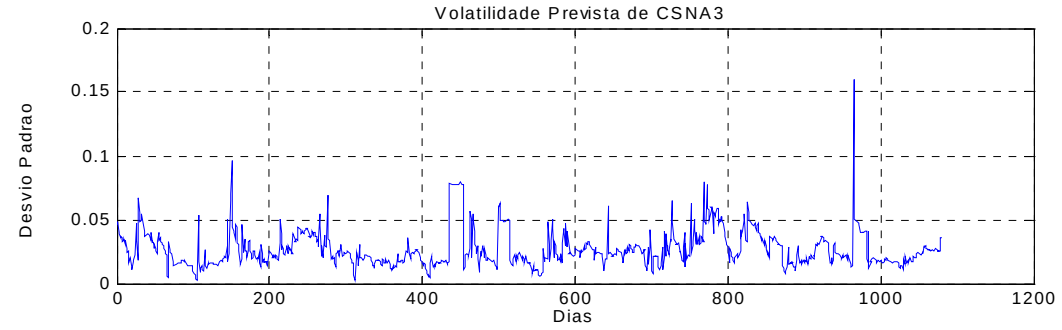

Volatilidade Prevista de ARCZ6
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0
200
400
600
800
1000
1200

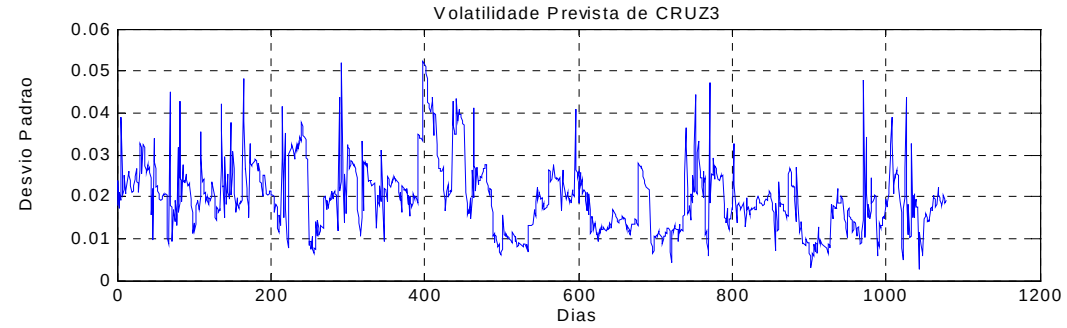

Volatilidade Prevista de BBAS3
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0
200
400
600
800
1000
1200

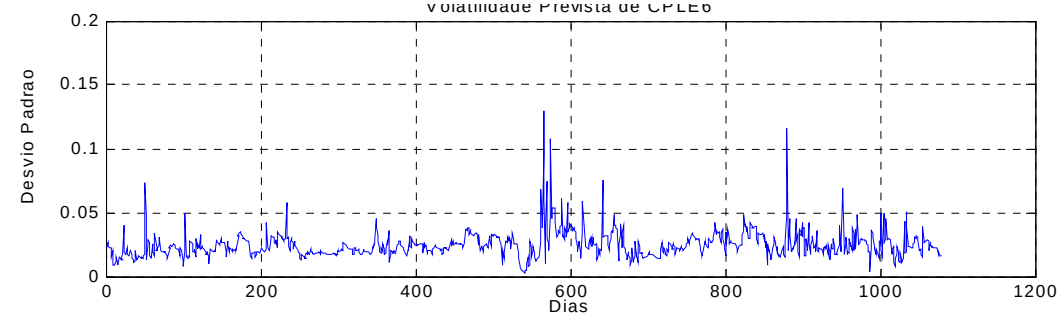

Volatilidade Prevista de BBDC4
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0
200
400
600
800
1000
1200

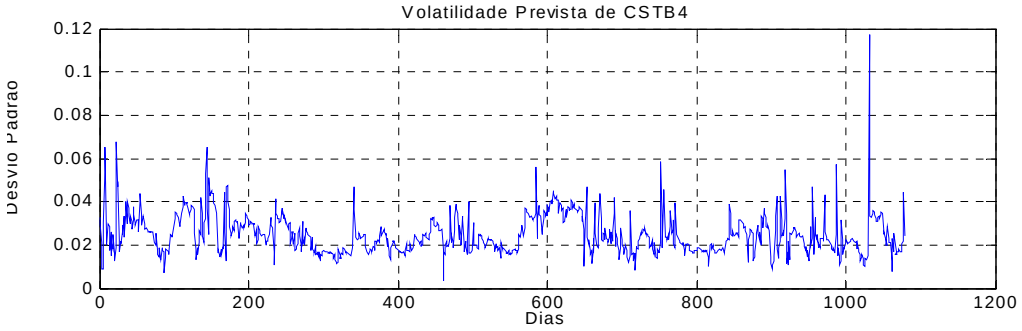

Volatilidade Prevista de BELG4
Desvio Padrao
Dias
0
0.05
0.1
0.15
0.2
0
200
400
600
800
1000
1200

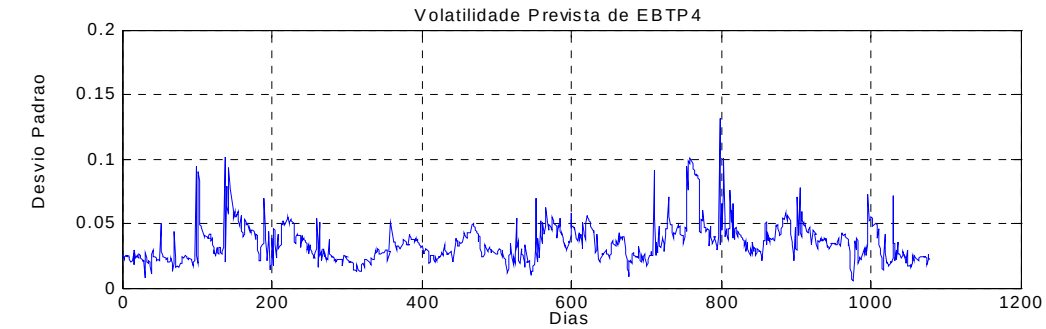

Volatilidade Prevista de CMIG4
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0
200
400
600
800
1000
1200

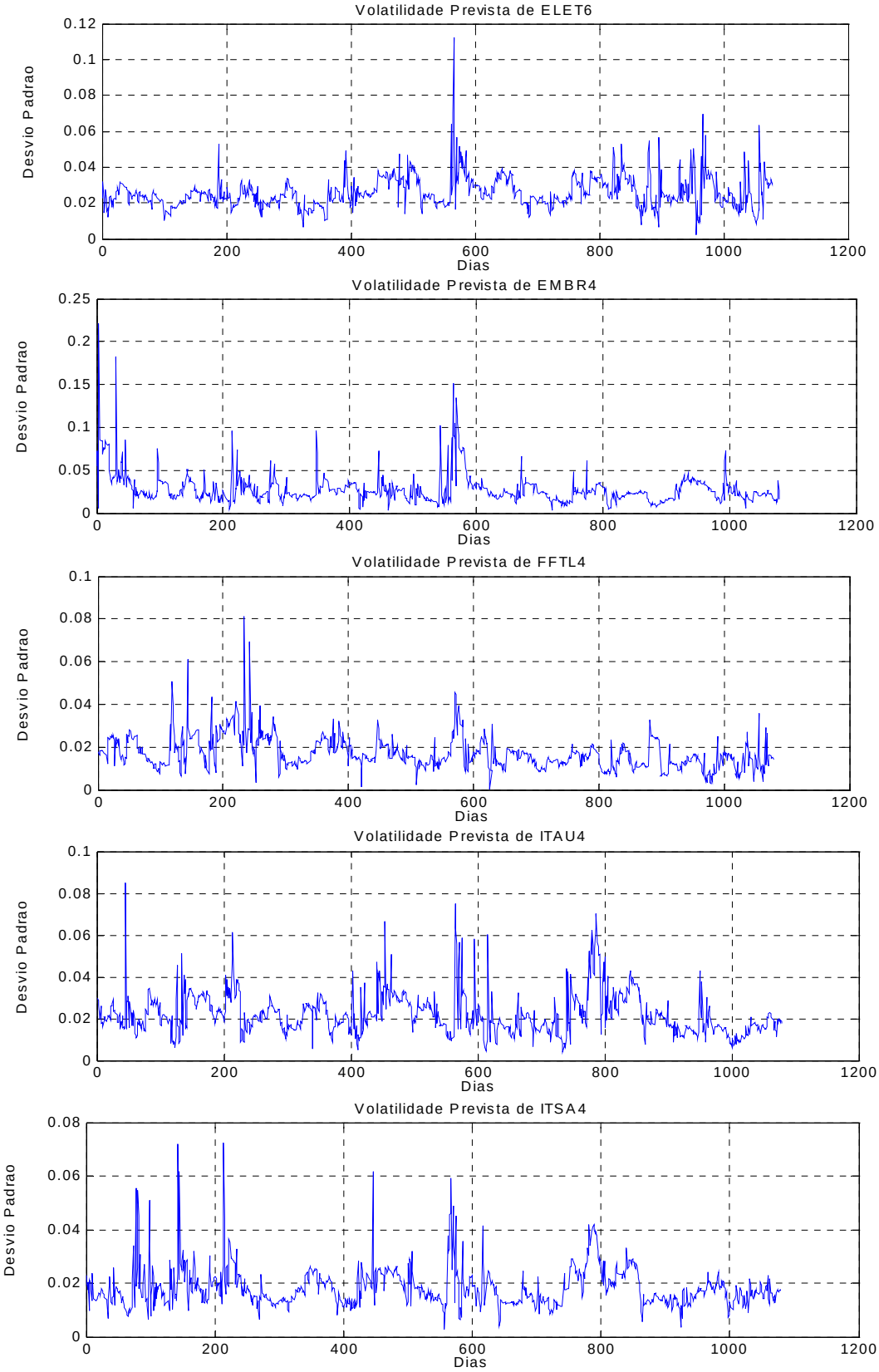

Volatilidade Prevista de CSNA3
Desvio Padrao
Dias
0
0.05
0.1
0.15
0.2
0
200
400
600
800
1000
1200

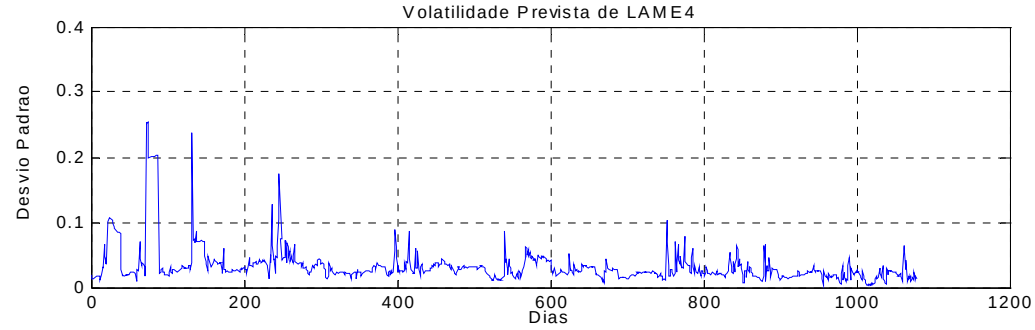

Volatilidade Prevista de CRUZ3
Desvio Padrao
Dias
0
0.01
0.02
0.03
0.04
0.05
0.06
0
200
400
600
800
1000
1200

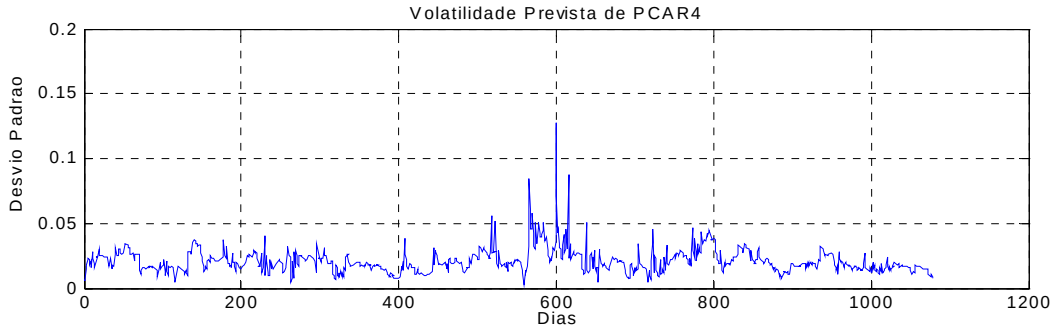

Desvio Padrao
Dias
0
0.05
0.1
0.15
0.2
0
200
400
600
800
1000
1200

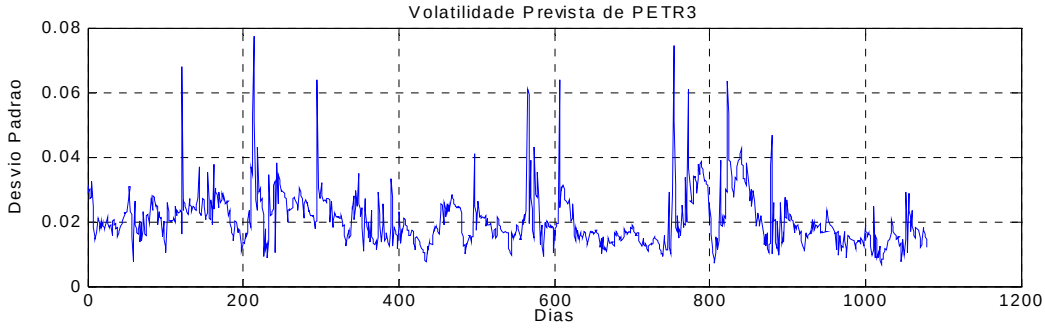

Volatilidade Prevista de CSTB4
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0.12
0
200
400
600
800
1000
1200

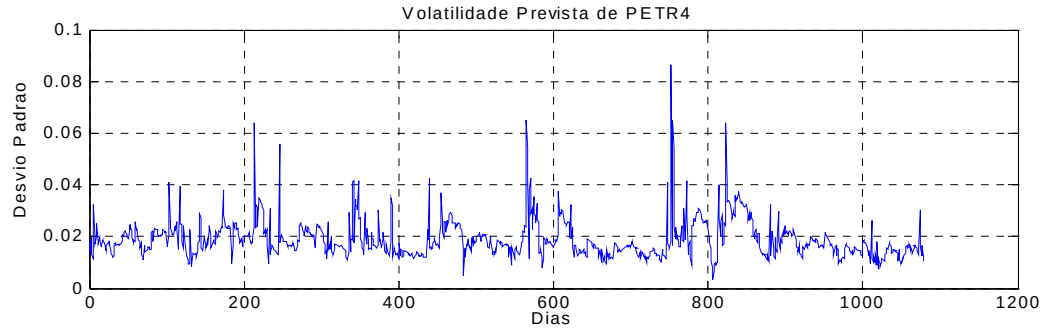

Volatilidade Prevista de EBTP4
Desvio Padrao
Dias
0
0.05
0.1
0.15
0.2
0
200
400
600
800
1000
1200

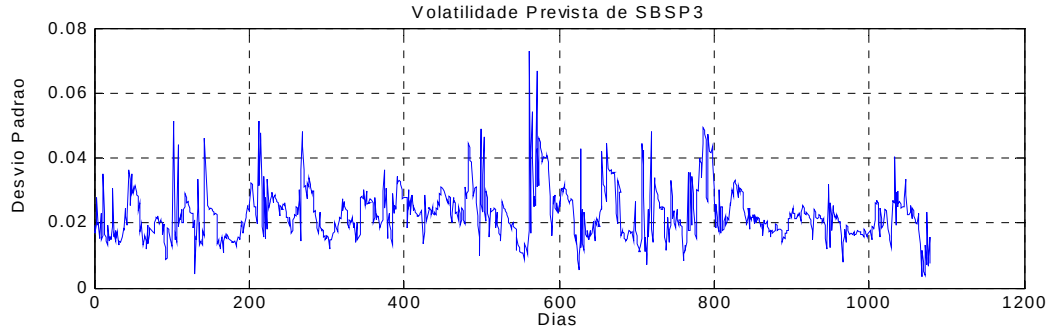

Volatilidade Prevista de ELET6
Desvio Padrao
Dias
Volatilidade Prevista de EMBR4
Desvio Padrao
Dias
Volatilidade Prevista de FFTL4
Desvio Padrao
Dias
Volatilidade Prevista de ITAU4
Desvio Padrao
Dias
Volatilidade Prevista de ITSA4
Desvio Padrao
Dias

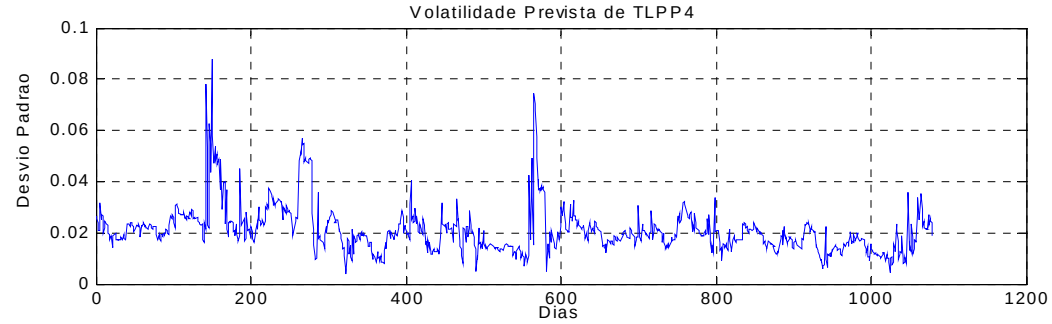

Volatilidade Prevista de LAME4
Desvio Padrao
0.4
0.3
0.2
0.1
0
0
200
400
600
800
1000
1200
Dias

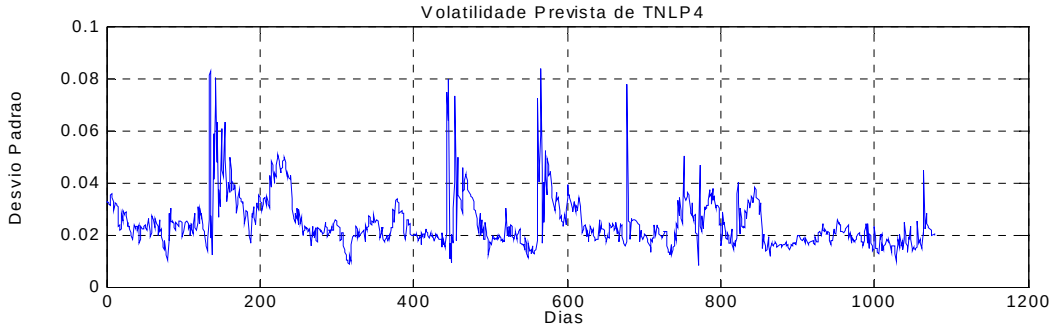

Volatilidade Prevista de PCAR4
Desvio Padrao
0.2
0.15
0.1
0.05
0
0
200
400
600
800
1000
1200
Dias

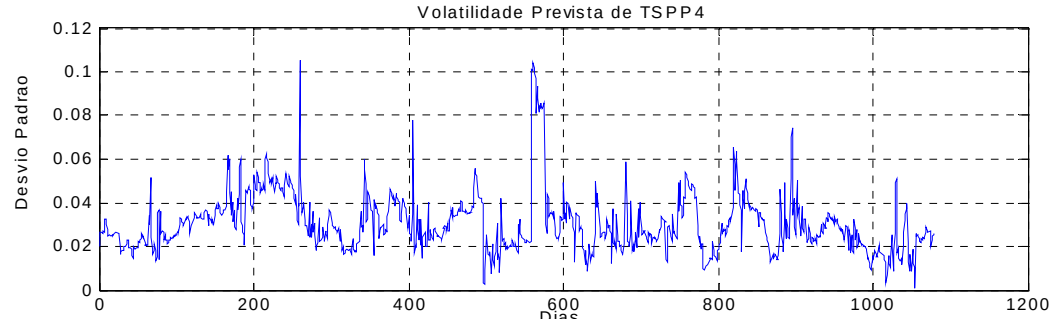

Volatilidade Prevista de PETR3
Desvio Padrao
0.08
0.06
0.04
0.02
0
0
200
400
600
800
1000
1200
Dias

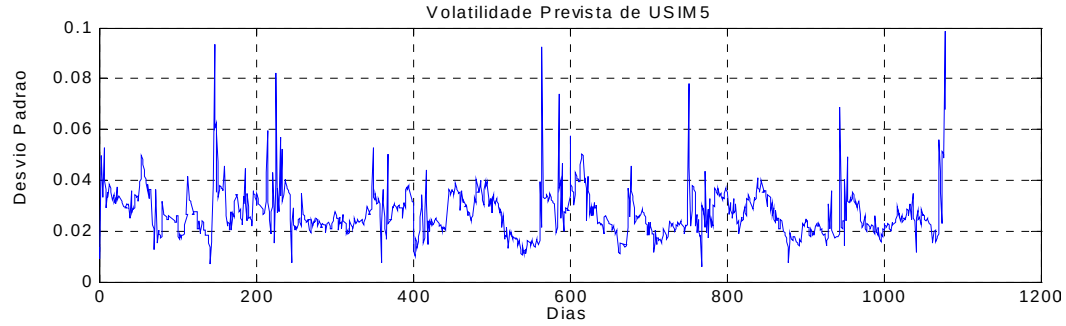

Volatilidade Prevista de PETR4
Desvio Padrao
0.1
0.08
0.06
0.04
0.02
0
0
200
400
600
800
1000
1200
Dias

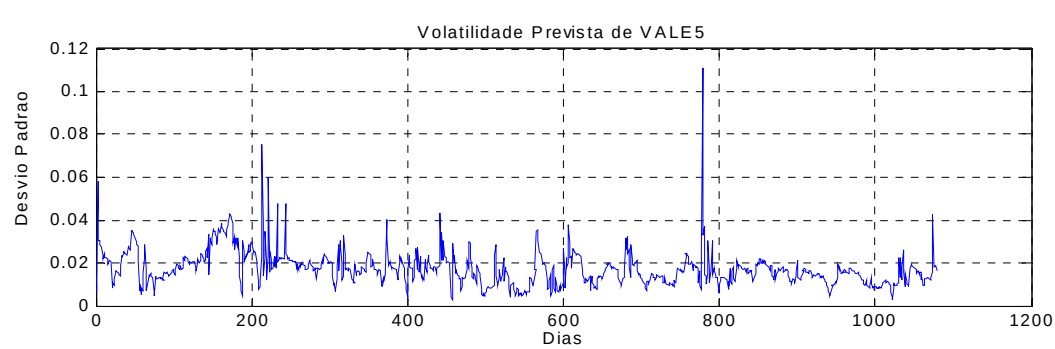

Volatilidade Prevista de SBSP3
Desvio Padrao
0.08
0.06
0.04
0.02
0
0
200
400
600
800
1000
1200
Dias


Volatilidade Prevista de TLPP4
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0
200
400
600
800
1000
1200


Volatilidade Prevista de TNLP4
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0
200
400
600
800
1000
1200


Volatilidade Prevista de TSPP4
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0.12
0
200
400
600
800
1000
1200


Volatilidade Prevista de USIM5
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0
200
400
600
800
1000
1200


Volatilidade Prevista de VALE5
Desvio Padrao
Dias
0
0.02
0.04
0.06
0.08
0.1
0.12
0
200
400
600
800
1000
1200