Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Aplicação de Redes Neurais Artificiais na Sugestão de Investimentos

Lucas Flasch Romani

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcus Vinicius Lamar

Brasília
2017

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Rodrigo Bonifácio de Almeida

Banca examinadora composta por:

Prof. Dr. Marcus Vinicius Lamar (Orientador) — CIC/UnB
Prof. Dr. Marcelo Grandi Mandelli — CIC/UnB
Prof. Msc. Marcos Fagundes Caetano — CIC/UnB

CIP — Catalogação Internacional na Publicação

Romani, Lucas Flasch.

Aplicação de Redes Neurais Artificiais na Sugestão de Investimentos / Lucas Flasch Romani. Brasília : UnB, 2017.
93 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2017.

1. Redes Neurais Artificiais, 2. Python, 3. Investimentos, 4. Perfil de Investidor

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Aplicação de Redes Neurais Artificiais na Sugestão de Investimentos

Lucas Flasch Romani

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcus Vinicius Lamar (Orientador)
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
CIC/UnB

Prof. Msc. Marcos Fagundes Caetano
CIC/UnB

Prof. Dr. Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 16 de fevereiro de 2017

Dedicatória

Dedico este trabalho a todos que direta ou indiretamente colaboraram com minha formação. *Auribus teneo lupum timere neminem.*

Agradecimentos

Agradeço à minha *família*, ao *Prof. Dr. Marcus Vinicius Lamar* e ao *Banco do Brasil S/A*.

Resumo

Este trabalho apresenta um meio de treinar uma Rede Neural Artificial para oferecimento de produtos de investimento que melhorem o rendimento de clientes, porém, mantendo-os adequados ao seu padrão de investimento. Através da utilização dos dados de clientes com a maior rentabilidade em suas carteiras de investimento e alinhados ao seu perfil de investidor, a rede é treinada. Cada aplicação feita pelos melhores clientes se torna a saída da rede neural, a carteia atual de produtos de cada cliente, assim como o perfil escolhido pelo mesmo, se tornam a entrada. Após o treinamento, a validade da rede é verificada para garantir que houve o aumento da rentabilidade na maioria dos casos. Dessa forma, clientes que estejam em situação semelhante, podem melhorar seus investimentos através de consulta à rede. Devido ao grande volume de dados, técnicas para otimizar a performance, como a utilização de placas de gráficas no processamento, foram utilizadas. Também foram observadas técnicas para a compensação do desbalanceamento dos dados, que resultaram na utilização de uma metodologia pouco ortodoxa, mas de grande efetividade, no tratamento desse desbalanceamento. Bibliotecas pré-formatadas para trabalho com redes neurais através da linguagem de programação Python também foram utilizadas, o que diminuiu o tempo de criação das redes. Melhorias futuras incluem evolução do algoritmo, testes com outros métodos de inteligência artificial e previsão de comportamento de mercado.

Palavras-chave: Redes Neurais Artificiais, Python, Investimentos, Perfil de Investidor

Abstract

This paper presents a way of training an Artificial Neural Network to offer investment products that improve the performance of clients, but keeping them adequate to their investment pattern. Through the use of the most profitable customer data in its investment portfolios and in line with its investor profile, the network is trained. Each application made by the best clients becomes the output of the neural network, the current product portfolio of each client, as well as the profile chosen by the client, become the input. After training, the validity of the network is checked to ensure that there has been increased profitability in most cases. In this way, customers in a similar situation can improve their investments by consulting the network. Due to the large volume of data, techniques to optimize performance, such as the use of graphics cards in the processing, were used. Techniques were also observed to compensate for the imbalance of the data, which resulted in the use of an unorthodox methodology, but of great effectiveness, in the treatment of this imbalance. Pre-formatted libraries for working with neural networks through the Python programming language were also used, which reduced the time the networks were created. Future improvements include evolution of the algorithm, testing with other artificial intelligence methods, and prediction of market behavior.

Keywords: LaTeX, scientific method, thesis

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ANBIMA Associação Brasileira das Entidades dos Mercados Financeiro e de Capitais. 11, 12

API Análise de Perfil de Investidor. 8–11, 17, 18, 40, 42, 71

BCB Banco Central do Brasil. 6

BPN Back-Propagation Nets. 25, 30

CDI Certificado de Depósito Interbancário. 16

CMN Conselho Monetário Nacional. 6

CNPC Conselho Nacional de Previdência Complementar. 6

CNSP Conselho Nacional de Seguros Privados. 6

COBOL COmmon Business Oriented Language. 35, 36

CRISP-DM CRoss Industry Standard Process for Data Mining. 33

CVM Comissão de Valores Mobiliários. 1, 2, 6–9, 11, 12

IA Inteligência Artificial. 24

IBM International Business Machines. 35, 36

IBOVESPA Índice BOVESPA. 16, 38

ICVM Instrução CVM. 7–10

IPCA Índice Nacional de Preços ao Consumidor Amplo. 16

JCL Job Control Language. 35

KDD Knowledge-Discovery in Databases. 33

MDS Master Data Services. 33

MLP Multi-Layer Parceptron. 25, 30

PREVIC Superintendência Nacional de Previdência Complementar. 6

RNA Redes Neurais Artificiais. 24–29, 39, 45, 55, 72, 73

SFN Sistema Financeiro Nacional. 5–7

SGD Stochastic Gradient Descent. 49, 55

SUSEP Superintendência de Seguros Privados. 6

Capítulo 1

Introdução

O mercado de investimentos é parte do sistema financeiro. Nele um agente superavitário empresta seu capital a um agente deficitário com o objetivo de compensação financeira. Este empréstimo ocorre através de uma instituição financeira, devidamente regularizada no sistema financeiro. Dessa forma, analogamente para o mercado de investimentos, um investidor empresta seu capital a outro agente com o objetivo de obter lucro. O investimento é visto como um produto, ofertado pela instituição financeira que intermediará a operação.

Os investimentos sempre foram um desafio para o setor financeiro no que tange à oferta de produtos. Com uma grande variedade, cada um com vantagens e desvantagens, comparar todos os produtos e sugerir uma combinação que seja a melhor para o investidor é uma tarefa complexa. A complexidade envolvida se deve por diversos fatores, dentre eles a grande quantidade de produtos, a análise das condições econômicas, a análise das preferências de cada investidor, além das questões legais envolvidas.

Dentre as instituições que regulam nacionalmente os investimentos, temos a Comissão de Valores Mobiliários (CVM). Conforme regulamentação da CVM de 2013, as instituições financeiras do Brasil tiveram de implementar mecanismos que garantissem a conformidade entre os interesses do cliente investidor, seus perfis de investimento e as aplicações que efetuassem. Estes mecanismos aumentaram a complexidade na sugestão de produtos de investimento. Agora, uma sugestão de aplicação deve melhorar a rentabilidade dos investimentos do cliente, estar em conformidade com os interesses do mesmo e com os dispositivos dessa regulamentação.

A principal característica da regulamentação é normatizar o que muitas instituições já faziam de um modo menos formal. Tais instituições procuravam: alinhar o comportamento do cliente no que tange a sua capacidade de assumir riscos, seu conhecimento do cenário econômico e de investimentos, sua situação financeira e seus interesses em curto, médio e longo prazo. O resultados da análise dessas características é conhecido como

perfil de investidor. Com o resultado da identificação do perfil do investidor é possível saber se os investimentos do mesmo estão enquadrados com esse perfil.

1.1 Problema

Dessa forma, este trabalho parte do seguinte problema de pesquisa: dado um conjunto de produtos de investimento, uma carteira de produtos que um cliente possui e uma regra de enquadramento que limite o tamanho das aplicações por risco de produto, encontrar, de maneira automática, um bom produto para ser sugerido como a próxima aplicação, aumentando os lucros da carteira sem afetar a regra de enquadramento.

1.2 Justificativa

A importância deste trabalho se reflete em melhorar as aplicações de investidores sem comprometer os risco do investimento sob o aspecto legal. Atualmente muitas instituições fazem a oferta de produtos de investimento conforme interesse negocial ou por simples adequação ao perfil. Dessa forma, raramente é ofertado um produto ao cliente que possui rentabilidade boa e adequação ao perfil.

Como analista da área de tecnologia de um banco público, atuando na análise de sistemas de investimentos, identifiquei que esse problema é comum entre instituições financeiras e tem dificultado a relação entre cliente e banco, dado que as restrições impostas pela CVM não permitem a oferta de qualquer produto de investimento, ainda que melhore a rentabilidade do portfólio de aplicações do mesmo.

1.3 Hipótese

É possível encontrar uma boa aplicação para um determinado cliente através do uso de redes neurais, baseando-se na experiência de aplicações de clientes experientes e/ou bem sucedidos.

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo geral desta pesquisa é propor e validar o uso de uma nova metodologia baseada em Redes Neurais na busca automática de boas ofertas de produtos de investimento personalizadas.

1.4.2 Objetivos Específicos

Para atingir o objetivo geral temos os seguintes objetivos específicos.

- Selecionar a partir dos bancos de dados disponíveis, dados reais de produtos e aplicações de clientes;
- Estudar a teoria na qual são baseadas as Redes Neurais;
- Propor e testar estruturas de Redes Neurais adequadas à solução do problema de pesquisa;
- Avaliar a eficiência da soluções encontradas na busca por investimentos mais rentáveis.

1.5 Organização

Este trabalho encontra-se assim organizado:

Capítulo 2 - Revisão Teórica. Neste capítulo são abordados os conceitos fundamentais para a interpretação do trabalho: Análise do Perfil do Investidor, Análise de Conceitos de Rentabilidade, Técnicas de Inteligência Artificial e Mineração de Dados.

Capítulo 3 - Sistema Proposto. Este capítulo descreve a obtenção e tratamento dos dados, assim como os experimentos realizados e os resultados obtidos com a aplicação das técnicas mencionadas no capítulo anterior.

Capítulo 4 - Resultados Obtidos. Neste capítulo são comparados os resultados provenientes das técnicas utilizadas e é feita a análise de tais resultados.

Capítulo 5 - Conclusão. Este capítulo sumariza a metodologia, os resultados e a análise, apresentado as conclusões obtidas.

Capítulo 2

Revisão Teórica

Este capítulo apresenta os conceitos básicos envolvidos neste trabalho, como o mercado financeiro, o Sistema Financeiro Nacional, a análise do perfil do investidor e da carteira do cliente. Apresenta ainda as técnicas de otimização, em especial as baseadas em conceitos de inteligência artificial, tais como as redes neurais artificiais.

2.1 Mercado Financeiro

O Mercado Financeiro é o meio onde se realizam as transferências de recursos entre os agentes financeiros de forma direta ou indireta. Ele é a forma como captações viram empréstimos ou investimentos[2].

Os investimentos ou ativos financeiros, possuem um risco, uma rentabilidade e uma liquidez associados. Existe um relação entre risco e rentabilidade para os investimentos[2]. Ela normalmente é proporcional, embora alguns fatores externos como a política, podem distorcer tal relação. A Figura 2.1 exemplifica tal relação em um mercado saudável (sem interferência externa).

Os conceitos de risco, rentabilidade e liquidez podem ser definidos como:

Risco: é a variação da rentabilidade de um investimento. Esta variação influencia na probabilidade de se obter um rendimento esperado. Quando há muita variação em sua rentabilidade, um investimento pode não trazer a rentabilidade esperada ou até trazer prejuízo.

Rentabilidade: é o retorno do capital investido, ou seja, se o investimento fosse retirado, subtraindo-se o capital após a retirada do capital inicialmente investido, teríamos a rentabilidade.

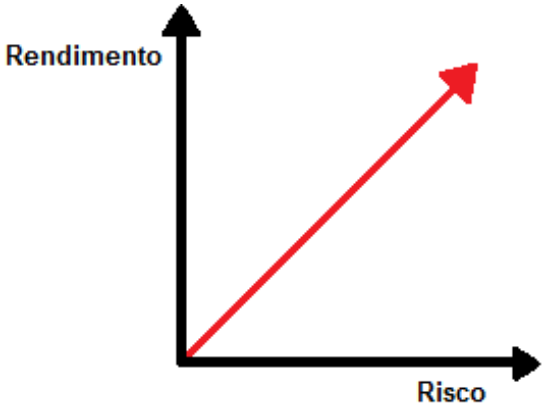

Figura 2.1: Risco por Rendimento em um mercado saudável .

Liquidez: é a facilidade com que um investidor pode converter seu investimento novamente em capital. Pode ter ligação, tanto com a natureza do investimento quanto com seu valor para o mercado.

Uma forma para se minimizar o risco é diversificar os investimentos. A diversificação não pode ser feita de qualquer forma, deve-se analisar a natureza de cada um e sua relação com os outros. Normalmente avalia-se a variação de um ativo em relação ao outro a fim de detectar o grau de proximidade entre eles[3].

Os ativos financeiros (comercializados no Mercado Financeiro) estão divididos em 3 grupos:

Renda Fixa: títulos de baixo risco cuja rentabilidade pode ser conhecida na sua contratação, tipicamente composta por produtos como títulos do governo, poupança, certificado de depósito bancário, debêntures, entre outros;

Ações: títulos de empresas de capital aberto comercializada em Bolsas de Valores. Por ser um mercado de alta volatilidade é considerado de elevado risco;

Derivativos: cujo preço deriva de outro ativo, comumente atrelado a uma operação real. Por exemplo: um agricultor que queira assegurar que quando sua plantação estiver prestes as ser exportada o preço dela se mantenha nos patamares atuais de câmbio, dessa forma cria-se um derivativo para segurar o preço de câmbio. Também pode ser entendido como uma aposta.

2.2 Sistema Financeiro Nacional

O Sistema Financeiro Nacional (SFN) abrange todas as operações envolvendo a moeda nacional. Títulos, empréstimos, investimentos, são parte desse sistema, bem como as

instituições que atuam nesses ramos. São divididos em três setores: o Setor Normativo, o Setor Supervisor e o Setor dos Operadores. A Figura 2.2 representa o SFN através de um diagrama.

	Moeda, Crédito, Capitais e Câmbio		Seguros privados	Previdência fechada
Órgãos Normativos	CMN		CNSP	CNPC
Supervisores	BCB	CVM	SUSEP	PREVIC
Operadores	Bancos	Bolsa de Valores	Seguradoras e Resseguradoras	Entidades Fechadas de Previdência Complementar
	Caixas Econômicas	Bolsa de Mercadorias e Futuros	Entidades Abertas de Previdência	
	Cooperativas de Crédito		Sociedades de Capitalização	
	Instituições de Pagamento			
	Administradoras de Consórcio			
	Corretoras e Distribuidoras			
	Demais Instituições não Bancárias			

Figura 2.2: Diagrama Representativo do SFN.

As instituições normativas, que compõem o Setor Normativo, do SFN são o Conselho Monetário Nacional (CMN), o Conselho Nacional de Seguros Privados (CNSP) e o Conselho Nacional de Previdência Complementar (CNPC). Tais instituições se encarregam de legislar sobre o sistema, além de traçar as diretrizes do mesmo com o objetivo de realizar as metas do governo. O CMN é responsável pela normatização dos assuntos de moeda, crédito, capitais e câmbio. O CNSP rege os assuntos de seguros e resseguros no SFN, enquanto o CNPC rege as previdências fechadas.

Exitem instituições supervisoras desse sistema que são responsáveis pela verificação do cumprimento das normas impostas pelo Setor Normativo. Tais instituições são: Banco Central do Brasil (BCB), Comissão de Valores Mobiliários (CVM), Superintendência de Seguros Privados (SUSEP) e Superintendência Nacional de Previdência Complementar (PREVIC). Embora o papel principal dessas instituições seja supervisionar, algumas delas também normatizam certos aspectos dos mercados que atuam. É papel do BCB supervisionar os bancos, instituições de pagamento, corretoras, entre outros, que efetuem crédito, câmbio e movimentação de moeda. A CVM é responsável pela supervisão do mercado de capitais, mercadorias e futuros. A SUSEP é responsável pelo mercado de seguros, resseguros e previdência aberta, enquanto a PREVIC é responsável pelas previdências fechadas (complementares) e fundos de pensão.

Os operadores são as instituições públicas ou privadas que atuam no SFN através comercialização de produtos e serviços. Tais instituições tem por finalidade a obtenção de lucro através dessa prática.

2.3 Análise do Perfil do Investidor

Para regulamentar o mercado de investimentos brasileiro, a CVM (Comissão de Valores Mobiliários) publicou uma série de instruções de forma a proteger as instituições e os consumidores desse mercado. Essas instruções têm o objetivo de prevenir que ativos sejam comprados sem que a correta informação sobre o mesmo seja antes passada ao cliente, mas também isenta a instituição de responsabilidades caso implemente a norma.

2.3.1 Legislação

Conforme Instrução CVM (ICVM) nº 539 de 13 de novembro de 2013 [4] em:

- Seu artigo 1º, as instituições integrantes dos sistemas de distribuição de valores mobiliários ou corretores que devem verificar a adequação ao perfil de investidor antes de realizar operações, consultoria ou qualquer outro tipo de serviço prestado;
- Seu artigo 2º, parágrafos 1º, 2º e 3º estão apresentadas as condições para a oferta de um produto de investimento do cliente, garantindo que o produto esteja alinhado aos objetivos e preferências do mesmo e que ele conheça as características do produto;
- Seu artigo 3º descreve que devem existir perfis onde os clientes devem ser classificados, tais perfis derivam do comportamento de cada cliente;
- Seu artigo 4º descreve como devem ser classificados os produtos, ele também especifica que cada produto deve ter um risco associado relativo aos ativos que o compõem, prazos, garantias e perfil da emissora;
- Seu 5º artigo impede que o cliente possa aplicar caso não sejam respeitados os artigos anteriores;
- Seu 6ª artigo define as diretrizes para caso o cliente opte por não participar do processo de análise do perfil ou o produto que está prestes a contratar não se enquadrar em seu perfil;
- Seu artigo 7º descreve como devem ser os procedimentos e políticas internas assim como os responsáveis pelas instituições em relação ao cumprimento deste processo;
- Seu artigo 8º fica definido o prazo para vencimento da análise do perfil e da avaliação de risco dos produtos;

- Seu 9º artigo delimita quais públicos estão dispensados do API;
- Seu artigo 10º os prazos de retenção e meio de armazenamento dos documentos do processo;
- Seu artigo 11º abre a possibilidade de que as instituições implementem outros mecanismos para assegurar o processo;
- Seu artigo 12º define como infração grave a não execução da referida instrução, conforme previsto na Lei nº 6.385, de 7 de dezembro de 1976 (Lei de criação da CVM);
- Seu 13º artigo o prazo que as instituições tinham para implementar a instrução. Originalmente em 5 de janeiro de 2015 todas as instituições deveriam implementar o proposto nos artigos anteriores.

A Instrução CVM (ICVM) 554 de 17 de dezembro de 2014[5] altera algumas considerações feitas em diversas instruções incluindo a 539. Dentre as alterações da Instrução CVM (ICVM) 539 estão o artigo 9º e artigo 13º. O primeiro inclui os investidores qualificados e profissionais como dispensados do API, seguindo o objetivo original do artigo que era a descrição desse público. O segundo posterga a data de implantação para 1º de julho de 2015.

Segundo a Instrução CVM (ICVM) 554 é considerado investidor profissional:

- Instituições autorizadas a funcionar pelo Banco Central do Brasil;
- Companhias seguradoras e sociedades de capitalização;
- Entidades abertas e fechadas de previdência complementar;
- Investidores que possuam mais de dez milhões de reais investidos e tenham assinado o termo de investido profissional;
- Fundos de investimento;
- Clubes de investimento vinculados a um administrador autorizado pela CVM;
- Agentes, administradores, analistas e consultores autorizados pela CVM;
- Investidores não residentes no país.

Já o investidor qualificado deve possuir as seguintes características:

- Investidores profissionais;
- Investidores que possuam mais de um milhões de reais investidos e tenham assinado o termo de investido qualificado;

- agentes, administradores, analistas e consultores que tenham sido aprovadas em exames de qualificação técnica ou possuam certificações aprovadas pela CVM em relação aos seus recursos próprios;
- clubes de investimento que tenham a carteira gerida por ao menso um cotistas, que seja investidores qualificados.

Também são considerados investidores qualificados ou profissionais os regimes próprios de previdência social (União, Estados, Distrito Federal e Municípios) desde que reconhecidos como tais conforme regulamentação específica do Ministério da Previdência Social.

A Instrução CVM (ICVM) 555 de 17 de dezembro de 2014[6] discorre sobre as características de um fundo de investimento e as restrições impostas aos administradores, sejam pessoas físicas ou jurídicas, para a legalização do mesmo. Dentre as características necessárias para a criação do fundo de investimento estão:

- Descrição dos termos utilizados como cota, fechamento, cessão fiduciária, entre outros;
- Regras para a emissão e cálculo doo valor das cotas de um fundo;
- Regras sobre o modo de comercialização das cotas, que deve ser feito exclusivamente por meio eletrônico;
- Regras de suspensão do recebimento de novas aplicações para fundos abertos;
- Cadastro das características extras na CVM caso seja um fundo fechado;
- Discriminação das características do fundo para todo cotista que deseje entrar no mesmo, atestando por meio da formalização da assinatura do termo de adesão;
- Subscrição por conta e ordem, garantindo que as cotas de uma conta ou ordem dentro de um fundo não se comuniquem com as outras;
- O resgate de cotas;
- Taxas de administração, ingresso, performance e saída do fundo.

Outro ponto importante que esta ICVM apresenta é a forma de cálculo da rentabilidade do fundo. Ela especifica que o valor da cota atual menos o valor da cota base subtraídos da previsão da taxa de performance, taxa de administração e taxa de saída do fundo é que devem compor a rentabilidade do mesmo.

Em relação a API a ICVM 555 discrimina alguns fundos que possuem característica diferenciada. Tais fundos são ditos simples, pois possuem características como ser de Renda Fixa, e possuir a maior parte do seu patrimônio atrelado à títulos de renda fixa ou

títulos do governo. Por ser diferenciados e de risco baixo, estão dispensados do processo API, não precisando que a ICVM 539 seja aplicada.

A Instrução CVM (ICVM) 564 de 11 de junho de 2015[7] altera o prazo para implantação da Instrução CVM (ICVM) 555 e altera da data de implantação dos controles para o artigo 9º referente a definição de investidores qualificados e profissionais na Instrução CVM (ICVM) 554.

A Instrução CVM (ICVM) 566 de 31 de junho de 2015 [8] define que investidores profissionais não precisam de intermediários quando estiverem adquirindo notas promissórias. A oferta pública de notas promissórias incorpora alguns critérios, mas as mesmas continuam sendo ofertadas no mercado de balcão, o que não acarreta na aplicação da Análise de Perfil de Investidor na oferta da mesma. Portanto, para estes produtos não é necessário a apresentação de termos e notificações da natureza do produto ou da situação do cliente.

2.3.2 Dispensa de Análise de Perfil do Investidor

Alguns investidores são dispensados da análise do perfil de investidor devido ao nível de proteção que necessitam poder variar significativamente em detrimento das atividades que executam. É o caso de clientes Pessoa Jurídica e que sejam investidores qualificados ou profissionais (investidores que possuem mais de um milhão e dez milhões de reais aplicados respectivamente[4]).

Outros investidores dispensados são as Pessoas Jurídicas de direito público, ou seja, aquelas criadas por lei, (países, estados e municípios), além de outros entes que formam a chamada Administração Pública (embaixadas, sociedades civis de previdência complementar de órgãos públicos, entre outros).

Também são dispensados clientes que tenham suas carteiras de investimento administradas por corretores de investimento ou que tenham se negado a participar da análise do perfil de investidor, dado que tenham sido avisados dos riscos dessa decisão.

2.3.3 API em um Banco Público

Mesmo que bem delimitada, a norma apresenta alguns pontos que são passíveis de múltiplas interpretações. Assim, cada agente criou sua solução para resolução do problema. Este trabalho é realizado com os dados de um banco público brasileiro conhecido. Portanto, talvez haja necessidade da adaptação dos tipos de dados para outras instituições, mas a metodologia aplicada no processo de obtenção e análise desses dados não difere.

2.3.4 Controles Internos

Controlar significa influenciar de modo que algo tenha o comportamento que garanta o resultado desejado. Os sistemas de controle são elaborados e analisados a partir da teoria do controle que é uma subárea da matemática. [9]

Em essência, a teoria do controle consiste em adicionar um agente corretivo que balize o comportamento dos outros agentes no fluxo do processo, de modo a restringir tal processo conforme os padrões determinados [10].

Esta busca pelos controles internos e externos de modo que um segmento da sociedade se auto-regule também é conhecido como *Compliance*.

Segundo SILVA, para uma instituição pública "não há risco financeiro ou patrimonial para os administradores porque o investimento vem do povo"[9]. Porém um banco público diverge dessa definição, uma vez que deve responder aos agentes reguladores e deve possuir retorno financeiro ao patrocinador. Por tanto as metodologias de *Compliance* se adequam a sua realidade.

Conforme regulamentação, os bancos públicos "não podem recomendar produtos, realizar operações ou prestar serviços sem que verifiquem sua adequação ao perfil do cliente"[4], porém também devem oferecer o melhor produto que maximize a rentabilidade da carteira do mesmo.

2.3.5 Regulamentação ANBIMA sobre API

Antes da promulgação da legislação CVM, houve regulamentações sobre tema API em caráter reduzido publicadas pela Associação Brasileira das Entidades dos Mercados Financeiro e de Capitais (ANBIMA). Essas regulamentações se destinavam a grandes investidores e possuíam características parecidas com o que se propôs na legislação posterior.

Dentre as características parecidas estavam a necessidade de aplicação de questionário para determinação do perfil de investidor e a análise da carteira para orientação dos investimentos.

Outra característica semelhante era a definição dos perfis de investidor. Conforme as respostas do questionário, era atribuído um perfil de investidor dentre Conservador, Moderado, Arrojado e Agressivo, cada qual com sua definição de nível de risco aceitável.

2.3.6 Perfil do Cliente e Mercado do Cliente

Dado a pré-existência de uma regulamentação sobre o tema, a CVM deixou em aberto que as instituições definissem os próprios perfis, desde que estivessem atrelados ao risco que o cliente aceita correr em suas aplicações. O artigo 3º define que as instituições "devem avaliar e classificar o cliente em categorias de perfil de risco previamente estabelecidas."[4].

Algumas instituições adotaram o padrão ANBIMA, com 4 perfis, outras adotaram o padrão de mercado, com 3 perfis: Conservador, Moderado e Arrojado [11].

A instituição cujas bases estão sendo analisadas adotou o seguinte padrão em relação ao mercado que o cliente se encontra:

- Para os clientes Pessoa Física ou Jurídica que pertenciam ao mercado de altíssima renda, alvo das publicações da ANBIMA, antes das instruções CVM, permaneceu o padrão com 4 perfis.
- Para os demais clientes fica o padrão de mercado de 3 perfis.

Os perfis do investidor podem ser definidos como Agressivo, Arrojado, Moderado e Conservador e seus significados são:

Investidor Agressivo "está disposto a correr riscos para conseguir a máxima rentabilidade, está propenso a investir a maior parte de seu dinheiro em aplicações que apresentam grande oscilação e a destinar uma parcela mínima para aplicações mais seguras que preservem o capital investido"[12].

Investidor Arrojado Possui equivalência com um investidor agressivo, porém o nível de risco aceitável em seus investimentos é um pouco menor.

Investidor Moderado "está disposto a correr um risco um pouco mais elevado para obter uma rentabilidade maior, está propenso a aplicar uma parcela significativa do dinheiro em investimentos que oscilam muito, destinando o restante para aplicações mais seguras"[11].

Investidor Conservador "é aquele que não está disposto a correr riscos e a aplicar dinheiro em investimentos com grande oscilação, ou ainda, sua atual situação não permite investimentos que envolvam altos riscos e que possam comprometer o capital investido"[11].

Conforme exposto pela legislação, um questionário deve ser aplicado com o objetivo de mensurar qual perfil cada cliente se encaixa. Os critérios que após a análise das respostas devem ser considerados para cálculo do perfil são:

Idade. Uma prática de mercado é considerar que pessoas mais velhas queiram manter seu patrimônio, não necessariamente adquirir mais. Em compensação pessoas jovens podem assumir mais riscos por estarem no início da vida, tendo tempo suficiente para se recuperar.

Grau de Escolaridade. Não necessariamente há uma relação entre grau de escolaridade e capacidade de assumir riscos, porém, entende-se que pessoas com mais escolaridade entendam melhor o mercado e façam investimentos mais conscientes. Portanto, podendo assumir mais riscos.

Valor de Aplicação. A quantidade para aplicação define se é interessante diluir em várias aplicações ou manter em uma mais segura. Isso se deve ao fato de que no aparecimento de uma emergência, para que possui pouco volume aplicado, deve ter garantia do resgate da aplicação e da manutenção do valor aplicado. Algo que não pode ser feito em aplicações mais agressivas, onde se tende a priorizar o longo prazo.

Conhecimento e Experiência em Investimentos. São necessários para realizar qualquer investimento. Ao fazer um investimento, deve-se estar atento ao nível de risco desse investimento, rentabilidade atual e possível rentabilidade futura. Há investimentos de variadas características, o que implica em investimentos de menor risco e maior rentabilidade, assim como rendimentos de menor rentabilidade e maior risco. Nessa perspectiva, investir em maior risco seria uma contradição, mesmo para um perfil agressivo.

Prazo para Resgate. Conforme mencionado em "Valor de Aplicação", o prazo para resgate deve ser considerado, não só quando o prazo é conhecido, mas também quando o investimento puder ser resgatado inesperadamente diante de eventual emergência. O prazo define a natureza da aplicação.

Objetivo da Aplicação. Um cliente cujas aplicações têm o objetivo de manter o valor investido ou apenas acúmulo de capital para realização de outro objetivo como viajem, casa, entre outros, então as aplicações destinadas a esse cliente deve ser diferentes de outros clientes que estejam mais interessados, por exemplo, em rentabilidade.

Resiliência a Perdas. Se o investimento pode gerar perdas, normalmente isso se dá a curto prazo, pode gerar um desconforto em investidores não acostumados a essa realidade. Esses investidores, no âmbito de reduzir as perdas, tendem a resgatar o valor antes da hora, o que faz com que percam o momento onde o ativo volta a crescer. Estar preparado para perdas é essencial na avaliação do perfil do cliente.

Nível de Informação sobre o Mercado. Se sabe-se pouco sobre o mercado que se procura investir, deve-se procurar outro mercado ou procurar orientação adequada para tal. Não saber a situação econômica, política e social da atualidade, assim como a situação do mercado, suas perspectivas e necessidades impacta diretamente no investimento que se pode fazer.

O questionário respondido pelos investidores tem o objetivo de sanar as dúvidas com relação aos critérios citados. Cada pergunta do questionário possui um peso dependendo das alternativas que foram escolhidas, onde, ao final, são somados tais pesos. A soma se enquadrará na faixa representada por cada perfil. Se for uma soma de valor baixo, significa que o investidor não está interessado em grandes captações ou é muito vulnerável ao risco, sendo enquadrado como os perfis conservador. Conforme o valor da soma cresce significa que para mais perguntas o investidor está disponível a mais risco, portanto pode ir sendo classificado em perfis mais agressivos. Pode-se ver a relação entre risco e perfil na Figura 2.3.

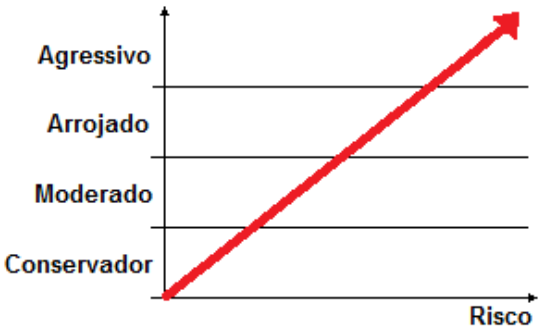

Figura 2.3: Modelo de Risco X Perfil.

2.3.7 Risco do Produto

Conforme o artigo 2º parágrafo 5º as instituições "devem considerar os custos diretos e indiretos associados aos produtos, serviços ou operações, abstendo-se de recomendar aqueles que, isoladamente ou em conjunto, impliquem custos excessivos e inadequados ao perfil do cliente"[4]. O que significa oferecer apenas aquilo que o cliente tenha conhecimento e notificar quando o cliente estiver prestes a ultrapassar os limites de seu perfil.

No artigo 4º a instrução descreve o processo para análise do risco de um produto. Para realizar a classificação do risco, as instituições "devem analisar e classificar as categorias de produtos com que atuem, identificando as características que possam afetar sua adequação ao perfil do cliente"[4]. Ou seja, devem criar classes de risco que alertem quando um produto de risco mais alto estiver sendo adquirido por um cliente cujo perfil não permita a aplicação nessa classe ou o máximo delimitado para a aplicação na classe de risco tiver sido ultrapassado.

Porém, para delimitar se um cliente deve ou não ser inquerido dos alertas sobre o risco do produto, conforme os incisos 1 e 2 do 3º parágrafo do artigo 2º, também deve-se avaliar a natureza do investidor, seus conhecimentos do mercado e seu padrão de investimentos.

Portanto, segmentar o cliente em mercados é de crucial importância, não só a análise simples do risco do produto ou o perfil do cliente. O que se subentende é que o risco do produto é relativo ao mercado em que o cliente está. Se ele estiver em um mercado habituado nas finanças e investimentos, um produto que seria de risco alto para outros mercados, pode ser um risco mediano ou até baixo para o mercado desse cliente.

2.4 Análise da Carteira do Cliente

A análise da carteira do cliente é de relevante importância para o processo, uma vez que para identificar a melhor oferta, deve-se filtrar os dados dos clientes que tenham uma rentabilidade boa, mas mantenham-se enquadrados no risco.

Na análise são considerados quatro processos: rentabilidade dos ativos, rentabilidade da carteira, índices e risco da carteira.

2.4.1 Análise de Rentabilidade dos Ativos

É complexa a tarefa do cálculo de rentabilidade para ativos de natureza diferente. Calcular a rentabilidade de um produto como Tesouro Direto é diferente de calcular a rentabilidade de um produto como Fundo de Investimento, que por sua vez é diferente do cálculo de ações. Cada produto possui suas características, taxas, prazo de liquidação, entre outros. Porém, quando tiverem se convertido em dinheiro novamente, os investimentos podem finalmente ter seu índice real de rentabilidade calculado. Calculando a diferença do que foi aplicado com o que foi investido pode-se ter o índice real de rentabilidade (a dedução dos impostos e a inflação no período não são considerados pois afetam a maioria dos produtos por igual).

A rentabilidade dos ativos é composta pela média das diferenças de preços do ativo supondo que o cliente fosse resgatá-lo no mesmo dia, para os últimos trinta dias. Ou seja, caso o ativo fosse resgatado, naquele dia, já deduzido das taxas subtrai-se o que se tinha do dia anterior, gerando a rentabilidade daquele dia específico. A média das rentabilidades dos últimos 30 dias é que será considerada como rentabilidade do produto.

Existem outras formas de cálculo para rentabilidade, cada qual com suas vantagens e desvantagens, por exemplo, se fossem considerados os últimos trezentos e sessenta e cinco dias (último ano), mais ações se destacariam frente a Fundos de Investimento, uma vez que o ideal para ações é o investimento em longo prazo. Porém, como estamos analisando as carteiras dos clientes com maior rentabilidade, há uma tendencia de que tais clientes já tenham considerado tal fator e tenham diluído seus investimentos entre os melhores produtos. O importante é detectar essa tendência na ora de ofertar algo.

2.4.2 Análise de Rentabilidade da Carteira

A rentabilidade da carteira do cliente é calculada através da média ponderada dos retornos dos ativos que a compõem [3]. Pode ser vista na Equação 2.1

$$R = \sum_{i=1}^{N} RA_i.X_i, \tag{2.1}$$

onde R é a rentabilidade total da carteira, N é o total de produtos pertencentes à carteira, RA_i é a rentabilidade do ativo i pertencente à carteira do cliente e X_i é o peso do ativo na carteira.

2.4.3 Índices e Benchmark

Um índice é um valor numérico referente ao cálculo de um ou conjunto de elementos, cujo valor varia com o tempo. Índices são importantes para o investidor, pois servem como referência antes da tomada de decisão dentro do mercado (aqui mercado é utilizado no conceito de conjunto de pessoas atuando no mesmo ramo econômico) em que o ativo que negocia atua, assim como fora dele, nacionalmente como internacionalmente[2].

É comum um índice ser composto por um conjunto de algo, como o índice de inflação IPCA, que é composto pela variação do preço do conjunto de produtos mais comuns ao consumidor brasileiro. Há também índices para o mercado de ações como o IBOVESPA, que medem a variação dos preços das ações das empresas mais significativas para o mercado nacional.

Esses índices orientam os especialistas de cada mercado sobre como está sua situação referente a algo conhecido.

Benchmark em Inglês significa ponto de referência. Pode ser um índice, indicador de desempenho, valor, entre outros [13]. Um *Benchmark* importante para o mercado nacional é o Certificado de Depósito Interbancário (CDI), dado por títulos emitidos entre instituições financeiras para sanarem o caixa ao final do dia. Este é um índice importante para a maioria dos investimentos que tem sua rentabilidade atrela ao percentual do mesmo.

Como a natureza dos investimentos é diferente para cada segmento de investimento, adotar um *Benchmark* comum a todos poderia prejudicar determinadas modalidades de investimento. Dessa forma, como *Benchmark* utilizado para o projeto foram selecionados os mil clientes com melhor rentabilidade por mercado (aqui mercado é utilizado no conceito original do trabalho). Isso garante que todos os mercados tenham algum cliente para treino.

2.4.4 Análise de Risco da Carteira

O risco atribuído à carteira de investimentos no processo API é diferente do risco da carteira atribuído na Moderna Teoria de Carteiras de Markowitz[3]. Para o processo API o produto pertence a uma classe de risco. Essa classe é está descrita na Tabela 2.1 e está associada à variação da rentabilidade de um produto de investimento, mas não necessariamente dita como deve ser classificado. Outros fatores podem influenciar como o mercado que o cliente está inserido. Dado que a regulação do API define que deve ser analisado o comportamento do cliente antes de oferecer um produto e que cada produto deve possuir um classe, então é considerado o risco de cada produto por tipo de mercado (o mercado também leva em consideração a familiaridade em investimentos do cliente), antes de oferecê-lo. Assim um produto de alto risco para um mercado menos instruído em investimentos pode ser um produto de médio ou até baixo risco para mercados mais instruídos, mesmo que o cálculo da variância seja o mesmo.

Tabela 2.1: Classes de Risco.

Classe	**Definição Risco**
1	Muito Baixo
2	Baixo
3	Médio
4	Alto
5	Muito Alto

Para Markovitz a busca por uma carteira saudável estaria em encontrar aqueles produtos cuja covariância entre eles fosse a menor possível, de modo a que sempre que se perdesse em um produto se ganharia em outro, diminuindo o risco total da carteira. Entende-se por covariância a média ponderada das diferenças entre os desvios de um produto para outro. Por exemplo na Tabela 2.2 podemos ver as rentabilidades dos produtos A, B e C por instante de tempo e o desvio associado às rentabilidades.

Tabela 2.2: Cálculo do Desvio.

Instante	**Rent. A**	**Rent. B**	**Rent. C**	**Desvio A**	**Desvio B**	**Desvio C**
1	10	4	6	-2	-1	4
2	12	6	0	0	1	-2
3	16	5	0	4	0	-2

Conforme os dados da Tabela 2.2 podemos criar a tabela de covariância entre um produto e outro conforme exposto na Tabela 2.3. Nela também pode-se ver as médias das covariâncias.

As médias das covariâncias são a covariância entre um produto e outro, que também pode ser dada pela Equação 2.2 entre um produto A e B:

Tabela 2.3: Cálculo da Covariância.

Cov. A e B	**Cov. A e C**	**Cov B e C**
$-2 \times (-1) = 2$	$-2 \times 4 = -8$	$-1 \times 4 = -4$
$0 \times 1 = 0$	$0 \times -2 = 0$	$1 \times -2 = -2$
$4 \times 0 = 0$	$4 \times -2 = -8$	$0 \times -2 = 0$
$2/3$	$-16/3$	-2

$$\sigma_{AB} = \frac{\sum_{i=1}^{N} (R_{Ai} - R_{\overline{A}})(R_{Bi} - R_{\overline{B}})}{N}, \tag{2.2}$$

onde N é o total de rentabilidades coletadas, $R_{\overline{A}}$ é a média das rentabilidades de A, $R_{\overline{B}}$ é a média das rentabilidades de B, R_{Ai} é a rentabilidade de A no instante i (entende-se por instante o momento em que os preços foram coletados) e R_{Bi} é a rentabilidade de B no instante i.

A relação entre o produto A e B é positiva, portanto A está intimamente ligado a B. Já a relação entre o produto A e C é negativa, ou seja, possuem um baixa correlação entre sí, de modo que compor uma carteira com A e B é pior do que com A e C em termos de risco. Esse é o princípio da administração de riscos: diminuir todo risco pela diversificação em ativos opostos em termos de rentabilidade [3].

Também há um tipo de risco não diversificável. Ele pode ter origem política ou econômica, por exemplo: guerras, crises internacionais, entre outros. Esse tipo de risco também pode ser administrado, porém não será abordado neste estudo.

O risco que avalia qual é a classe de risco de um ativo dentro de um mercado provém da análise da covariância entre esses produtos, porém, após a avaliação, o que é considerado no API é apenas a classe de risco atribuída ao produto conforme a regra de enquadramento.

Enquadramento

Estar enquadrado no API significa que os produtos do cliente estão na proporção certa para cada classe de risco, de modo que exime a instituição financeira de comunicá-lo caso não estivesse de acordo com essas proporções. As proporções são definidas de modo a limitar o máximo que um cliente pode correr de risco e seguem o formato na Tabela 2.4

Para verificar se um investidor está enquadrado, deve-se verificar se ele está enquadrado em cada classe. Se estiver desenquadrado em alguma classe, então não estará enquadrado. O enquadramento na classe segue o seguinte comportamento:

1. Verifica-se se o somatório dos produtos que o cliente possui naquela classe dividido pelo total de investimentos está acima do valor mínimo da classe.

Tabela 2.4: Percentual da Classe de Risco.

Mercado	**Classe Risco**	**Perc. Min.**	**Perc. Máx.**
1	1	10%	100%
1	2	0%	100%
1	3	0%	70%
1	4	0%	40%
1	5	0%	20%
2	1	20%	100%
2	2	0%	100%
2	3	0%	50%
...	...	...	...

2. Verifica-se se o somatório dos produtos do cliente naquela classe e em todas as classes de maior risco dividido pelo total de investimentos estão baixo do máximo da classe.

Se passar nesses dois itens de verificação, então estará enquadrado para a classe que está sendo avaliada.

2.5 Algoritmos de Otimização

Muitos problemas cotidianos e científicos podem ser formulados como problemas de busca e otimização, ou seja, existem diversos fatores que limitam os máximos que um problema pode atingir. Tais fatores restringem as características do problema, mesmo assim, conforme os dados mudam, o problema pode atingir infinitas soluções possíveis.

2.5.1 Conceitos Básicos

Algoritmos de otimização são onde "o objetivo é encontrar a melhor combinação dos fatores, ou seja, a combinação de fatores que proporcione o melhor desempenho possível para o sistema em questão. Em termos técnicos, o conjunto de todas as combinações possíveis para os fatores constitui o espaço de busca"[14]. Busca e otimização são duas faces da mesma moeda. É perfeitamente possível converter um em outro através do seguinte método:

- Para um problema de otimização: elenque todas as soluções possíveis e busque a melhor;
- Para um problema de busca: guarde a melhor solução até ter elencado todas as soluções possíveis.

Problemas de otimização são classificados como problemas de minimização ou maximização, onde um pode ser convertido em outro através de artifícios simples como a inversão do objetivo do problema ou da função de otimização [15]. Otimização é a busca do melhor prático, não o melhor absoluto. Conhecer a relação entre uma variável e outra na solução do problema, por si só, já é um tipo de otimização. Porém um processo de otimização é a "determinação de uma ação que proporciona um máximo de benefício, medido por um critério de desempenho pré-estabelecido"[16]. Conforme exposto, a eficácia e eficiência da solução devem ser garantidas: eficácia, pois o resultado não deve ferir as regras preestabelecidas, eficiência, pois deve obter o melhor resultado possível, dado que foi eficaz. Tarefas típicas do cotidiano para otimização são:

- maximização da relação de custo benefício em projetos de construção;
- minimização dos erros através do rearranjo dos componentes de um processador;
- minimização do tempo gasto com a arrumação da casa; entre outros.

A divisão dos problemas de otimização pode ser da seguinte forma: problemas com variáveis contínuas e discretas(problemas combinatórios). Problemas de variáveis contínuas são onde se procura um conjunto de números ou função como resultado. Problemas de variáveis discretas se procura um conjunto finito ou infinito das melhores combinações de soluções [17]. No final da década de 1960 houve a suposição, ainda não verificada, de que existe uma classe de problemas de otimização onde qualquer algoritmo requer um esforço combinatório que cresce super-polinomialmente com o tamanho do problema [14]. Assim, algumas soluções ótimas ainda possuem custo elevado para sua obtenção (tempos de execução longos), o que as torna impraticáveis. Diante disso também, se optou por encontrar soluções sub-ótimas com tempos de execução aceitáveis. Essa outra forma de otimização é amplamente utilizada, principalmente em inteligência artificial. Há ainda no conceito de algoritmos de otimização os algoritmos gerais que podem ser aplicados a uma grande variedade de problemas e os algoritmos costurados que usam informações específicas do problema tornando sua aplicação limitada em relação a outros problemas.

2.5.2 Critérios de Otimização

Há duas premissas que são consideradas nas técnicas de otimização [14]:

- Existe uma função objetivo "que exprime através de uma escala única a medida de mérito do sistema analisado, ou seja, o quanto à solução analisada é boa dentro do problema em questão".

- O caráter determinístico da avaliação, sendo que "um determinado conjunto de valores das variáveis independentes deve produzir apenas um resultado na função objetivo".

Frequentemente estas premissas não são satisfeitas, observa-se uma pluralidade dos objetivos a satisfazer.

2.5.3 Métodos de Otimização

Há em essência três modelos de métodos de otimização. Estes três tipos podem ser combinados entre si, porém respeitam sempre as seguintes classificações:

Métodos probabilísticos: são métodos que empregam a ideia de busca probabilística, utilizam a convergência da probabilidade em um fator a fim de detectar o resultado.

Métodos numéricos: podem ser divididos em analíticos ou baseados em cálculos numéricos conforme:

- Os métodos analíticos, quando é possível derivar ou pode-se aproximar por outra função derivável a função de custo do problema, bastando resolver a equação.
- Nos métodos baseados em cálculo numérico, técnicas de pesquisa operacional, como o método simplex, técnicas de gradientes ou de estatísticas de alta ordem são empregados na busca do resultado.

Métodos enumerativos: buscam por pontos ótimos no espaço amostral, o que frequentemente resulta em alto custo operacional para quantidades grandes de amostras.

2.5.4 Problemas Conhecidos

Problemas clássicos conhecidos em otimização são: caixeiro viajante, o problema da mochila, o problema de roteamento de veículos, entre outros.

Caixeiro Viajante: A popularidade do problema do caixeiro viajante provavelmente se dá pelo fato de ser de fácil entendimento, mas de difícil resolução. Consiste em: dado um número de cidades, com um número de estradas ligando tais cidades, cada estrada com um custo, encontrar o caminho que minimize o custo total de trafegar por todas as cidades. Pode-se ver uma representação do problema na Figura 2.4. Nela possuímos quatro cidades (A, B, C e D) e cinco rotas (A e B com peso 1, A e C com peso 2, B e C com peso 2, B e D com peso 1 e C e D com peso 1). Para percorrer o caminho de menor custo saindo de A, o caixeiro seguiria ABDC conforme a figura.

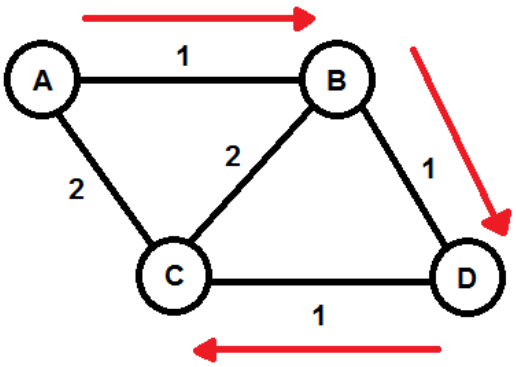

Figura 2.4: Exemplo Caixeiro Viajante.

Se ele saísse de B ou C a rota provavelmente mudaria. Determinar a solução para o problema do Caixeiro Viajante significa determinar também a solução para diversos problemas da computação, um vez que outros problemas podem ser convertidos no problema do caixeiro sem perda de generalidade.

Problema da Mochila: Consiste em preencher uma mochila com itens que possuem um peso e um valor. A mochila possui uma delimitação máxima do peso que é possível carregar. O objetivo é maximizar o valor carregado nos itens da mochila sem que a soma de seus pesos ultrapasse o máximo permitido para a mesma.Pode-se ver uma representação do problema na Figura 2.5.

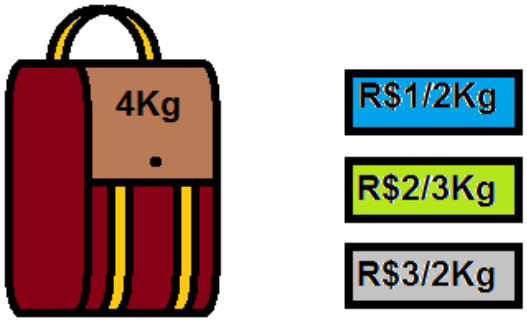

Figura 2.5: Exemplo Problema da Mochila.

Problema de Roteamento de Veículos: É uma composição do problema da Mochila com o problema do Caixeiro Viajante. Consiste em um número de clientes posicionados em pontos de uma cidade, cada cliente possuindo um quantidade de produtos para serem enviados a um depósito. Os pontos da cidade se ligam por rotas de determinado peso e um frota de caminhões, com capacidade máxima cada, deve passar

por esses pontos coletando todos os produtos. Deve-se descobrir qual a menor rota para os caminhões percorrerem.

2.5.5 Custo dos Problemas

Os problemas citados possuem um custo. Este custo é dado pelo melhor algoritmo conhecido para resolvê-los. Os algoritmos podem ser classificados conforme o comportamento da função de seu custo em relação ao número de itens na entrada. Classes comuns de custo são P (Polinomial) e NP (Não Polinomial), ou seja, respectivamente aquelas cuja função de custo é dada por uma função polinomial e um função maior do que polinomial. Em NP, "o espaço de busca cresce exponencialmente com as dimensões do problema"[14].

Muitos acreditam que os problemas NP nunca poderão se tornar P, mesmo com a invenção de outros métodos de otimização. O Método da Redutibilidade pode ser aplicado nestes algoritmos de modo que se um algoritmo eficiente em P para um determinado problema existir, então o algoritmo poderá ser modificado para resolver todos os outros problemas NP [17].

O problema proposto para este trabalho de conclusão se assemelha ao problema da mochila que está em P. Pode-se observar que cada produto possui um risco envolvido (peso do item) e uma rentabilidade (valor do item). Dessa forma deve-se sugerir a melhor aplicação, dado a composição atual da carteira de um cliente, considerando o peso e rentabilidade da próxima aplicação nessa carteira. Porém, tem-se que considerar outros fatores importantes na sua concepção como:

- São múltiplas mochilas e uma mochila influencia no peso máximo da outra;
- Não se pode pedir para o cliente desaplicar seus produtos atuais trocando-os por outros, pois isso poderia ocasionar prejuízo das operações atuais;
- Os prazos de resgate e valor mínimo do produto tem que ser considerados, pois em determinados casos o cliente não pode ficar com o investimento retido por muito tempo;
- Nem sempre a escolha da melhor rentabilidade mantendo o cliente enquadrado é o melhor para diminuir o risco da carteira.

2.5.6 Detecção de Subjetividade

Mesmo que existisse uma solução viável em P para o problema deste trabalho, esta solução não seria boa o suficiente, uma vez que existe uma relação subjetiva entre a aplicação que se está prestes a fazer e as aplicações que já existem na carteira.

Um exemplo dessa relação subjetiva seria: supondo que a carteira do cliente seja em grande parte composta por um fundo que investe em empresas petrolíferas e a melhor rentabilidade para a próxima aplicação seria em ações de um empresa petrolífera ainda mantendo o cliente enquadrado em seu risco. O algoritmo de otimização indicaria as ações dessa empresa como o melhor investimento sem saber que há um risco subjetivo na composição total da carteira. Esse risco reduziria significativamente a rentabilidade da carteira em caso de crise no setor petrolífero.

Um algoritmo bom indicaria outro tipo de aplicação de forma a diluir a carteira em mercados diversos. Essa é uma tarefa trivial para um algoritmo de inteligência artificial, bastando selecionar o espaço amostral corretamente.

2.6 Técnicas de Inteligência Artificial

A inteligência artificial teve seu nome cunhado em 1956, após o final da Segunda Guerra Mundial. O termo abrange uma série de subcampos, de aprendizado e percepção até campos específicos como detecção de doenças entre outros [18].

Essa área pode ser conceituada como sendo todo algorítimo que pertença a algum dos subgrupos: Agir como humano, pensar como humano, agir racionalmente, pensar racionalmente. Porém, suas bases remontam a antiguidade, da concepção filosófica do pensar, do que é racional e o que é ser humano.

As questões filosóficas sobre Inteligência Artificial (IA) não foram conclusivas, de forma que a definição ficou em aberto. Para resolver esse problema testes foram conceituados, de modo que se passasse nos testes seria IA. Pode-se citar como testes que surgiram dessa área o Teste de Turing (elaborado por Alan Turing em 1950), Teste do Jogo da Imitação, entre outros.

Assim como os testes, vários algorítimos de IA surgiram, cada um utilizando um método que identifica melhor algum problema. Algoritmos de Busca Heurística, Algoritmos Genéticos, Redes Neurais Artificiais (RNA) são exemplos das diversas manifestações que a área de Inteligência Artificial obteve.

Atualmente se destacam entre as demais técnicas de inteligência artificial as Redes Neurais Artificiais. São vários os motivos de seu sucesso, dentre eles:

- Barateamento do hardware, principalmente hardware especializado;
- Baixa complexidade de implementação;
- Alta resiliência a falhas;
- Alta precisão do resultado.

Também há pontos negativos nessa área, como por exemplo:

- Desconhecimento de como ocorre as deduções que a rede realiza (os modelos atuais ainda não apresentam justificativas para suas respostas);
- Tempo de treinamento extensivo (embora hardwares especializados como as placas de vídeo tenham reduzido esse tempo de forma considerável equivalendo a outros métodos mais ágeis);
- Falta de definição da arquitetura para uma rede (a dedução de como se organiza a rede para um determinado problema ainda é feita empiricamente, o número de combinações é demasiadamente variado, de modo que não há garantias de que um modelo seja melhor do que o outro a menos que seja testado).

2.6.1 Redes Neurais Artificiais

O sucesso das Redes Neurais Artificiais especificamente as redes Multi-Layer Parceptron (MLP), também conhecidas como Back-Propagation Nets (BPN), foi devido a eficácia do método além de sua ampla versatilidade, ao ponto de várias redes puderem ser compostas em uma única (técnica denominada *Deep Learning*). Essa técnica começou em um estudo realizado em 1943 por McCullock e Pitts sobre a criação de um modelo matemático para o comportamento do neurônio humano [19]. As conclusões da pesquisa foram de que o modelo formal do comportamento de um neurônio deveria seguir duas premissas. A primeira de que um neurônio apenas estará no estado ativado (ou seja, transmite sua sinapse) se sua saída ultrapassar um valor limite. A segunda define que cada entrada de um neurônio terá um conjunto de pesos associados, que podem levar a excitação do neurônio. Conforme as premissas, deduz-se que uma rede neural possui a informação armazenada nos pesos das entradas, que o coeficiente de aprendizagem é proporcional ao produto dos valores de ativação, que os pesos são simétricos, ou seja o peso entre os neurônios A e B é igual ao peso entre B e A e que só ocorre aprendizado quando os pesos são alterados.

O primeiro modelo que surgiu da implementação de redes neurais foi o *Perceptron*, implementado por Frank Rosenblatt em 1958[1]. Ele consistia de uma simples camada de neurônios de entrada e uma de saída que computava uma função. A função é a composição dos resultados dos neurônios, onde cada neurônio é ativado se a soma dos produtos dos pesos com suas entradas ativas for maior do que o valor limite de sua ativação. Esse modelo foi a base para os trabalhos dos modelos de aprendizagem não supervisionada e supervisionada. Exemplos desses modelos são o *Kohonen* e *Backpropagation* respectivamente[1]. O modelo supervisionado treina sua rede baseado em dados de entrada e saída. Os padrões de entrada e saída vão corrigindo os pesos da rede de modo produzir o aprendizado.

Após a demonstração de que o *Perceptron* não poderia representar uma função ou-exclusivo (XOR) em uma única camada. Estabeleceu-se um conturbado período para a área de redes neurais vinculados ao Perceptron. Nesse período houve diminuição de trabalhos sobre o tema. Era necessário a criação de um novo modelo que contemplasse mais camadas para que o Perceptron voltasse ao foco da pesquisa em Redes Neurais Artificiais. Paul J. Werbos propôs um modelo eficiente de *Backpropagation* que possibilitou a inclusão de mais camadas. Nesse modelo o erro é o resultado da comparação entra a saída do processamento da rede nas entradas e a saída do conjunto de dados de treino. Ele é avaliado pelo caminho inverso, calculando a influência do erro das últimas camadas para as primeiras. A adição da função limiar de valores fracionários permitiu também um aumento no espectro de funções passíveis de computação, caso das funções não lineares.

Um exemplo de modelo de Redes Neurais Artificiais pode ser visto na Figura 2.6. Nele encontramos a camadas de entrada, onde os neurônios fazem conexão direta com os dados de N entradas, a camada de saída, onde é mostrado o resultado da simulação em K saídas possíveis da rede para uma determinada entrada e as camadas intermediárias ou ocultas, que possuem todos os neurônios não contemplados nas camadas de entrada e saída.

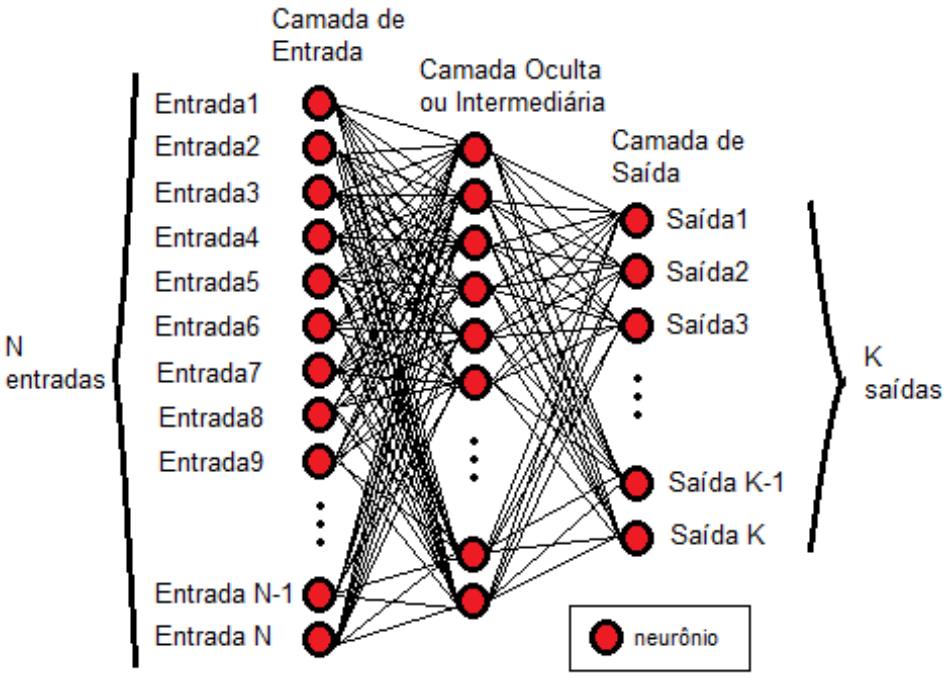

Figura 2.6: Modelo de Rede Neural Artificial.

2.6.2 Funções

As RNAs distinguem dois tipos de função: funções para transferência de sinais e funções de aprendizado. Tais funções possuem comportamentos diversos e afetam a curva de aprendizado e o resultado da rede.

Funções para transferência de sinais, também conhecidas como funções de limiar são responsáveis pela determinação da intensidade em que um neurônio transmite o sinal a outro. São quatro funções de limiar mais utilizadas na academia: a linear, a hard-limiter ou step, a em rampa ou relu, a sigmoid, e a gaussiana, e seguem o seguinte padrão:

Linear: funções da forma da Equação 2.3

$$f(x) = ax \tag{2.3}$$

e se comportam conforme a Figura 2.7.

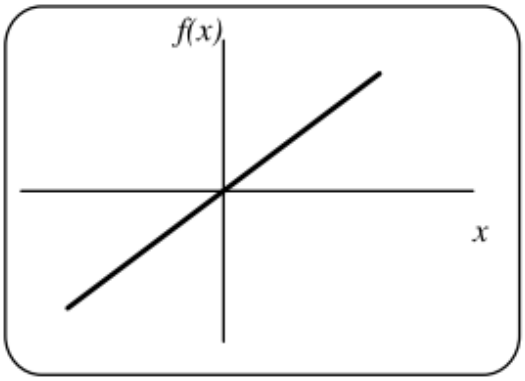

Figura 2.7: Linear:reta resultante da função linear de transferência. (Fonte [1]).

Hard-limiter: funções da forma da Equação 2.4

$$f(x) = \begin{cases} \beta & : se \quad x \geq 0 \\ -\delta & : se \quad x < 0 \end{cases} \tag{2.4}$$

e se comportam conforme a Figura 2.8.

Em rampa: funções da forma da Equação 2.5

$$f(x) = \begin{cases} \gamma & : se \quad x \geq \gamma \\ x & : se \quad |x| < \gamma \\ -\gamma & : se \quad x \leq -\gamma \end{cases} \tag{2.5}$$

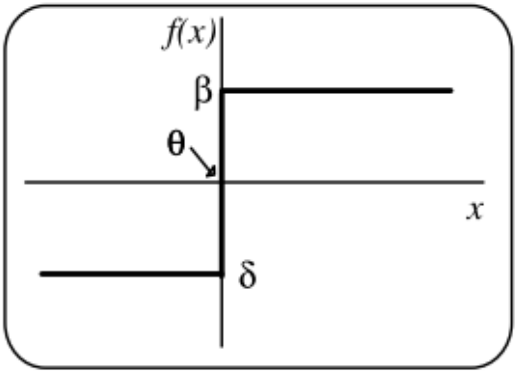

Figura 2.8: *Hard-limiter*: o coeficiente de limiar determina onde será o limite de transferência. (Fonte [1]).

e se comportam conforme a Figura 2.9.

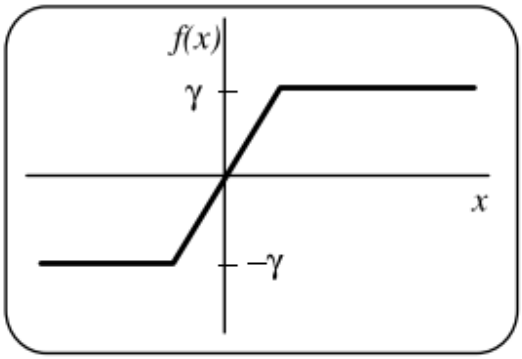

Figura 2.9: Rampa:esta função permite a delimitação de uma área de transição durante a variação da transferência. (Fonte [1]).

Sigmoid: funções da forma da Equação 2.6

$$f(x) = \frac{1}{1 + e^{-ax}} \tag{2.6}$$

e se comportam conforme a Figura 2.10.

Gaussiana: funções da forma da Equação 2.7

$$f(x) = e^{\frac{-x}{v}} \tag{2.7}$$

e se comportam conforme a Figura 2.11.

Uma RNA é um grafo orientado. Também conhecido como dígrafo, porém com pesos nas arestas. Esse peso modula a entrada do próximo nó ou neurônio. Os pesos são

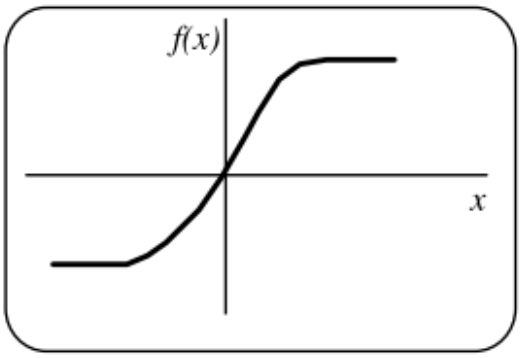

Figura 2.10: Sigmoid: uma transição mais detalhada.(Fonte [1]).

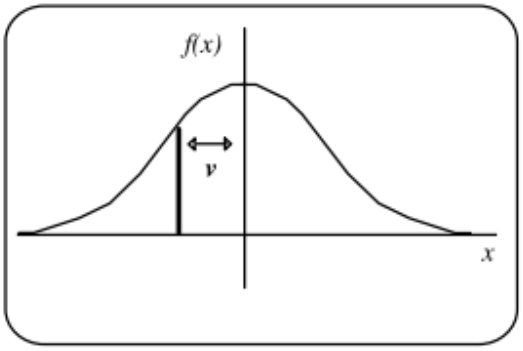

Figura 2.11: Gaussiana: distribuição uniforme baseada na variância. (Fonte [1]).

ajustados durante o processo de aprendizado, porém no processo de simulação ou teste não são.

Os pesos das arestas podem ser positivos ou negativos, indicando se vão inibir ou excitar o próximo neurônio respectivamente. Cada neurônio possui outros neurônios ligados a ele e só possui uma saída que se propaga para todos os outros que estejam ligados a esta. Normalmente, um neurônio propaga sua informação se a soma do produto dos pesos de suas arestas pela saída de cada neurônio ligado às mesmas ultrapassa o valor limite estipulado. Em seguida, o valor é convertido pela função de limiar que inicia o processo novamente para a próxima camada. Isso permite que os neurônios de uma mesma camada atuem em paralelo. A Equação 2.8 descreve a ativação de um neurônio (também conhecida como saída linear)

$$Ativacao_i = f\Big(\sum_{i=1}^{n}(Padrao_{ij}Peso_{ij})\Big) \tag{2.8}$$

Redes Neurais Artificiais podem ser classificadas de diversas formas. Dentre as formas temos pelos tipos de entrada, forma de conexão e tipo de aprendizado. Estas formas de

classificação podem ser combinadas para especificar melhor uma rede[18].

As classificações por tipos de entrada podem ser binários, que aceitam apenas uns e zeros como os modelos de Hopfield e ART. Ou intervalar, que aceita valores numéricos como o caso do *Backpropagation* e *Kohonen*.

As classificações por forma de conexão especifica como o modelo matemático representa a transmissão de sinais. Nesta classificação há três categorias:

- Alimentação à frente, onde os sinais são repassados da entrada para a saída sem intervenção;
- Retroalimentação, onde a saída do neurônio serve de entrada para o mesmo, além das demais entradas;
- Concorrente, onde a saída dos outros neurônios da mesma camada alimenta a entrada do neurônio sendo avaliado.

As classificações por forma de aprendizado podem ser do tipo supervisionado ou auto-supervisionado(não supervisionado). Suas caraterísticas são:

- Para o tipo supervisionado há uma definição sobre a saída que se deseja para cada conjunto de dados, o que força o ajuste dos pesos de modo a representar o sinal desejado;
- Para o tipo auto-supervisionado faz-se uma representação da distribuição das entradas na rede, através do cálculo de probabilidade de cada padrão.

As classificações citadas não são excludentes. Para especificar uma rede pode-se definir um item de cada classificação, por exemplo as redes que serão abordadas neste projeto são intervalares, de alimentação à frente e do tipo supervisionado, características da rede BPN.

O modelo BPN também conhecido como Multi-Layer Parceptron (MLP) é uma derivação do modelo Percetron, porém possui 3 ou mais camadas e uma função de limiar na saída de cada neurônio, que amplia o potencial de classificação do modelo. Por exemplo, a função de limiar do tipo sigmoide para uma BPN seria da forma da Equação 2.9.

$$sgm(S_i) = \frac{1}{1 + e^{-(S_i - \theta)}}, \tag{2.9}$$

onde θ é o coeficiente de limiar e S_i é a soma ponderada das entradas do neurônio (saída linear)i.

Neste modelo o erro obtido na comparação da saída projetada com a saída real é transferido para as camadas anteriores de forma inversa. Assim calcula-se o erro da última

camada, depois dado a variação na última camada, calcula-se o erro na penúltima e assim sucessivamente. Desse método que surge o nome *backpropagation* (retro-propagação).

O cálculo do erro da última camada tem a forma da Equação 2.10.

$$\varepsilon s_i(t) = S(t)(1 - S(t))(d_i(t) - S(t)), \tag{2.10}$$

onde o erro da última camada para o neurônio da saída i é dado por $\varepsilon s_i(t)$, o resultado da simulação t é dado por $S(t)$ e o resultado real do conjunto de dados de treino é dado por $d_i(t)$, sendo i o neurônio de saída sendo avaliado. A partir do erro obtido cada peso da última camada é ajustado segundo a Equação 2.11.

$$P_i(t + 1) = P_i(t) + \alpha\varepsilon s_i(t)En(t), \tag{2.11}$$

onde P_i é o peso do neurônio de saída i, α é o coeficiente de aprendizagem e En o vetor resultante da saída da camada anterior.

O erro das camadas intermediárias segue o mesmo princípio do erro da camada de saída, dado pela Equação 2.12.

$$\varepsilon_i(t) = En(t)(1 - En(t)) \sum_k \varepsilon_k \rho_{ik}(t), \tag{2.12}$$

onde En é o vetor resultante da saída da camada anterior à camada sendo avaliada, k é o número de neurônios da próxima camada que este neurônio se conecta, ε é o erro do nodo k da próxima camada, ρ_{ik} é o peso entre o neurônio sendo avaliado (i) e o neurônio da próxima camada (k).

O peso das camadas intermediárias é dado pela Equação 2.13.

$$P_i(t + 1) = P_i(t) + \alpha\varepsilon_i(t)En(t) + \mu(P_i(t) - P_i(t - 1)), \tag{2.13}$$

onde μ é o coeficiente de aceleração de convergência denominado *momentum*. Esse coeficiente evita que o erro da rede fique presa em um mínimo local, além de aumentar a velocidade com que se encontre um mínimo global do mesmo.

O algoritmo para o treinamento da rede é composto pelos seguintes passos:

1. Inicialização da rede com valores randômicos;
2. Cálculo da saída limiar da rede através da entrada e aplicação da função de limiar;
3. Cálculo e atualização dos pesos na última camada;
4. Cálculo do erro total obtido;
5. Cálculo e atualização dos pesos da camada intermediária anterior;

6. Cálculo do erro da camada intermediária anterior;
7. Repetir o item 5 até que não haja mais camadas;
8. Repetir o item 2 até que os dados tenham acabado ou que o valor da saída desejada tenha sido atingido.

2.7 Mineração de Dados

Segundo Berry e Linoff [20], mineração de dados é a busca por padrões e regras significativas através da análise e exploração dos dados. Essa busca pode ser feita de forma automática ou semi-automática e auxilia na tomada de decisão, prospecção de novos campos e significados, administração de dados, entre outras atividades.

2.7.1 Conceitos básicos

A mineração de dados possui três metodologias, que são aplicadas para descoberta dos dados, onde cada uma possui uma série de passos[21]. As metodologias utilizadas em mineração de dados são: Teste de Hipótese, Descoberta de Conhecimento Direto e Descoberta de Conhecimento Indireto.

O comportamento do analista de dados (responsável pela mineração de dados) deve seguir para cada metodologia está descrito como:

Teste de hipótese: Nessa metodologia, há uma hipótese e se verifica qual a validade dela através dos dados. É a metodologia padrão, importada do modelo científico. O analista de dados deve criar a hipótese e definir qual os dados necessários para sua validação. Em seguida deve preparar os dados e desenhar as validações finais do processo, a fim de confrontar a hipótese. Por fim, deve validar a hipótese e agir baseado na mesma, começando o processo novamente.

Descoberta de conhecimento direto: Essa metodologia é usada quando já se tem conhecimento prévio dos dados e de como interpretá-los, porém sem nenhuma hipótese. Para ela, identifica-se os dados que se queira trabalhar, prepara-se a análise, dividi-se em conjuntos de treino, teste e avaliação. Em seguida, treina-se o modelo com os conjuntos de treino, refina-se com os dados de teste e avalia-se com os de avaliação.

Descoberta de conhecimento indireto: Usa-se essa metodologia quando não há conhecimento da natureza dos dados e se quer descobrir um padrão a partir deles. Nesse caso, deve-se identificar as bases disponíveis, prepara para análise essas bases,

selecionar técnicas de descoberta de estruturas, identificar alvos potenciais obtidos no passo anterior e proceder com a validação desses alvos.

2.7.2 Modelos

Há diferentes modelos que regem os processos de extração de conhecimento no campo da mineração de dados. Estes modelos definem como é o processo para garantir uma boa extração.

As extrações feitas pelo modelo Knowledge-Discovery in Databases (KDD) são boas para identificar relações que não são observáveis pelos especialistas, assim como na validação de conhecimento extraído anteriormente. O KDD é composto pelas etapas de seleção, pré-processamento, transformação, mineração, interpretação e avaliação. Na etapa de seleção agrupa-se os dados, na etapa de pré-processamento os dados passam por uma adequação, retirando duplicidades e inconsistências, na etapa de transformação modificam-se os dados facilitando o uso dos algoritmos da etapa de mineração, que por sua vez aplica tais algorítimos na descoberta de padrões. Por fim, avalia-se se esses dados condizem com a interpretação lógica do ponto de vista negocial, validando-os com outra base.

O CRoss Industry Standard Process for Data Mining (CRISP-DM) é outro modelo criado com as abordagens padrão utilizadas por especialistas na área de mineração. Suas vantagens estão em ser independente do segmento da empresa ou de ferramentas e possuir proximidade com o modelo KDD. Ele também é dividido em etapas. Na primeira etapa se identifica o negócio e seus objetivos, na segunda se identifica os dados que podem interessar para obter tais objetivos, após, na terceira etapa, há a preparação dos dados, que serão consumidos pela quarta etapa, a etapa de modelagem onde ocorre o treinamento e captura de padrões. Avalia-se o modelo criado anteriormente na quinta etapa buscando as justificativas e oportunidades que circundam o modelo. Ao final, na sexta etapa, ocorre a implantação do modelo, organizando o conhecimento gerado.

Outros modelos surgiram, acadêmicos e comercias. Entre modelos acadêmicos pode-se citar “Metodologia de Klemettinen” e “Metodologia de Feldens” [22]. Já há uma infinidade de modelos comerciais, normalmente aliados a produtos para análise de dados. Pode-se citar o Master Data Services (MDS) da Microsoft, entre outros.

2.7.3 Técnicas e Tipos de Informação

Há uma variedade de técnicas na mineração de dados, cada qual mais adequada para o tipo de informação que se deseje obter. Porém, para encontrar a resolução de um problema, deve-se identificar uma série de informações, que consistem da execução de uma série de técnicas em uma base de dados.

São conhecidas as seguintes técnicas de mineração de dados:

Análise de seleção estatística: é uma forma de agrupamento para detecção de ocorrências concomitantes. Tipicamente usada na criação de pacotes de produtos.

Raciocínio baseado em memória: utiliza exemplos conhecidos para a previsão de exemplos desconhecidos, combinando os valores mais próximos de um determinado padrão detectado nessa geração. Pode ser utilizado para preenchimento de uma base de dados, ou geração aleatória de valores dentro de um padrão.

Algorítimos Genéticos: calcula qual os padrões mais bem sucedidos e gera mutações desse padrão, a fim de que, ao final de uma sequência de iterações, tenha um padrão bom para as condições estipuladas. Usado para otimização de padrões.

Detecção de agrupamentos: detecta os agrupamentos em um conjunto de dados de modo a separar os grupos por semelhança. Essa é uma técnica exploratória, quando não se conhece a natureza dos dados desconhecidos e se quer obter alguma relação dos mesmos.

Análise de Vínculos: utiliza os vínculos já existentes na base para detectar padrões indiretos. Muito utilizada em grandes bases cujas ligações se confundem.

Árvores de decisão e indução de regras: criam um modelo de classificação baseado em decisões, o que explicita o comportamento que se está adotando para classificar os dados. É muito utilizada para aprimorar regras negociais, uma vez que explicita qual é seu comportamento.

Redes Neurais: de grande versatilidade, podem classificar, prever, estimar, entre outras atividades da mineração de dados. São usadas em todos os tipos de atividade nessa área.

2.7.4 Tarefas

Harrison (1998) [23] identificou que a mineração de dados possui uma série limitada de tarefas válidas apenas sob certas circunstâncias. São 6 as tarefas associadas à mineração de dados e são descritas como: Classificação, Estimativa, Previsão, Agrupamento por afinidade, Segmentação ou *Clustering* e Descrição. Cada tarefa possui a seguinte definição:

Classificação: A classificação é uma tarefa corriqueira na mineração de dados. Consiste em detectar um padrão em algo e ligá-lo à classe daquele padrão. A classificação trata de valores discretos, idealmente, que possam ser separados em um bom conjunto de classes e que sua representação esteja bem definida na base amostral.

Estimativa: A estimativa trabalha com valores já conhecidos estipulando um valor contínuo para uma entrada. Ela supõem um valor baseado em dados que identificam um range, se comportando como uma fórmula.

Previsão: A previsão é equivalente à classificação com exceção de que o resultado é um conjunto de classes futuras, que não estejam presentes nos dados atuais.

Agrupamento por afinidade: Agrupamento por afinidade representa encontrar os conjuntos de elementos que tenham similaridade. Ou seja, dado um elemento, quais outros tem maior probabilidade de serem escolhidos em conjunto com este.

Segmentação ou *Clustering*: Segmentação é a tarefa de encontrar as classes de um conjunto de dados. Sem que se saiba anteriormente como será essa divisão. Cabe ao interpretador verificar qual é a informação que pode ser retirada de cada classe descoberta.

Descrição: Descrição consiste em encontrar em uma base de dados complexa regras simples que descrevam a mesma.

2.7.5 Ferramentas

Existem ferramentas pagas e gratuitas para a extração de conhecimento das bases de dados. Várias otimizadas para essa atividade, porém é possível fazê-lo através de outros métodos, onde se constrói a solução do princípio. Para extração e tratamento de dados de bases corporativas fechadas de grandes volumes com o caso dos bancos, é necessário mesclar métodos atuais com métodos legados, uma vez que tais instituições não costumam migrar todo seu parque tecnológico para novas tecnologias sem que antes estejam consolidadas. Para muitos casos deve se fazer a portabilidade do antigo para o novo. Tecnologias legado podem ser utilizadas para se fazer a ponte com o processamento de dados das novas. Pode-se citar entre as tecnologias legado Job Control Language (JCL) e COmmon Business Oriented Language (COBOL), assim como pode-se citar entre as novas tecnologias a linguagem *Python*. Tais tecnologias apresentam as seguintes características:

JCL: É uma linguagem utilizada para execução de sequencias de programas e comandos diretamente no sistema operacional IBM. Tais comandos incluem a transformação direta de uma tabela do banco de dados em um arquivo, chamada de programas para ordenar arquivos, combinar arquivos já ordenados, juntar dois arquivos através da combinação de chaves, entre outras possibilidades [24].

COBOL: Esta linguagem foi conceitualmente criada para processamento de arquivos dos meios de negócios (processamento que precisava de pouca precisão nos cálculos),

porém, em sua segunda versão, também incorporou tratamento de banco de dados, o que prolongou sua vida em relação às demais linguagens de programação de sua geração. Mais tarde foram incorporados conceitos de orientação a objetos, execução em ambientes distintos da IBM, novos tipos de dados e compatibilidade com o tratamento de dados de outras linguagens, o que deu uma sobrevida aos programas COBOL [25].

Python: Esta linguagem representa um grande avanço nas linguagens de computação. É uma linguagem de última geração, interpretada, orientada a objetos, funcional e de tipagem dinâmica. Foi criada por Guido van Rossun, em 1991, e possui um modelo de desenvolvimento aberto organizado pela fundação que leva seu nome. Suas maiores vantagens são possuir uma sintaxe clara e incorporar diversas recursos de suas bibliotecas, módulos e *frameworks* de terceiros [26]. Dentre as bibliotecas interessantes para o estudo em redes neurais pode-se citar *Theano*, *Numpy*, *Keras* e *Matplotlib*. Cada biblioteca pode ser descrita como:

- **Numpy** é um pacote para computação científica em *Python*. Este pacote possui diversas funções para criação e operação com matrizes de várias dimensões. No *Python* os tipos de dados são tratados como objetos, o que acrescenta complexidade ao processamento. Porém a biblioteca *Numpy* trata os números como variáveis simples de tipagem estática (baseada em C), o que reduz a complexidade. Possui ferramentas para integrar *C*, *C++* e *Fortran*, além de contar com diversas funções de álgebra linear, transformadas de Fourier e números randômicos[27].
- **Theano** é uma biblioteca do *Python* que permite utilizar expressões matemáticas, em particular as utilizadas em *arrays* multidimensionais. Sua performance está próxima da linguagem *C*, uma vez que o compilador otimiza a forma de acessar os dados e de executar as operações em um alto nível. Outros recursos com o processamento em placas de vídeo, permitem que seja ainda mais rápido que um programa em *C*. O *Theano* combina aspectos de computação algébrica e otimização, principalmente em expressões que são avaliadas frequentemente, dessa forma pode minimizar o código gerado, aumentando sua performance. Ainda conta com algumas vantagens, como expressões já otimizadas disponíveis para uso, como diferenciação automática [28]. Entre outras de suas características estão a utilização da biblioteca *Numpy* para cálculo com *arrays* multidimensionais, simplificação aritmética, otimização do uso de memória, fusão de *loopings* para operações elemento a elemento de uma matriz e melhora da estabilidade numérica em operações como as logarítmicas.

- **Keras** é uma biblioteca de alto nível escrita em *Python* para criação de redes neurais. É capaz de usar as bibliotecas *Theano* ou *ThensorFlow*. Foi desenvolvida com o objetivo de diminuir o tempo de criação de redes neurais. Suporta redes neurais convolucionais e redes recorrentes, conectividade arbitrária de diversos tipos de redes neurais e pode rodar em placas de vídeo [29].
- **Matplotlib** é uma biblioteca escrita em *Python* para visualização de gráficos. Ela interpreta uma tabela conforme a função chamada. Cada função apresenta um tipo de gráfico, desde gráficos de barra a gráficos de dispersão em três dimensões [30].

2.8 Revisão Bibliográfica

Conforme Wuerges e Borba[31] em um estudo feito sobre a distribuição dos artigos de inteligência artificial aplicados à área financeira durante os anos de 2000 e 2007, a maior parte da distribuição engloba bolsas de valores, derivativos, *commodities*, *forex*, gestão de carteira e juros. O estudo de Wuerges e Borba detecta que ainda há um crescimento nessa área, o que é uma evidência da atratividade e contemporaneidade do tema. Também ficou evidente que os estudos sobre investimentos não levavam em conta diversos tipos de investimentos ao mesmo tempo, se limitando a uma área como ações, derivativos ou fundos.

O Trabalho de Conclusão de Curso de Marília Terra de Mello em Ciência da Computação na Universidade Federal de Pelotas [32] com o título "Aplicação de Redes Neurais Artificiais no processo de precificação de ação", propõem uma rede neural com o objetivo de prever o preço futuro de ações, embasando as decisões de compra na composição de uma carteira. Esse trabalho conseguiu demonstrar que pode se aproximar o valor real do ativo no futuro, com o obtido através de uma rede neural.

A Dissertação de Mestrado de Carlos Henrique Dias de nome "Um novo algorítimo genético para otimização de carteiras de investimento com restrições de cardinalidade" [33], de 2008, propõem que um investidor possa escolher o melhor investimento conforme sua aversão ao risco. Esse trabalho foi apresentado na Universidade Estadual de Campinas e os resultados foram promissores, uma vez que indica uma diminuição no tempo para treino do algorítimo que tinha por finalidade compor um carteira de investimentos saudável ao cliente.

O Trabalho de Formatura de Luiz Paulo Rodrigues de Freitas Parreiras da Escola Politécnica da Universidade de São Paulo [34], em 2003, sob o título "Modelo Genético-Neural de gestão de carteiras de ações", propôs um novo algorítimo de inteligência artificial para análise de carteiras de ações, ao juntar redes neurais e algorítimos genéticos. A

conclusão foi de que as redes neurais podem ser eficientes na previsibilidade do preço de ações quando aliadas ao algorítimo genético que calcule a relação risco-retorno dos investimentos.

No Trabalho de Conclusão de Curso, "Uso de Data Mining no mercado financeiro" [35] de 2008, Fernando Rafael Stahnke, o autor, demonstra como é possível utilizar técnicas de mineração de dados para identificar padrões de comportamento e assim determinar a tendência futura de ativos.

Giuliano Padilha Lorenzoni demonstrou em sua Dissertação de Mestrado de 2006 sob o título "Uma investigação estatística sobre análise técnica" [36], que há ligação direta entre a análise técnica e a antecipação dos preços dos ativos.

Um estudo realizado em 2010 por Berenstein [37] fez uma análise de diferentes algorítimos de classificação aplicados à 10 ativos da bolsa de valores. Esses algorítimos tinham o objetivo de prever quando aplicar ou não em algum dos ativos. Os resultados obtidos demonstraram que as técnicas de mineração de dados podem ser aplicadas no mercado acionário com grande eficiência.

O artigo escrito por Luiz Henrique Debei Herling, Marcus Vinícius Andrade de Lima, Gilberto de Oliveira Moritz e Pedro Henrique Marangoni [38] em 2012 descreve como criar uma rede neural que indique qual a melhor próxima compra dentre os dez principais ativos do IBOVESPA. Esse algorítimo criado apresentou grande performance em comparação com as previsões de mercado.

Em 2011 foi escrito o livro "Inteligência Artificial nos Investimentos" [39] por Roberto Pontes, onde em seu terceiro capítulo descreve a comparação entre diversos tipos de fundos e como a inteligência artificial pode ajudar na detecção da hora de comprar e sair de um fundo de ações e até renda fixa. O livro analisa as variações no preço da cota dos fundos entre 2001 e 2010 e treina a rede de forma a saber qual dos fundos deve fazer uma aplicação mensal de quinhentos reais. Ao final compara o fundo híbrido criado pela rede com o desempenho desses fundos caso a aplicação mensal tivesse sido feita apenas em um deles. Os resultados mostram que uma combinação de fundos, distribuindo o risco de investimento, tende a ter melhor rendimento à longo prazo. Também demonstra que a rede pode alterar a distribuição em cada fundo ao longo do tempo a fim de maximizar os lucros. Este estudo contém apenas fundos de investimento, não se refletindo para todos os tipos de investimento.

Em sua tese para o título de Mestre em Engenharia de Produção de 2013 [13], Paulo Henrique Kaupa descreve a aplicação de Redes Neurais e a Teoria de Rough Sets para a composição de carteiras acionárias. A comparação entre os modelos criados aconteceu com *benchmarks* de mercado em relação à rentabilidade e com relação ao risco foi utilizado o Modelo de Markowitz e a Equação de Ponderação. As técnicas utilizadas demonstraram

tendência de alta dos ativos nos dias subsequentes, além da diminuição do risco da carteira. Este trabalho foi o ápice de uma série de estudos realizados nos anos anteriores sobre o mercado de ações.

Entre os trabalhos mais recentes sobre a área está o trabalho de Ciniro A. L. Nametala, Alexandre Pimenta, Adriano César M. Pereira e Eduardo G. Carrano [40]. O trabalho descreve uma estratégia automatizada de investimentos através de RNA. Há duas RNAs envolvidas, uma para realizar a previsão do preço da ação, e outra para decidir se utiliza o resultado da RNA anterior ou outros métodos de previsão de preço. Os resultados da abordagem são comparados com os preços reais obtidos nas ações, para a maioria dos ativos, o robô investidor obteve uma taxa de acerto maior do que a taxa de erro. Esse trabalho é uma demonstração da versatilidade das redes neurais, que podem empregar diversos métodos na aplicação de investimentos.

Diante dos conhecimentos e metodologias apresentadas, verificou-se a necessidade de aprofundamento nos temas referentes a investimentos em redes neurais, uma vez que, muitos fatores não são considerados no estado da técnica atual.

Capítulo 3

Sistema Proposto

Neste capítulo é descrita a metodologia proposta para a solução do problema de se encontrar um bom investimento conforme o perfil de investidor e sua carteira de investimentos, baseada em redes neurais artificiais. Inicialmente os dados disponíveis são descritos e a sua análise é realizada de modo a justificar as escolhas subsequentes.

3.1 Estrutura dos dados

A coleta de dados foi realizada em um banco público de projeção nacional, com uma base de mais de onze milhões de investidores, seis milhões, dos quais, são pessoas físicas. O sistema de origem dos dados é responsável pela aplicação do Análise de Perfil de Investidor (API). Por esse motivo possui os dados relevantes para criação da rede neural, como risco de produtos, saldo por produto do cliente, enquadramento, entre outros.

Apenas as pessoas físicas que realizaram aplicações foram selecionadas para o treino e testes, uma vez que as pessoas jurídicas necessitem de uma análise mais detalhada devido a diversos fatores. Entre os fatores estão questões legais, pois são mais propensas a não participar do API, e questões de volume, pois possuem tendência a conseguir taxas diferenciadas nos investimentos devido ao volume de suas aplicações, o que invalida o objetivo deste trabalho.

A fim de efetuar uma coleta representativa dos dados, foram considerados todos os investidores que realizaram aplicações em setembro de 2016.

Os dados dos experimentos são obtidos das bases necessárias para a aplicação do processo API no banco em questão.

3.1.1 Questionários do Cliente

É uma exigência da legislação vigente que cada cliente investidor possua um ou mais questionários respondidos ou assine o termo de recusa de resposta. Dessa forma, o sistema só permite prosseguir com as aplicações caso o investidor tenha passado pelo processo API. Caso o cliente tenha optado por responder o questionário, o resultado das respostas será o perfil de investidor. Ou seja, lhe é atribuída de uma característica dentre: Conservador, Moderado, Arrojado ou Agressivo. Essa característica, também conhecida como perfil, orienta qual comportamento o investidor deve possuir na hora de investir.

Cada questionário varia conforme o mercado em que o cliente se encontre. Clientes de mercados diferentes recebem questionários diferentes, porém todos medem o perfil de investidor ao final. O mercado revela qual o nível de entendimento do cliente sobre investimentos. Os clientes são segmentados através diversas características que indiquem qual seu nível de entendimento sobre investimentos, através de um processo automático do sistema. Portanto, quando um questionário é respondido, também é associado o mercado que o clientes se encontra ao questionário, por exemplo: se o cliente se encontra em um mercado "Varejo" ou "Alta Renda". A combinação perfil e mercado dita quais devem ser as distribuições de risco dos produtos do investidor.

Para fins legais, sempre deve ser considerado na análise do perfil e do mercado, o questionário mais recente do cliente. O que torna a data de preenchimento do questionário relevante para consolidação dos dados na entrada da rede.

3.1.2 Produtos do Cliente

Para verificação da conformidade dos investimentos do cliente ao seu perfil de investidor, é necessária a informação dos saldos dos produtos de investimento do mesmo, assim como na obtenção da rentabilidade da carteira. O sistema deve alertar se a próxima aplicação que será realizada está conforme com o perfil e mercado do cliente, o que implica no gerenciamento do risco dos produtos pelo próprio cliente. Outra informação importante obtida pelo saldo dos produtos do cliente é a rentabilidade da carteira. Essa informação deve ser considerada na distinção dos investidores que possuem rentabilidade acima da média, que é a entrada da rede neural.

3.1.3 Enquadramento do Cliente

Com base nas informações coletadas dos questionários e dos produtos dos clientes é possível identificar quais clientes estão aderentes ao seu perfil. Por exemplo: para um cliente que pertença ao mercado "Varejo" e possua um perfil de investidor "Conservador", é permitido apenas dez porcento de seus investimentos em produtos de alto risco. Caso

o cliente possua todos os seus investimentos em baixo risco, como poupança, o mesmo estará aderente. Se ele possuir até dez porcento de seus investimentos em alto risco, ainda se encontrará aderente. Porém se o cliente possuir todos os seus investimentos em alto risco, como ações, estará inaderente. A demonstração deste conceito pode ser visto na Figura 3.1.

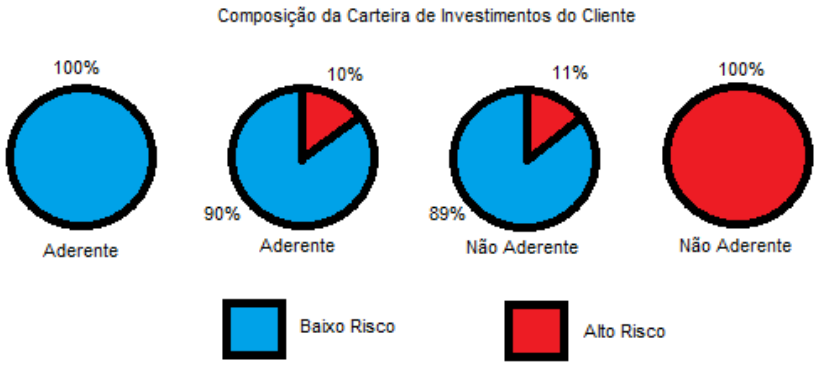

Figura 3.1: Comparativo das situações possíveis da carteira de um cliente.

Considerando a regra de enquadramento anterior, é possível calcular quanto um cliente pode aplicar em determinado produto. Cada produto possui um risco associado, que, para que o cliente não fique desenquadrado, não deve ultrapassar o máximo permitido para o risco desse produto. Dessa forma é possível calcular o máximo aplicável antes do desenquadramento em um produto através da Equação 3.1

$$VA = \frac{PM \cdot TO - TR}{1 - PM}, \tag{3.1}$$

onde PM é o percentual máximo para o risco da aplicação, TO é o total investido pelo cliente, TR é o total que o cliente já possui alocado no mesmo risco do produto e VA é o valor da aplicação resultante.

3.1.4 Rentabilidade do Produto

Cada produto possui uma rentabilidade associada. O cálculo da rentabilidade é realizado subtraindo-se o valor final obtido de uma aplicação hipotética no produto há 365 dias. Em seguida, é dividido o resultado da subtração pelo valor aplicado inicialmente. Quando um produto não possui 365 dias de existência, é feita uma progressão geométrica levando em consideração os valores dos dias já conhecidos, projetando-se 365 dias à frente. Então é feito o cálculo anterior tendo como valor final o projetado.

A rentabilidade dos produtos é utilizada na obtenção da rentabilidade da carteira. Cada produto da carteira de investimentos de um cliente tem sua rentabilidade multi-

plicada pelo percentual de sua participação na carteira. Em seguida são somados os resultados das multiplicações anteriores. O valor final é a rentabilidade da carteira.

3.1.5 Aplicações

Uma aplicação pode ser feita em qualquer produto de investimento, por qualquer cliente, a qualquer hora. Como cada aplicação realizada deve passar pelo processo API, a fim de conferir a aderência do cliente, o valor da aplicação é passado ao processo e gravado. Caso a aplicação que o cliente esteja prestes a fazer o desenquadre, o mesmo é alertado de sua condição. Apenas os clientes que realizaram aplicações são considerados na análise, uma vez que, a aplicação é a saída da rede proposta.

3.1.6 Metodologia

Para a montagem da entrada da rede são consideradas os clientes que realizaram aplicações no mês de setembro do 2016, cuja rentabilidade da carteira ficou acima da média e tenha permanecido enquadrados ao final da coleta. Para isso foram coletados os clientes que realizaram aplicações, desses clientes foi analisada a composição de cada carteira utilizando a regra de enquadramento. Dos que se encontravam na condição de enquadrados, a rentabilidade da carteira de cada cliente foi calculada. Foi feita a média das rentabilidades e foram selecionados os clientes cuja rentabilidade de sua carteira estava acima dessa média.

Por fim, são geradas as entradas e saídas da rede neural. Cada aplicação realizada se torna uma coluna para a saída da rede. Cada produto da carteira do cliente se torna uma coluna de entrada seguido de outra coluna com o percentual carteira. Essas informações devem ser importantes na tomada de decisão da rede, que pode perceber nuances entre a relação de uma aplicação e de um percentual de determinado produto da carteira. O perfil do cliente se torna uma coluna de entrada, uma vez que há uma relação entre o código do perfil e o grau de risco que o cliente está disposto a tolerar. Cada tipo de mercado também se torna uma coluna, isso se deve por não haver uma relação direta entre o código do mercado e o nível de conhecimento sobre investimentos do cliente.

3.2 Distribuição dos dados

O período utilizado para a coleta dos dados corresponde às aplicações realizadas no mês de Setembro de 2016. A base de insumo das redes neurais possui as seguintes características:

Vetores: O base possui 122496 (cento e vinte e dois mil quatrocentos e noventa e seis) vetores.

Clientes: O número de clientes no arquivo de treino é de 113503 (cento e treze mil quinhentos e três). O número de clientes é menor do que o número de vetores devido a um cliente poder realizar mais de uma aplicação no mesmo mês. Ao contar o total de produtos e aplicações, obtemos 718 (setecentos e dezoito).

Produtos Carteira: O total de produtos comercializados pelo banco é de 1322 (um mil trezentos e vinte e dois), porém os clientes considerados só possuíam 718 (setecentos e dezoito) desses produtos junto com as aplicações realizadas, dos quais 681 (seiscentos e oitenta e um) eram produtos de suas carteiras e 276 (duzentos e setenta e seis) eram aplicações. A distribuição dos produtos das carteiras na base de treino segue o gráfico Figura 3.2.

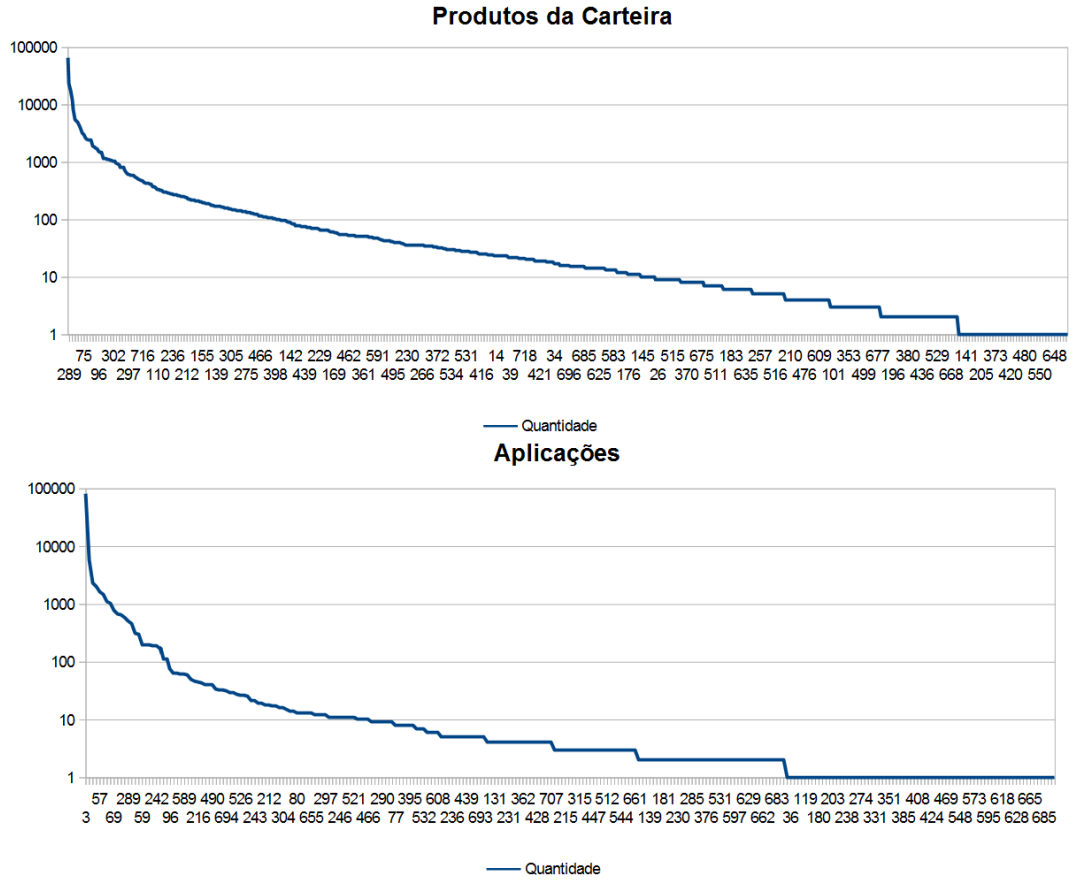

Figura 3.2: Incidência de Aplicações por produto na base de treino.

Pode-se notar que a incidência de alguns produtos é maior do que outros. Essa diferença não deveria ser considerada na montagem da rede neural, pois pode tendenciar a rede a apresentar sempre os mesmos produtos. O que se quer é que ela apresente qualquer produto que maximize o rendimento sem perder o enquadra-

mento do cliente. Para isso é necessário que as aplicações tenham um volume de dados próximo, de modo que não se apresente sempre os mesmos produtos.

Aplicações: Da mesma forma, é possível efetuar 718 aplicações diferentes, porém a quantidade de aplicações efetivas se concentra e 276 possibilidades.

3.3 Modelo Proposto

Como modelo de implementação proposta, a rede neural seria a parte intermediária de um sistema, que contemplaria uma camada de apresentação e busca ao banco de dados dos insumos necessários para a simulação da rede.

A Figura 3.3 exemplifica a implementação do sistema. No caso, o cliente começa a interação se autenticando, acessa a área de investimentos, busca por uma sugestão de investimentos. O sistema obtém os insumos da rede, que são os investimentos atuais do cliente, bem como o perfil de investidor e o mercado que se encontra, então a rede simula as entradas e retorna a melhor aplicação para o cliente. Por fim o cliente realiza a aplicação proposta.

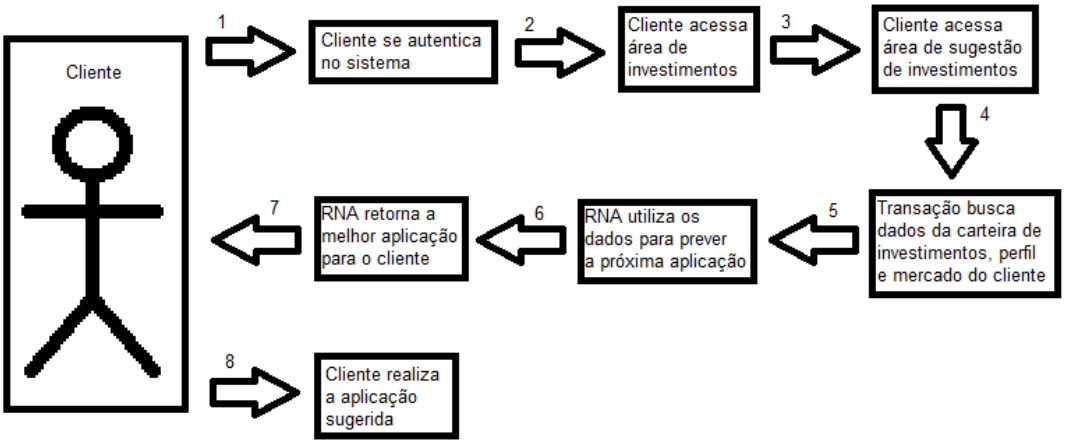

Figura 3.3: Modelo do contexto de implementação da RNA.

Para futuras atualizações da RNA, seria incluída uma rotina de processamento de dados novos e os integraria à rede, através de trechos de código descritos posteriormente neste trabalho.

3.4 Implementações Propostas

O arquivo principal de treino foi processado através do código que segue. Para determinação de um bom ponto de eficiência, onde há maior ganho de tempo em relação a acurácia resultante, foram feitos testes com diversas configurações do código abaixo.

Exemplo de código 1:

```
# importacao de bibliotecas
import theano
import time

from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import SGD
import numpy

import matplotlib.pyplot

# numeros randomicos
seed = 7
numpy.random.seed(seed)

# carregamento dos arquivos
X = numpy.loadtxt("PRDLINHATREINO.TXT", delimiter=",")
Y = numpy.loadtxt("APLLINHATREINO.TXT", delimiter=",")

# criacao do modelo
model = Sequential()
model.add(Dense(1441, input_dim=1441, init='uniform', activation='sigmoid')
    )
model.add(Dropout(0.5))
model.add(Dense(1080, init='uniform', activation='sigmoid'))
model.add(Dropout(0.5))
model.add(Dense(718, init='uniform', activation='sigmoid'))

# inicio da contagem do tempo para treino
t0 = time.time()

# compilacao do modelo
sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['
    accuracy'])

# treino do modelo
model.fit(X, Y, nb_epoch=8, batch_size=32)
```

```
37
# fim da contagem do tempo
39 t1 = time.time()

41 # avaliacao do modelo
scores = model.evaluate(X, Y)
43
print("\n%f seconds %s: %.2f%%" % ((t1 - t0), model.metrics_names[1],
    scores[1]*100))
45
#matriz de confusao
47 def confusion_matrix(X, Y, model):
     model_results = model.predict(X)
49   assert model_results.shape == Y.shape
     num_outputs = Y.shape[1]
51   confusion_matrix = numpy.zeros((num_outputs, num_outputs), dtype=numpy.
    int32)
     predictions = numpy.argmax(model_results,axis=1)
53   true_results = numpy.argmax(Y,axis=1)
     for actual_line in range(Y.shape[0]):
55       confusion_matrix[true_results[actual_line], predictions[
    actual_line]] = numpy.sum([confusion_matrix[true_results[actual_line],
    predictions[actual_line]], 1])
     return confusion_matrix
57


59 #apresentacao da matriz de confusao
logconfmat = numpy.log(confusion_matrix(X, Y, model))
61 image  = matplotlib.pyplot.imshow(logconfmat)
matplotlib.pyplot.show()
```

Este código possui as seguintes características:

Importação das bibliotecas: parte onde são importadas as bibliotecas do *Theano*, *Time*, *Numpy* e *Keras*. A biblioteca *Theano* possui os métodos para utilização otimizada de tabelas e para execução em placas de vídeo. A biblioteca *Time* possui os métodos de captura de tempo para o teste da performance. A biblioteca *Numpy* possui os métodos de criação de números e tratamento de tabelas otimizados para pesquisa científica. A biblioteca *Keras* possui as definições dos métodos para execução em alto nível de redes neurais (para que o *Keras* funcione é necessário a importação das bibliotecas *Numpy* e *Theano*). A biblioteca *Matplotlib* possui os métodos para visualização de gráficos em várias formas.

Números randômicos: parte onde é declarado que as tabelas internas do Keras possuirão números randômicos gerados igualmente entre uma execução e outra. Dessa forma garante-se que se esteja comparando implementações iguais das redes neurais. No caso, o método que gera os números randômicos é o `numpy.random.rand(x,y)`, utilizado pelo *Keras* para inicializar os neurônios e acrescentar ruído. Esse método alimenta uma tabela de `x` colunas e `y` linhas e funciona recursivamente, gerando números a partir da multiplicação da semente, definida em `seed(7)`, por um número grande e retirando o módulo desse número. Como a semente é a mesma, sempre são gerados os mesmos números.

Carregamento dos arquivos: a própria biblioteca *Numpy* possui métodos para carregamento de arquivos em diversos formatos como *csv*, *txt*, entre outros. O método utilizado carrega um arquivo texto de qualquer formato separado por vírgulas.

Criação do modelo: parte onde são definidas as características do modelo a ser criado, camadas da rede, métodos intermediários como ruído, entre outros. No caso, é um modelo sequencial cujas camadas e propriedades vão sendo adicionadas em sequência. Cada camada pode conter neurônios ou funções que alimentam a rede ou evitam alguns comportamentos. Cada camada possui o seguinte comportamento:

- A primeira camada adicionada é uma rede neural com 1441 neurônios, cada neurônio representa uma coluna do arquivo de entrada. Não é necessário que a primeira camada tenha o mesmo número de neurônios que a quantidade de entrada, o que define a quantidade de entrada é o argumento `input_dim` que precisa ser especificado apenas no início, a partir disso as outras camadas já terão como padrão de entrada o número de saídas das camadas anteriores. A função de ativação utilizada nas camadas é a função sigmoide, outras funções também podem ser utilizadas. `init=` define de que forma serão atribuídos os pesos aleatórios iniciais nos neurônios daquela camada;
- A camada de `Dropout` é inserida com o objetivo de reduzir a probabilidade de *overfitting*, que é o especialização excessiva da rede. Ela altera a primeira camada de neurônios, funcionando como uma propriedade da mesma. Seu funcionamento se dá na retirada aleatória de alguns neurônios a cada passo de treino e finaliza uma rede diferente para cada *batch*. Ao final, na hora do teste, junta-se essas redes diferentes compondo uma única e fazendo a média dos pesos. Isso evita que surjam relações complexas na rede;
- A terceira camada possui 1080 neurônios, seguindo os padrões anteriores. Seus neurônios são automaticamente ligados com a saída da primeira camada;

- Outro `Dropout` é adicionado à terceira camada.
- Como saída é adicionada uma camada de 718 neurônios, que representam o total de aplicações possíveis.

Início da contagem de tempo: parte onde se começa a contagem do tempo para aquisição dos dados de performance. Essa contagem começa antes do passo de compilação e treino. Isso se deve ao fato de que pode influenciar o tempo de compilação para placa de vídeo em relação ao tempo de compilação para o CPU. Se fosse considerado apenas o tempo de treino, o tempo de compilação poderia ter inviabilizado o tempo total, talvez fazendo com que a CPU fosse mais rápida.

Compilação do modelo: a compilação montará o modelo, e carregará o métodos de execução na placa de vídeo se for utiliza essa modalidade. Ela começa com a definição de qual método de otimização será utilizado na rede. No caso, o método escolhido foi o Stochastic Gradient Descent (SGD). Este método busca o mínimo local da função de otimização após cada iteração. No código é definida a taxa de aprendizagem de 0.1 (`lr=0.1`), com taxa de decaimento da taxa de aprendizagem (`decay=1e-6`) de 0.000001, momento, que multiplica o vetor velocidade na direção do gradiente (`momentum=0.9`), é de 0.9 e utiliza o método de Nesterov (`nesterov=True`), que primeiro acumula o vetor de velocidade na direção do gradiente para depois corrigir esse vetor [41]. Uma vez montado o método de otimização, a compilação do modelo também precisa da definição da função de perda, no caso foi definida para múltiplas classes (`loss='categorical_crossentropy'`). Essa definição é devida a natureza dos dados que possuem um vetor de valores binários e indicam qual aplicação foi realizada. Como última entrada para a compilação, mas sem a obrigatoriedade de defini-la, a medida de performance para o sistema adotada foi da acurácia (`metrics=['accuracy']`), ou seja, número de predições corretas sobre todas as predições. Há outros métodos que podem minimizar os falsos positivos mesmo que sacrifiquem a precisão, mas a natureza desse problema não tem forte impacto nesse aspecto, uma vez que, mesmo que não se ofereça o melhor investimento, oferecer algum investimento bom já é o suficiente.

Treino do modelo: nessa fase o modelo é treinado com os dados da fase de `Carregamento dos arquivos` (conjunto `X` e `Y` dos vetores de treino). No treinamento é definido o tamanho do *batch*, que é a quantidade de registros simulada para que haja uma atualização do gradiente. O *batch* escolhido foi 32 (`batch_size=32`), pois apresentou a melhor performance para o experimento. O *batch* influencia na performance, pois também é a taxa de carregamento da memória da placa de vídeo. Essa influência depende da arquitetura da máquina e tamanho do vetor simulado. *Batches* de

tamanho grande também degradam o modelo, fazendo com que perca a capacidade de generalização. Foram testados *batches* de 4 até 512, acima disso a degradação é perceptível [42]. Defini-se época quando todos os registros foram usados para atualizar o modelo. No caso foram definidas 8 épocas (`nb_epoch=8`), pois apresentaram a melhor taxa de performance.

Fim da contagem do tempo: finaliza-se a contagem antes da fase de avaliação, pois ela ocorre da mesma forma, mesmo que o treino tenha sido executado na placa de vídeo.

Avaliação do modelo: são testados os mesmos vetores de entrada para verificação da acurácia total. Também pode ser utilizada para prever validar o modelo com um conjunto de testes ou verificar qual é o resultado da rede para uma entrada específica.

Matriz de confusão: função onde se retorna a matriz de confusão para determinados dados e modelo. Essa função prediz qual o resultado da computação de uma entrada a partir da rede neural criada pelo comando `model.predict(X)`. Em seguida cria a matriz de confusão com valores zerados. A função `numpy.argmax` retorna uma tabela com a posição do maior valor para cada linha (essa configuração é passada em `axis=1`, do contrário retornaria a maior posição da tabela inteira). Antes de retornar a matriz de confusão, alimenta cada célula que esteja com a posição do maior argumento previsto e do resultado real.

Apresentação da matriz de confusão: área onde se transforma em escala logarítmica a matriz de confusão e apresenta o gráfico criado. Essa característica melhora a apresentação do gráfico, pois reduz a distância entre os resultados, facilitando para que dados que foram pouco testados também sejam apresentados, porém com coloração mais opaca.

3.4.1 Implementação com Múltiplos Tipos de Entrada

É possível através do *Keras* fazer a combinação de diferentes tipos de entrada em uma mesma rede neural desde que tenham a mesma saída. Essa metodologia é conhecida como *Merge* e pode ser utilizada para combinar entradas de texto com imagens, por exemplo, todas na mesma rede. Também pode ser utilizada para reduzir o grau de importância de uma determinada classificação que não se deseje enfatizar. Através do tratamento dos dados, separando a rede em duas, uma com todos os dados e outra retirando os dados relativos as classes que se deseje excluir, é possível diminuir o grau de influência dos dados menos desejados no resultado final. Essa técnica pode ser vista no exemplo de código:

Exemplo de código 2:

```
# Separacao dos dados
Xoutros = numpy.zeros((X.shape[0],X.shape[1]), dtype=numpy.float32)

for actual_line in range(Y.shape[0]):
    if  Y[actual_line, 3] != 1:
        Xoutros[actual_line] = X[actual_line]

# Criacao do modelo
model1 = Sequential()
model1.add(Dense(1441, input_dim=1441, activation='relu'))
model1.add(Dropout(0.5))

model2 = Sequential()
model2.add(Dense(1441, input_dim=1441, activation='relu'))
model2.add(Dropout(0.5))

model3 = Sequential()
model3.add(Merge([model1, model2], mode='concat'))
model3.add(Dense(1080, activation='relu'))
model3.add(Dropout(0.5))
model3.add(Dense(718, activation='softmax'))

# Compilacao do modelo
model3.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['
    accuracy'])

# Treino do modelo
model3.fit([X, Xoutros], Y, nb_epoch=8, batch_size=32, show_accuracy=False)
```

As características desse código são:

Separação dos dados: área onde se criará uma matriz de dados apartados, que possuam a característica de, quando aparecer a ocorrência do dado indesejado, se grave uma linha de registros vazios na matriz, para qualquer outro caso se grave a linha equivalente à matriz original. No caso, o registro menos desejado é da classificação três. Caso esse registro tenha muita influência sobre o conjunto de treino e isso não seja uma característica desejada, é recomendável que se diminua essa influência produzindo duas entradas, uma com ele e outra sem.

Criação do modelo: nessa fase o modelo segue o exemplo de código anterior, porém cada modelo de entrada é integrado em um único modelo através da camada de

`Merge`. No caso, o primeiro modelo é integrado com o segundo através da camada de `Merge` que faz parte do terceiro modelo.

Compilação do modelo: essa fase segue a implementação do exemplo anterior, o modelo todo é visto como uma grande rede neural, por isso, mesmo que monte um modelo diferente, as métricas anteriores para a montagem desse modelo permanecem inalteradas.

Treino do modelo: nessa fase que o treinamento se diferencias do exemplo anterior. Devem ser consideradas as duas entradas, uma para o primeiro modelo e outra para o segundo. Como os dados foram alterados, o primeiro e segundo modelos serão treinados diferentemente. Como o segundo modelo possui entradas zero para o treino da classificação três, seus pesos não são alterados para os registros dessa classificação, o que anula a importâncias desse registros nessa rede.

3.4.2 Implementação de Ponderação no Treino

O *Keras* permite associar aos vetores de entrada a ponderação na atualização de rede. Isso significa que um vetor pode ser mais ou menos importante na atualização dos pesos da rede conforme parâmetros passados na forma de *array* na entrada do treino (função `fit` no programa), onde os registros variam entre 0 e 1 para cada vetor. Essa técnica pode ser utilizada quando o conjunto de treino não reflete com exatidão as proporções do conjunto de teste ou há um desbalanceamento nos dados e pode ser vista no exemplo de código:

Exemplo de código 3:

```
def pesos(Y, weigths):
    positions = numpy.argmax(Y, axis=1)
    sample_weight = numpy.ones(Y.shape[0], dtype=numpy.float32)
    for actual_line in range(Y.shape[0]):
        sample_weight[actual_line] = 1 / weigths[positions[actual_line]]
    return sample_weight

X = numpy.loadtxt("PRDLINHATREINO.TXT", delimiter=",")
Y = numpy.loadtxt("APLLINHATREINO.TXT", delimiter=",")
peso_aplic = numpy.loadtxt("PESOAPLIC.TXT", delimiter=",")
sample_weights=pesos(Y, peso_aplic)
```

No exemplo acima, é criado um *array*, através da função `pesos`, contendo os inversos da frequência de cada aplicação recebidos através do arquivo `PESOAPLIC.TXT`. Esse `array`

será utilizado como peso associado a cada vetor durante o treino da rede, como mostrado no exemplo de código abaixo:

Exemplo de código 4:

```
model.compile(loss=tipo, optimizer='sgd', metrics=['accuracy'],
    sample_weight_mode=None)
model.fit(X, Y, nb_epoch=32, batch_size=32,show_accuracy=False,
    sample_weight=sample_weights)
```

Nesse exemplo, a compilação discrimina que há um arquivo de pesos através do texto `sample_weight_mode=None`. Em seguida, no treino, é passado o *array* como ponderação a ser utilizada através do texto `sample_weight=sample_weights`. Dessa forma é possível determinar o quanto uma rede deve aprender de cada exemplo.

3.4.3 Performance

O tamanho da entrada `batch_size` no treino da rede define qual é a taxa de carregamento dos dados antes do *backpropagation*. Essa entrada possui relação direta com a performance da máquina. O valor da época (`nb_epoch`) também foi alterado para garantir que houvesse atualizações suficientes conforme o *batch* crescia. O resultado desses testes pode ser visto na Tabela 3.1.

Tabela 3.1: Custo de Tempo e Acurácia por Batch/Época.

Batch	Época	Tempo GPU	Tempo CPU	Acurácia GPU	Acurácia CPU
512	512	1660s	16284s	77,71%	77,71%
256	256	1227s	9428s	77,71%	77,71%
128	128	718s	5847s	77,71%	77,71%
64	64	577s	4331s	77,71%	77,71%
32	32	487s	3721s	77,71%	77,71%
16	16	415s	3432s	77,71%	77,71%
8	8	357s	4102s	77,65%	15,98%
4	4	381s	4410s	0,65%	0,01%

A partir dos dados foram feitas comparações para estimar onde há o maior ganho do processamento da placa de vídeo em relação a CPU, com o objetivo de tornar o treinamento mais rápido, mantendo a acurácia do modelo. A Figura 3.4 mostra a relação entre as duas unidades. Essa relação de ganho está expressa no gráfico como a linha amarela. Os valores do eixo inferior (tamanho *Batch*) foram expressos em escala logarítmica para facilitar a visualização. Pode-se observar que o maior ganho está para valores menores do *Batch* e da Época. Porém a precisão do modelo decai muito para tais valores.

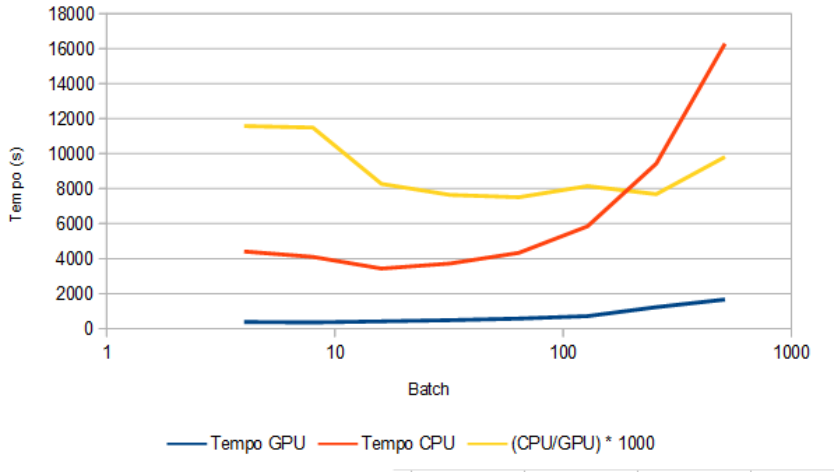

Figura 3.4: Comparativo do ganho entre GPU e CPU.

Devido a perda da acurácia para valores baixos, foi necessário definir uma taxa de performance para cada unidade de processamento. Tal taxa era necessária para se verificar onde há real ganho de custo de processamento em relação a acurácia obtida. Para seu cálculo foi definido como performance a Acurácia sobre o Tempo de cada unidade. O cálculo foi multiplicado por 1000 para que houvesse acerto da escala de ambos. Os valores do Batch foram expressos em escala logarítmica para facilitar a visualização. O resultado pode ser visto na Figura 3.5

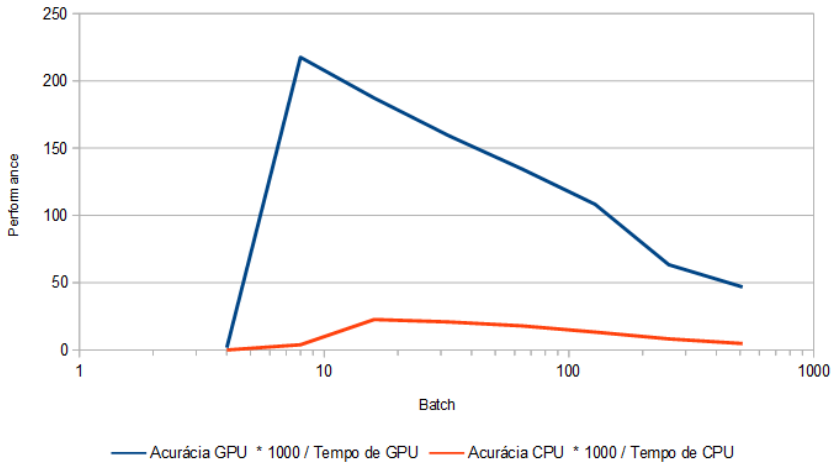

Figura 3.5: Comparativo da performance do treino entre GPU e CPU.

A região que compreendia o *Batch* de tamanho 8 até 64 se destaca em relação às demais em termos de performance. Dessa forma considerou-se encontrar um máximo local nessa

região. A Tabela 3.2 expressa os valores obtidos para a região entre *Batch* 8 e 64 e época 8 e 32.

Tabela 3.2: Máximo Local de Performance.

———	**Batch 8**	**Batch 8**	**Batch 16**	**Batch 32**
Época 8	77,65%/357s	77,71%/227s	77,71%/124s	77,71%/556s
Época 16	0,78%/8536s	77,71%/415s	77,71%/1905s	77,71%/1138s
Época 32	10,95%/17510s	77,71%/7386s	77,71%/487s	77,71%/2262s

Esta tabela pode ser traduzida no gráfico da Figura 3.6, onde é apresentada a performance local para cada valor combinação de *Batch* e época.

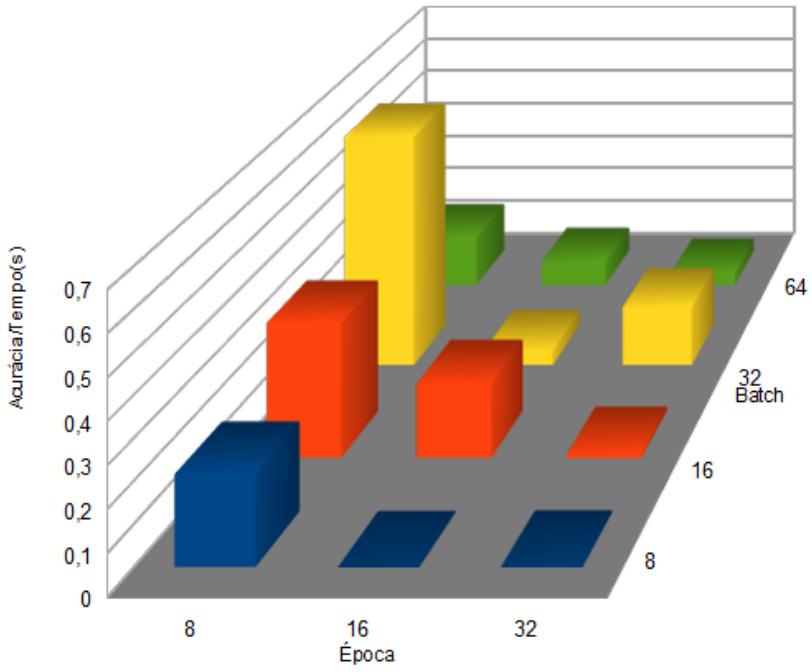

Figura 3.6: Máximo Local para GPU.

Da gráfico obtém-se um máximo local para quantidade de épocas 8 e *Batch* 32. Dessa forma esses valores foram utilizados na nos treinos subsequentes da rede.

3.5 Comentários Finais

Na camada de `Dropout`, a probabilidade escolhida foi de 0.5 (50%) na retirada de cada neurônio, o que maximiza o resultado dessa camada [43]. Valores diferentes apresentaram diminuição da efetividade.

Os testes iniciais de performance foram realizados com a relação entre *batch* e época iguais. Isso se deve pois a rede só realiza o *backpropagation* (atualização dos pesos da rede) após o término de cada *batch*. Assim, para comparar a performance entre um valor de *batch* e outro, era necessário manter a quantidade de atualizações da rede constante. Portanto, quando se divide a quantidade de vetores pelo tamanho do *batch* (atualizações da rede por época) e depois se multiplica pela quantidade de épocas, ao final a quantidade de atualizações na rede permanece a mesma se o valor do *batch* e época são iguais.

Capítulo 4

Resultados Obtidos

Diversos cenários foram considerados para teste das soluções possíveis. Tais cenários podem ser classificados em quatro casos de teste principais, alguns com ramificações, mas seguindo a mesma ideia. Os casos foram considerados conforme o objetivo de cada teste, que é a verificação das relações entre os dados e as acurácias da RNA.

4.1 Caso 1

No primeiro caso foram analisados os dados *in natura*, com o objetivo de verificar o comportamento deles e a complexidade do problema. Uma rede simples de três camadas foi criada seguindo o modelo do primeiro exemplo de código. Cada rede composta de 1441 neurônios de entrada e 718 neurônios na camada de saída, conforme representação da Figura 4.1. O número de neurônios da camada escondida, variava para cada sequência de experimentos. Foi inicialmente estimado para, a primeira sequência, a metade da média geométrica dos números de neurônios das camadas de entrada e saída, ou seja 509. Em seguida foi repetido para os valores de 1018 neurônios, e 2160. Houve 10 experimentos para cada sequência. Foi utilizada a função de ativação sigmoide como saída de todas as camadas, assim como o otimizador SGD.

A média dos resultados dos experimentos pode ser vista nas tabelas Tabela 4.1 de treino e Tabela 4.2 de teste, respectivamente.

Tabela 4.1: Médias obtidas nos treinos.

Número Neurônios	**Acurácia**	**Desvio Padrão**
509	74,1742793902%	0,9996270828
1018	76,3661092299%	0,315004635
2160	76,5626068094%	0,4799811332

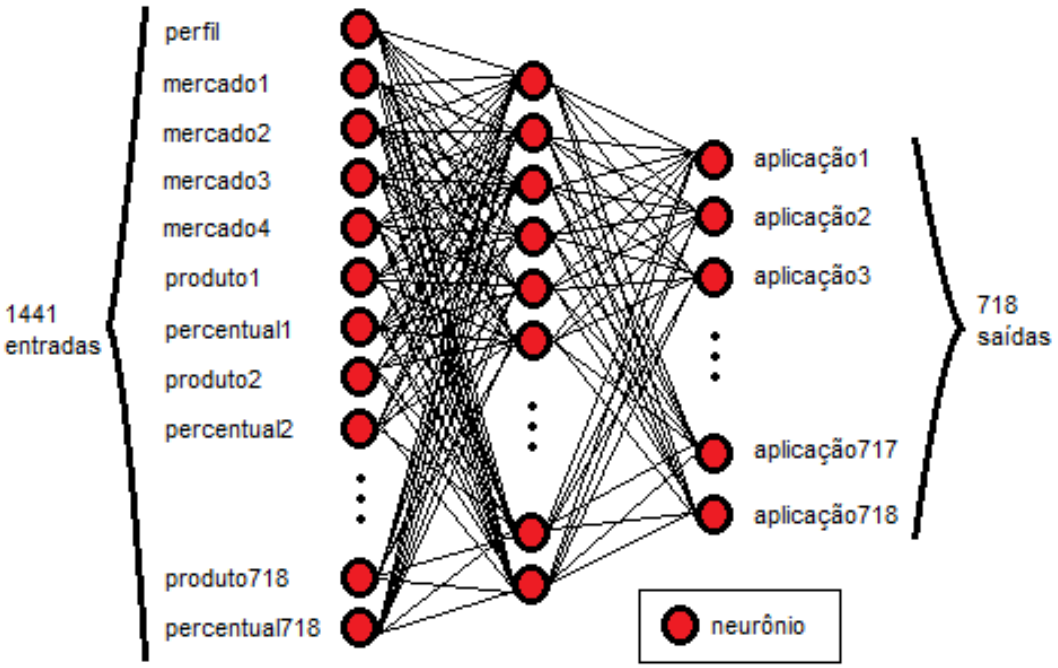

Figura 4.1: Representação da rede neural proposta.

Tabela 4.2: Médias obtidas nos testes.

Número Neurônios	Acurácia	Desvio Padrão
509	62,8068870951%	0,6560415913
1018	63,4107710288%	0,3034145608
2160	63,8451716164%	0,4531064671

Uma acurácia média de 76% foi obtida no treino de redes com 2160 neurônios na camada intermediária. Os resultados obtidos mostram que a rede sempre retorna a aplicação de maior frequência como resultado da simulação, através da observação da distribuição dos valores na matriz de confusão, que estava concentrada no produto três, conforme Figura 4.2.

Para esse caso, a rede neural aprendeu a aplicar apenas no produto com maior quantidade de aplicações, devido ao forte desbalanceamento dos dados.

4.2 Caso 2

No segundo caso foi verificado o ganho de acurácia ao se inserir duas camadas de *Dropout*, antes e depois da camada intermediária. Esse caso verifica se há *overfitting* no treino do caso 1, seguindo os mesmos padrões do caso anterior. Foram definidos 1441 neurônios de entrada da rede, 718 neurônios na camada de saída e as camadas intermediária variando

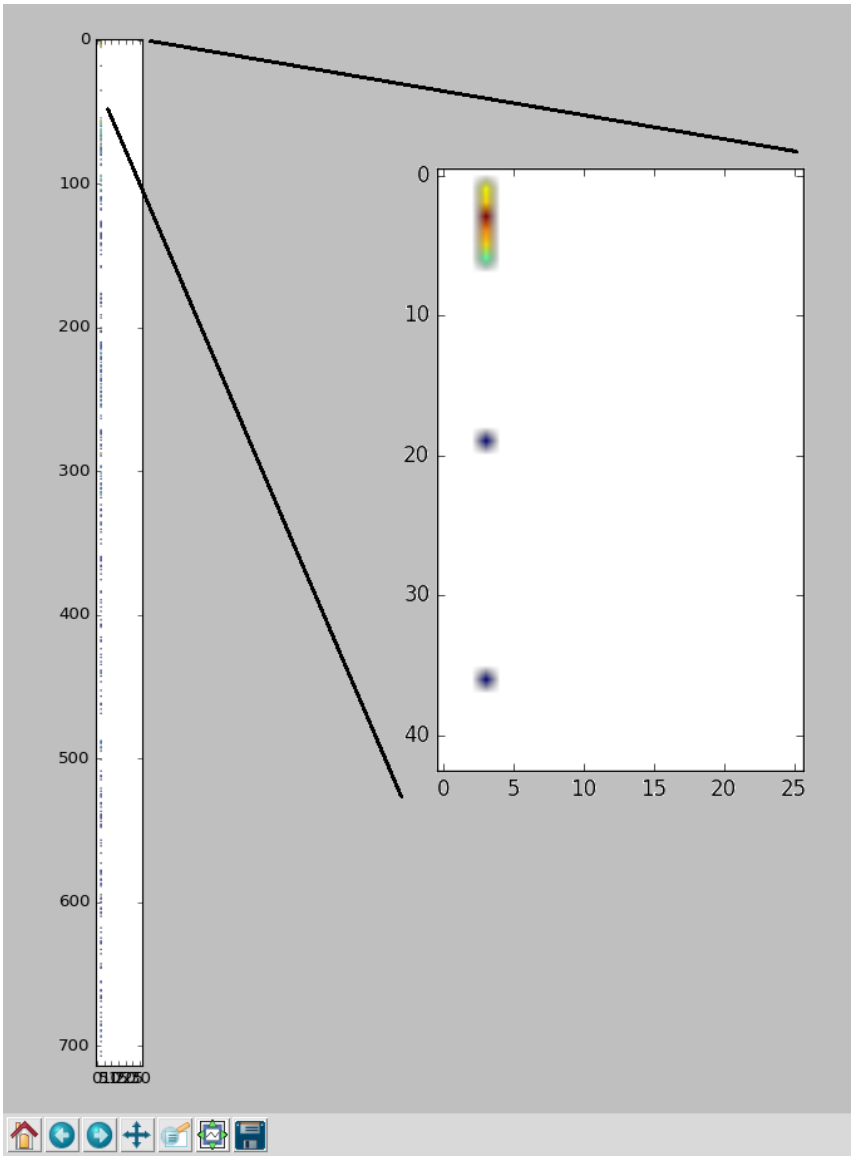

Figura 4.2: Matriz de Confusão para sequência 1 do caso 1.

conforme os valores: 509, 1018 e 2160. Todas as camadas possuíam a função de ativação Sigmoide e otimizador SGD. Foram realizados 10 treinos e testes para cada rede criada. A média dos resultados obtidos no treino e testes seguem das tabelas Tabela 4.3 e Tabela 4.4 respectivamente

Mesmo com o risco de *overfitting* diminuído pela camada *Dropout*, o resultado foi similar ao caso anterior. O que demonstra que a rede não está sendo comprometida com o excesso de treino.

Tabela 4.3: Médias obtidas nos treinos.

Número Neurônios	Acurácia	Desvio Padrão
509	74,2317729815%	0,9666232487
1018	76,7764952485%	0,3775325222
2160	77,0728022147%	0,2227642629

Tabela 4.4: Médias obtidas nos testes.

Número Neurônios	Acurácia	Desvio Padrão
509	62,7197182108%	0,6513984284
1018	64,3329854886%	0,4174406436
2160	64,5017429277%	0,5023099111

4.3 Caso 3

No terceiro caso foi verificada se a adição de mais uma camada intermediária afetaria a acurácia da rede neural. Esse caso tem por objetivo eliminar a possibilidade de uma rede com apenas uma camada intermediária ter a capacidade limitada por falta de complexidade. Foi criada uma rede com 1441 neurônios de entrada, 718 neurônios na camada de saída. As duas camadas intermediárias tiveram os valores 1018 e 855 para a primeira sequência de experimentos, 2160 e 1710 para a segunda sequência. Todas as camadas possuíam a função de ativação Sigmoide e otimizador SGD. Foram realizados 10 testes para cada configuração. Os resultados dos testes e treinos podem ser vistos nas tabelas Tabela 4.5 e Tabela 4.6, nesta sequência.

Tabela 4.5: Médias obtidas nos treinos.

Número Neurônios	Acurácia	Desvio Padrão
1018/855	76,2476298423%	0,8017921409
2160/1710	76,8918189931%	0,5707737024

Não foi observado ganho com a utilização da segunda camada intermediária (oculta) de neurônios, o que demonstra que a complexidade do problema não está limitada pelo número de camadas ou quantidade de neurônios, para o caso simples com a função de ativação Sigmoide.

4.4 Caso 4

No quarto caso foi verificado se havia a ocorrência de *overfitting* para as redes com duas camadas intermediárias de neurônios do caso 3. Foi feita a adição da camada de *Dropout* intercalando cada camada das redes montadas no caso anterior, exceto na última camada.

Tabela 4.6: Médias obtidas nos testes.

Número Neurônios	Acurácia	Desvio Padrão
1018/855	64,4798845754%	0,2538593579
2160/1710	64,2202861354%	0,4373218226

Foi adotada essa estratégia a fim de detectar se a performance da adição da segunda camada intermediária estava limitada pelo excesso de dados. Seguindo os moldes do caso anterior, foi criada uma rede com 1441 neurônios de entrada e 718 neurônios na saída. Foram executados 10 experimentos com camadas intermediarias de 1018 e 855 neurônios, assim como com as camadas intermediárias de 2160 e 1710. Todas as camadas possuíam a função de ativação Sigmoide e otimizador SGD. Os resultados dos testes e treinos podem ser vistos nas tabelas Tabela 4.7 e Tabela 4.8, respectivamente.

Tabela 4.7: Médias obtidas nos treinos.

Número Neurônios	Acurácia	Desvio Padrão
1018/855	76,9859987349%	0,3477799235
2160/1710	77,0859987349%	0,3008324349

Tabela 4.8: Médias obtidas nos testes.

Número Neurônios	Acurácia	Desvio Padrão
1018/855	64,255902711%	0,2143537191
2160/1710	64,679514995%	0,407152604

Com a introdução da camada de *Dropout* pode-se comprovar que o excesso de treino também não afeta o resultado geral da rede com mais camadas intermediárias de neurônios.

4.5 Caso 5

No quinto caso verificou-se se há influência da função de ativação em relação ao resultado obtido no primeiro caso. Desse modo foram feitos os mesmos experimentos com funções de ativação diferentes da Sigmoide. Foi criada uma rede com uma camada de entrada como 1441 neurônios e função de ativação *relu* (Unidade Linear Retificada) e uma camada de saída com 718 neurônios e função de ativação *softmax* (Função Exponencial Normalizada). Para a camada intermediária foram abordadas as configurações de 509, 1018, 2160 e 4320 neurônios. Estas camadas possuíam a função de ativação *relu*. Foram realizados 10 experimentos para cada configuração da rede. Os dados obtidos para os treinos seguem da Tabela 4.9 e os dados obtidos para os testes seguem da Tabela 4.10.

Tabela 4.9: Médias obtidas nos treinos.

Número Neurônios	Acurácia	Desvio Padrão
509	77,240134358%	0,373972571
1018	77,6197101953%	0,3031746375
2160	77,666229387%	0,2273467473
4320	77,7399035705%	0,5595653442

Tabela 4.10: Médias obtidas nos testes.

Número Neurônios	Acurácia	Desvio Padrão
509	64,1263586957%	0,0440500595
1018	64,0387228261%	0,2048396109
2160	64,0754076087%	0,1046308381
4320	64,1257310844%	0,4369018492

Em termos semânticos, obteve-se resultados diferentes dos apresentados anteriormente. O resultado da Matriz de Confusão destes testes possui um comportamento mais próximo da diagonal da matriz para várias aplicações, como pode ser visto em parte na Figura 4.3. Esse comportamento é mais desejado para solução do problema, uma vez que, apresenta diversidade nas aplicação sugestionadas.

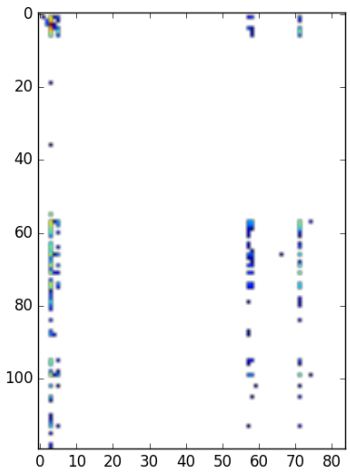

Figura 4.3: Matriz de Confusão para sequência 2 do caso 1.

4.6 Caso 6

No sexto caso, verificou-se a influência do uso de *Dropout* em relação ao resultado obtido no quinto caso. Desse modo, procura-se mitigar a possibilidade de *overfitting* no treino do

caso anterior. As camadas de *Dropout* foram intercaladas com cada camada de neurônios da rede, exceto a última. Foi criada uma rede com uma camada de entrada como 1441 neurônios e função de ativação *relu* e uma camada de saída com 718 neurônios e função de ativação *softmax*. Para a camada intermediária foram abordadas as configurações de 509, 1018, 2160 e 4320 neurônios. Estas camadas possuíam a função de ativação *relu*. Foram realizados 10 experimentos para cada configuração da rede. Os dados obtidos para os treinos seguem da Tabela 4.11 e os dados obtidos para os testes seguem da Tabela 4.12.

Tabela 4.11: Médias obtidas nos treinos.

Número Neurônios	**Acurácia**	**Desvio Padrão**
509	77,1547623035%	0,5481313645
1018	77,4118682943%	0,0527814483
2160	77,4055634668%	0,0593409042
4320	77,4026886434%	0,33706165

Tabela 4.12: Médias obtidas nos testes.

Número Neurônios	**Acurácia**	**Desvio Padrão**
509	64,089673913%	0,0323434259
1018	64,0774456522%	0,0436524665
2160	64,0862771739%	0,0467660245
4320	64,1408481451%	0,3553075878

Em comparação com os dados obtidos no quinto caso, não houve melhora na acurácia, de modo que o excesso de dados não afeta o treino da rede neural.

4.7 Caso 7

No sétimo caso, verificou-se o efeito da mudança da função de ativação para o caso 4, onde a rede possui duas camadas intermediárias intercaladas com as camadas de *Dropout*. Com a melhora na interpretação da rede para as novas funções de ativação, foi necessário verificar se a camada adicional e as camadas de *Dropout* representam ganho na acurácia. Foi criada uma rede com 1441 neurônios de entrada com função de ativação *relu* e 718 neurônios na saída com função *softmax*. Foram executados 10 experimentos com camadas intermediarias de 1018 e 855 neurônios, com as camadas intermediárias 2160 e 1710 e com camadas . Essas camadas possuíam a função de ativação *relu*. Intercalando cada camada de neurônios havia uma camada de *Dropout*. O otimizador utilizado foi o SGD. Os resultados dos testes e treinos podem ser vistos nas tabelas Tabela 4.13 e Tabela 4.14, respectivamente.

Tabela 4.13: Médias obtidas nos treinos.

Número Neurônios	Acurácia	Desvio Padrão
1018/855	77,1249327128%	0,5299181481
2160/1710	77,3890404704%	0,0667411203
4320/3420	77,3278459331%	0,3971180931

Tabela 4.14: Médias obtidas nos testes.

Número Neurônios	Acurácia	Desvio Padrão
1018/855	64,0528449477%	0,4452614579
2160/1710	64,078125%	0,0726086923
4320/3420	64,2791631728%	0,353863014

Para o sétimo caso houve diminuição da variância em relação ao quarto caso. Mesmo que não houvesse melhora na acurácia, a precisão das previsões melhorou. Em termos semânticos, foi detectada maior distribuição de valores ao longo da diagonal inversa da Matriz de Confusão, fato que não foi observado no caso 4.

4.8 Caso 8

Conforme observado no caso 1, as quantidades de investimentos em determinados produtos tendenciaram a rede a prever as aplicações com maiores frequências. Com o objetivo de mitigar a influência de tais aplicações, foram feitos testes que limitavam a quantidade de amostras por aplicação, de modo que fosse compensado o desbalanceamento dos dados. Os dados foram divididos nos conjuntos de no máximo um exemplar por aplicação, máximo de 10 exemplares por aplicação, máximo de 100 exemplares por aplicação, máximo de 1000 exemplares por aplicação e máximo de 10000 exemplares por aplicação. Para cada conjunto foram realizados 10 experimentos. Os experimentos foram testados 3 vezes, uma com o conjunto de treino, outra com o conjunto de treino original, envolvendo todos os registros, e outra com o conjunto de teste. A rede criada segue os padrões do caso 5, para uma camada intermediária de 2160 neurônios com função de ativação *relu*, uma camada de entrada de 1441 neurônios com função de ativação *relu* e uma camada de saída de 718 neurônios com função de ativação *softmax*. Os resultados dos testes, testes originais e treinos podem ser vistos nas tabelas Tabela 4.15, Tabela 4.16 e Tabela 4.17, respectivamente.

Para o oitavo caso, pode ser observado como a distribuição afetou a acurácia. Devido a discrepância entre as quantidades de aplicações, a rede não consegue decidir corretamente quando se limita essas quantidades. Isso se deve pelas redes, com menor número de vetores de treino, se tornarem altamente especializadas, o que explica os altos índices de acerto

Tabela 4.15: Médias obtidas nos treinos.

Máximo por aplicação	Acurácia	Desvio Padrão
1	83,922481133%	0,1723176367
10	70,7044409817%	0,5410113557
100	56,8742731081%	0,4622183248
1000	46,2728126387%	1,9249984877
10000	48,9640003721%	2,8279390045

Tabela 4.16: Médias obtidas nos treinos originais.

Máximo por aplicação	Acurácia	Desvio Padrão
1	0,8883617887%	0,1522360327
10	5,3828420286%	0,8847037309
100	5,0655351661%	0,3467904198
1000	13,4054511329%	0,8494711622
10000	70,6459547423%	0,7245067437

no conjunto de treino, mas a deficiência de acerto no conjunto original de treino ou de teste. Porém, pode-se notar que há uma dispersão maior dos dados no gráfico de matriz de confusão o que implica em uma maior distribuição em diferentes tipos de aplicação antes sequer considerados como resultado das simulações. Mesmo assim a manipulação do desbalanceamento não contribuiu com a acurácia da rede.

4.9 Caso 9

Com o objetivo de utilizar as dispersões das simulações obtidas no caso anterior e compensar o desbalanceamento das aplicações, sem alterar o arquivo de entrada, foi criada uma rede composta por cinco redes de entrada, onde cada rede é treinada separadamente. Cada rede se especializa em um conjunto de dados como no caso anterior. Após os treinamentos, essas redes são congeladas (seus pesos não são mais atualizados) e servirão de entrada de uma rede composta.

Tabela 4.17: Médias obtidas nos testes.

Máximo por aplicação	Acurácia	Desvio Padrão
1	0,6546202286%	0,1751029808
10	4,0908868541%	0,7042348196
100	3,930277219%	0,2883034289
1000	8,3335410747%	0,4032775557
10000	54,2513346133%	1,4920139904

Para o treinamento das sub-redes, o arquivo de entrada é dividido em aplicações cujo critério para divisão é a frequência de aplicações. O critério de divisão dos arquivos é definido como: até 1 registro por aplicação, até 10 registros, até 100 registros, até 1000 registros e até 10000 registros, seguindo os moldes do exercício anterior. Dessa forma, os resultados do treino das sub-redes também seguem o exercício anterior.

Cada sub-rede possui 1441 neurônios de entrada, uma camada intermediária de 1018 neurônios e 718 neurônios na camada de saída. As funções de ativação são `relu` para a camada de entrada, `relu` para a camada intermediária e `softmax` para a saída. Esta configuração foi adotada seguindo o modelo do caso 5, por possuir uma boa dispersão das saídas. Após o treino, essas redes são congeladas, ou seja não ocorre mais sua atualização nos próximos treinos. Para a criação da rede que englobaria as outra foi criada uma camada intermediária de 3661 neurônios com função de ativação `relu` e uma camada de saída com 718 neurônios com função `softmax`.

Foram realizados 10 treinamentos para cada sub-rede. Os dados obtidos seguem da Tabela 4.18.

Tabela 4.18: Acurácia para Experimento 1 do Caso 9.

	Acurácia	**Desvio Padrão**
Treino	76,0756058196%	0,7625445302
Teste	63,9684410165%	1,0117504575

Para o nono caso, pode ser observado que a subdivisão em redes menores, pré-treinadas, não representou ganho na acurácia da rede. Dessa forma, compensar o desbalanceamento dos dados através de especialização, não apresenta ganho para o problema proposto, no caso da utilização de sub-redes congeladas. De fato, a rede voltou aos estados iniciais dos primeiros experimentos, onde sempre se tendia a aplicar no mesmo produto. Este experimento revela que o congelamento das sub-redes não é uma abordagem válida, ao menos sem garantir que a rede maior também tenha acesso aos dados integrais de treino.

4.10 Caso 10

Para o décimo caso, foi realizada uma abordagem alternativa para o problema. A abordagem consiste na modificação da entrada em diversas outras. Tais entradas servem de insumo para diversas redes que ao final são agrupadas. As modificações consistem em substituir as linhas da entrada cujas aplicações sejam as mais frequentes por zeros. Dessa forma, as entradas tem o mesmo número de linhas entre si, porém somente as linhas que pertencem às aplicações menos frequentes são preenchidas. Cada entrada é processada

por uma rede. As redes são integradas em uma saída através de uma camada de integração (*merge*).

Essa metodologia tem o objetivo de introduzir o cálculo do erro na parte da rede que possua os produtos com mais aplicações, sem atualizar a parte da rede dos produtos com menos. Isso ocorre devido ao momento em que se está processando os produtos com mais frequência, a entrada da rede dos produtos com menos estar zerada. Dessa forma, não é calculado o erro para essa parte da rede e os valores dos pesos não são atualizados.

Ao se introduzir o erro na parte da rede com os produtos mais frequentes, tem-se o objetivo de diminuir sua importância na classificação das aplicações, melhorando o desempenho.

A rede foi composta por três entradas, uma com todos os registros, uma com as linhas cujas aplicações do produto mais aplicado foi substituída por zeros e uma com as linhas cujas aplicações dos oito produtos mais aplicados foram substituídas por zeros. Cada entrada possui uma rede de 1441 neurônios associados, uma camada de integração (*merge*) por concatenação das redes, uma camada intermediária de neurônios e uma camada de saída de 718 neurônios. A quantidade de neurônios da camada intermediárias foi alterada a cada experimento. A representação da rede pode ser vista na Figura 4.4. A camada intermediária varia de tamanho conforme os valores: 509, 1018, 2160 e 4320. As funções de ativação são *relu* para todas as camadas, exceto a última cuja função é a *softmax*. A Tabela 4.19 e a Tabela 4.20 possuem os valores respectivos para o treino e teste da rede.

Tabela 4.19: Médias obtidas nos treinos.

Número Neurônios	**Acurácia**	**Desvio Padrão**
509	81,1987829593%	1,2565246033
1018	85,1237860079%	0,2762605541
2160	85,6916287109%	0,323275976
4320	85,8882604111%	0,45974187

Tabela 4.20: Médias obtidas nos testes.

Número Neurônios	**Acurácia**	**Desvio Padrão**
509	70,0842277859%	0,7883280064
1018	71,3561963583%	0,4804959367
2160	71,9252717391%	0,3482934144
4320	72,011576087%	0,3816440451

No décimo caso, pode-se observar como o método de separação das entradas é mais eficiente no tratamento de entradas desbalanceadas. Esse método compensa as distâncias nas quantidades dos dados sem comprometer os mesmos, o que demonstra um forte acoplamento entre as aplicações realizadas e a acurácia do modelo. Isolar as aplicações

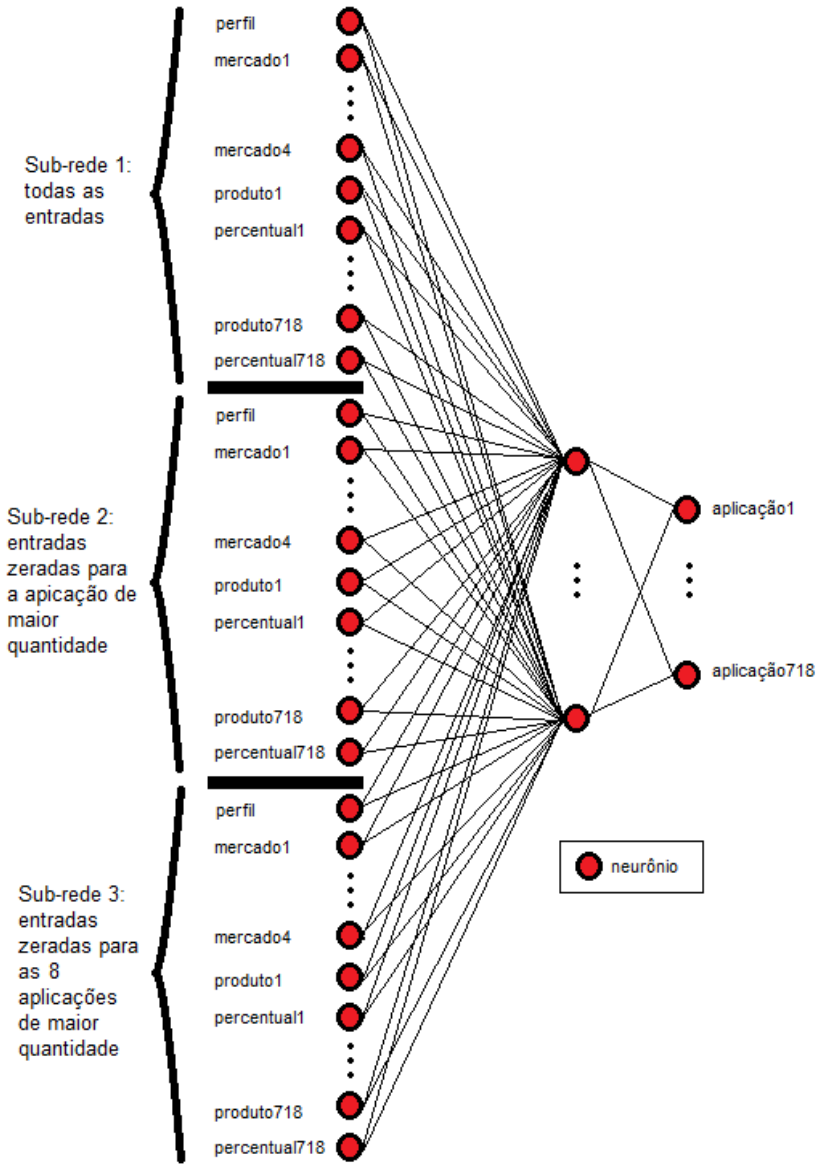

Figura 4.4: Representação da Rede Neural do Caso 10.

por ocorrências é um avanço significativo para a interpretação da rede. A Figura 4.5 demonstra como há uma dispersão mais acentuada das aplicações na Matriz de Confusão, o que significa a melhora na interpretação da rede.

4.11 Caso 11

Para o décimo primeiro caso, foi realizada a mesma abordagem do caso 10, porém utilizando o conceito de ponderação no aprendizado da rede. Nesse conceito, a atualização dos pesos da rede depende do peso associado ao vetor de treino. Cada vetor possui o peso

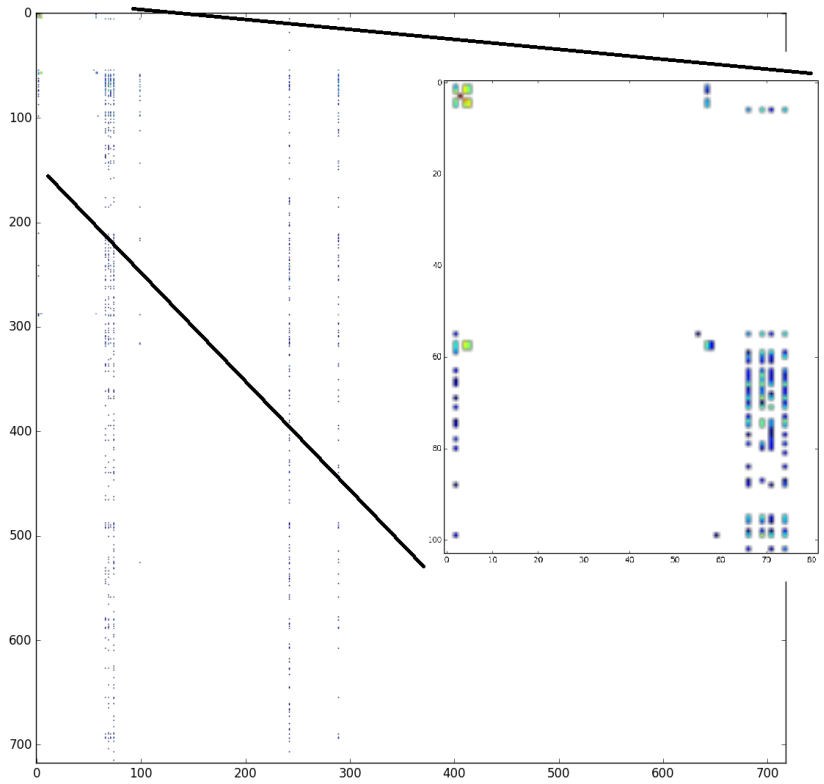

Figura 4.5: Matriz de Confusão para o experimento do caso 10.

inversamente proporcional à frequência da aplicação referente ao mesmo. Por exemplo, se o vetor é referente a uma aplicação que aparece 50 vezes no conjunto de treino, então o peso associado ao vetor é 1/50. O objetivo deste caso é verificar se a influência das aplicações mais frequentes pode ser reduzida ainda mais, aumentando a acurácia da rede. A rede criada possuía 2160 na camada intermediária com função de ativação *relu*. As outras características eram idênticas a do caso 10. Foram executados 10 testes para essa rede. Os dados de treino e teste seguem das tabelas Tabela 4.21 e Tabela 4.22 respectivamente.

Tabela 4.21: Média obtida no treino.

Acurácia	**Desvio Padrão**
0,1048992858%	0,0659057223

Tabela 4.22: Média obtida no teste.

Acurácia	**Desvio Padrão**
0,0136417271%	0,0040653478

No décimo primeiro caso, constatou-se que a introdução de pesos que compensassem a frequência das aplicações, igualando essa frequência através da taxa de aprendizagem, não

representou ganho para a rede. A rede passou a não aprender os casos mais frequentes, o que trouxe similaridade com os resultados apresentados no experimento com até uma aplicação do caso 8, ou seja, aplicações com até 1 ocorrência foram preteridas em relação as demais. Esse método foi equivalente a ter treinado a rede com um conjunto de dados diferente, o que ocasionou o baixo desempenho.

4.12 Validação da Rede Neural Artificial

O resultado geral dos experimentos, observando o melhor resultado para cada caso, segue da Figura 4.6. Esse resultado mostra a predominância do caso 10 ante os demais. Os

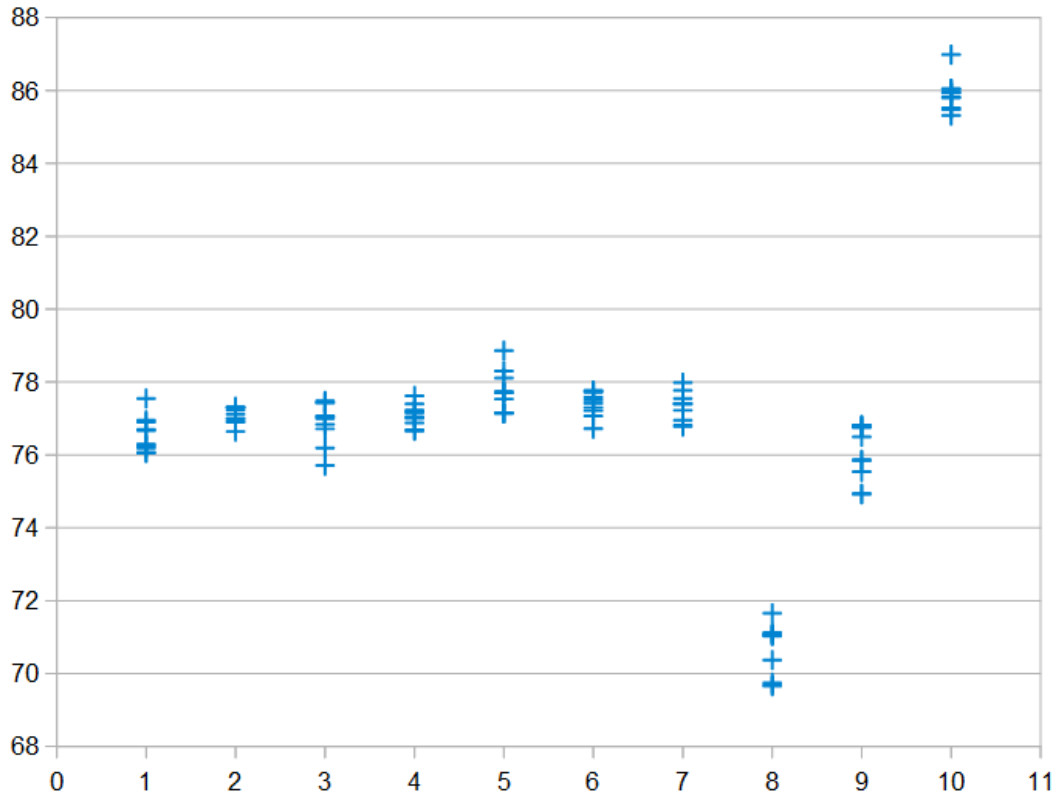

Figura 4.6: Comparação entre os resultados dos melhores experimentos de cada caso.

valores do caso 11 não são mostrados, pois distinguem-se muito da escala dos demais.

Através da simulação da rede neural do caso 10 foi possível a verificação da efetividade da mesma. Utilizando a rede previamente treinada para o caso e simulando-a com a base de testes, foi possível identificar quais produtos representaram ganho na rentabilidade das carteiras dos clientes. Em um primeiro passo foi calculado o valor máximo de aplicação do produto antes que a carteira fosse desenquadrada. Em seguida, o valor calculado foi adicionado a carteira e os percentuais de todos os produtos foram recalculados em

relação ao novo valor total da carteira. Cada percentual foi multiplicado pelo valor da rentabilidade do respectivo produto e os resultados das multiplicações foram somados.

Ao final, aproximadamente 61% do conjunto de testes tiveram ganho de rentabilidade, 4% se manteve com a mesma rentabilidade e 35% perdeu rentabilidade. Dessa forma, a rede melhorou ou manteve iguais os investimentos em 65% dos casos, ainda mantendo os clientes enquadrados.

4.13 Comparação com trabalhos Correlatos

Os trabalhos apresentados sobre o tema de investimentos vinculados a redes neurais artificiais, conforme exposto anteriormente, possuem proximidade com o abordado neste trabalho. Dentre as características abordadas nos trabalhos correlatos, pode-se verificar como predominantes:

- Concentram-se em áreas específicas de aplicações, que possuem características mais controladas para o treino da rede;
- As comparações de risco de produto utilizam o cálculo da teoria da Markowitz, que relaciona a covariância entre produtos para a formação de carteiras;
- Tentam deduzir qual a próxima melhor aplicação levando em consideração séries históricas dos valores de ativos.

As características tratadas também limitam o oferecimento de um bom produto, pois excluem uma série de fatores que são importantes para os investidores. Tais fatores podem ser elencados como:

- Oferecem produtos sem considerar a legislação vigente, no que tange à API;
- Consideram uma aplicação equivalente a outra sem incorporar os interesses do investidor para a aplicação em questão como: perfil de investimento, prazo de resgate, entre outros;
- Limitam as opções de investimento, pois apresentam quantidade reduzida de investimentos e de tipos de investimento.

Mesmo apresentando proximidade de temas, há uma grande distância entre a produção acadêmica atual e o tema deste trabalho. Tendo em vista as limitações do problema em questão e pelos fatores anteriormente apontados, o mesmo se tornou ímprobo de correlação.

4.14 Comentários Finais

As redes criadas tiveram sua acurácia medida para os dados dos melhores clientes enquadrados. Dessa forma, acertar o resultado da classificação para uma determinada entrada significa que a RNA oferecerá um produto baseada no que é mais comum nas aplicações dos melhores aplicadores, levando em consideração seu perfil de investidor. Isso não significa que, mesmo que tenha errado para algumas entradas, o produto oferecido não seja um bom investimento. Pode ocorrer a possibilidade de ser um produto melhor do que o conjunto de treino sugeriu. Devido a complexidade do segmento de investimentos, não há como afirmar se foi bom ou ruim até que seja exercido o resgate do mesmo.

Capítulo 5

Conclusão

Neste trabalho foi apresentado uma aplicação de Redes Neurais Artificiais (RNA) no oferecimento de produtos de investimento, através da análise do perfil do investidor e carteira de investimentos que possui.

O estudo é parte de uma solução para oferecimento, de forma automatizada, de um bom investimento, considerando a situação específica de um cliente. De forma geral, foi realizado o treino de uma RNA que possui como entrada o comportamento padrão do cliente em relação aos investimentos e como saída uma aplicação. Para o treino, foram considerados os melhores investidores de uma instituição financeira, de modo que a rede aprendesse a imitar o comportamento desses investidores, ofertando bons produtos de investimento. Outra preocupação na criação da rede foi considerar apenas investidores que estivessem com os investimentos aderentes ao seu perfil de investidor, o que traz segurança no investimento ofertado, uma vez que diminui a probabilidade de oferta de investimentos fora do padrão de consumo do cliente.

A simulação levou em conta os investimentos realizados no mês de setembro de 2016. Devido ao enorme fluxo de aplicações da instituição escolhida, não houve carência de dados. Para tratar do volume elevado, técnicas para otimizar a performance das redes, como a utilização de placas gráficas no processamento, foram utilizadas. Os dados apresentados possuíam grande desbalanceamento. Para compensar isso, diversas soluções de contorno tiveram de ser testadas. Bibliotecas pré-formatadas para trabalho com redes neurais através da linguagem de programação Python foram utilizadas, o que diminuiu o tempo de criação e possibilitando o teste dessas diversas redes.

A rede que apresentou o melhor resultado foi a rede do caso 10. Neste caso, a RNA é composta por 3 sub-redes. A primeira sub-rede contempla todo o conjunto de dados, a segunda possui entradas zeradas para a aplicação de maior frequência e a terceira com entradas zeradas para as 8 aplicações de maior frequência. A saída de cada sub-rede é ligada a uma rede maior. A saída da rede maior representa cada aplicação possível,

onde a que possui a maior probabilidade é a escolhida como resultado da simulação. Através da subdivisão em redes menores, que contenham entradas diferentes da original, é introduzido erro na rede sempre que as aplicações mais frequentes são feitas, o que diminui seu grau de importância. A validade da rede foi verificada através do resultado da simulação da mesma para cada cliente, o investimento simulado é aplicado até o limite antes do desenquadramento. Para a maioria dos casos a rentabilidade total das carteiras aumentou.

5.1 Trabalhos Futuros

Como possíveis trabalhos futuros, pode-se apontar:

- Implementação da utilização de mais variáveis como entrada da RNA e verificação do efeito no desempenho da mesma, assim como, verificar se o resultado é mais adequado ao esperado pelos clientes. Um exemplo de possíveis entradas que se pode elencar são as características da data em que a aplicação foi realizada, como o dia do mês ou o mês de aplicação. Esse exemplo pode melhorar a interpretação do comportamento de cada investidor, através da separação dos investidores esporádicos e profissionais, dado que investidores profissionais tendem a aplicar em dias diferentes dos investidores esporádicos, ou pela época em que um investimento tem maior probabilidade de ser adquirido. Outro exemplo de entrada que teria efeito na performance do modelo seria a classificação das perspectivas econômicas nacional e global. Esse exemplo seria importante para aplicações de longo prazo que podem ser mais afetadas por tais perspectivas.

- Verificação teórica da abordagem utilizada para compensar o desbalanceamento das aplicações utilizada no Caso 10. Verificar os motivos pelos quais a técnica de manipulação das entradas, sem alterar suas características, pode ser empregada na melhora da interpretação da rede.

- Implementação de outras técnicas de aprendizagem de máquina, tais como Maquinas de Vetores de Suporte e Árvores de Decisão, com objetivo de comparação dos resultados. Essa implementação pode identificar outras técnicas que possuam maior precisão e performance em comparação com as RNAs para o problema em questão.

- Implementação de limitações das aplicações na decisão do cliente. Um exemplo de limitação seria o prazo que o cliente pretende resgatar o investimento. Outro exemplo seria se o investimento deve possuir as características de aplicação automática ou investimento automático. Tais limitações impediriam que fossem ofertados produtos sem essas características, o que muda os dados de treino da mesma.

Referências

[1] André Cardon e Daniel Nehme Müller. Introdução Às Redes Neurais Artificiais, *Universidade Federal do Rio Grande do Sul*, Porto Alegre, Rio Grande do Sul, Brasil. `http://www.inf.ufrgs.br/~danielnm/docs/intro_rna.pdf`, 1994. [Online; acessado 15-12-2016]. xi, 25, 27, 28, 29

[2] Z. Bodie, A. Kane, e A. J. Marcus. *Investments*, volume 8. The MacGraw-Hill Companies, New York, EUA, 2009. 4, 16

[3] H. M. Markowitz. Portifolio Selection, *Journal of Finance*, 7, USA. `http://cowles.yale.edu/sites/default/files/files/pub/mon/m16-all.pdf`, 1952. [Online; acessado 01-10-2016]. 5, 16, 17, 18

[4] CVM BRASIL. Instrução n.º 539, de 13 de novembro de 2013, Dispõe sobre o dever de verificação da adequação dos produtos, serviços e operações ao perfil do cliente. Diário Oficial [da] Republica Federativa do Brasil. Brasília, DF. `http://http://www.cvm.gov.br/legislacao/inst/inst539.html`. [Online; acessado 10-10-2016]. 7, 10, 11, 14

[5] CVM BRASIL. Instrução n.º 554, de 17 de dezembro de 2014, Inclui, revoga e altera dispositivos nas Instruções 155/91, 209/94, 278/98, 332/00, 356/01, 391/03, 399/03, 414/04, 429/06, 444/06, 461/07, 472/08, 476/09 e 539/13. Diário Oficial [da] Republica Federativa do Brasil. Brasília, DF. `http://http://www.cvm.gov.br/legislacao/inst/inst554.html`. [Online; acessado 10-10-2016]. 8

[6] CVM BRASIL. Instrução n.º 555, de 17 de dezembro de 2014, Dispõe sobre a constituição, a administração, o funcionamento e a divulgação das informações dos fundos de investimento. Revoga as Instruções 409/04; 411/04; 413/04; 522/12; 524/12; 536/13 e 549/14; os arts. 1º a 11 e 14 da Instrução 450/07; os arts. 1º a 3º da Instrução 456/07; os arts. 1º e 2º da Instrução 465/08 e os arts. 1º a 2º da Instrução 512/11. Diário Oficial [da] Republica Federativa do Brasil. Brasília, DF. `http://http://www.cvm.gov.br/legislacao/inst/inst555.html`. [Online; acessado 10-10-2016]. 9

[7] CVM BRASIL. Instrução n.º 564, de 11 de junho de 2015, Altera as Instruções 554 e 555, ambas de 17 de dezembro de 2014. Diário Oficial [da] Republica Federativa do Brasil. Brasília, DF. `http://http://www.cvm.gov.br/legislacao/inst/inst564.html`. [Online; acessado 10-10-2016]. 10

[8] CVM BRASIL. Instrução n.º 566, de 31 de julho de 2015, Dispõe sobre a oferta pública de distribuição de nota promissória. Altera o Anexo II da Instrução 400/03.

Revoga as instruções 134/90, 155/91, 422/05, 429/06 e os arts. 4º e 11 da instrução 554/14. Diário Oficial [da] Republica Federativa do Brasil. Brasília, DF. http:// http://www.cvm.gov.br/legislacao/inst/inst566.html. [Online; acessado 10-10-2016]. 10

[9] Pedro Gabriel Kenne Silva. O papel do controle interno na administração pública, *ConTexto UFRGS*, 2(1). http://www.ufrgs.br/necon/pciap.pdf, 2001. [Online; acessado 10-11-2016]. 11

[10] A. M. FLORENTINO. *Auditoria contábil*, volume 5. FGV, Rio de Janeiro, 1988. 11

[11] Roberto Ribeiro e Mari Rosa Machado. Análise do comportamento dos investidores no Multinve$t, *GEPROS: Gestão da Produção, Operações e Sistemas [1984-2430]*, 8(1). http://revista.feb.unesp.br/index.php/gepros/article/viewFile/993/482, 2013. [Online; acessado 10-11-2016]. 12

[12] Daniela Pasotto, Marcela Gonçalves Pazo, e Silvana Vicente Lobão. Uso de sistema especialista para decisão do perfil de um investidor, *Faculdade de Computação e Informática, Universidade Presbiteriana Mackenzie*. http://meusite.mackenzie.com.br/rogerio/tgi/2003Jess.PDF, 2003. [Online; acessado 10-11-2016]. 12

[13] Paulo Henrique Kaupa. Aplicação de Técnicas de Inteligência Artificial na Seleção de Ações de Investimento na Bolsa de Valores de São Paulo, *Universidade Nove de Julho*, São Paulo, São Paulo, Brasil. https://bibliotecatede.uninove.br/handle/tede/200, 2013. [Online; acessado 01-11-2016]. 16, 38

[14] Tiago Boechel. Algoritmo de Otimização: Uma abordagem híbrida utilizando o algoritmo das formigas e genético, *Universidade Federal de Santa Catarina*. https://repositorio.ufsc.br/bitstream/handle/123456789/85456/206968.pdf?sequence=1&isAllowed=y, 2003. [Online; acessado 10-11-2016]. 19, 20, 23

[15] Emile Aarts e Jan Korst. *Simulated annealing and boltzmann machines, a stochastic approach to combinatorial optimization and neural computing*, volume 1. Courier, Grã-Bretanha, 1989. 20

[16] Takashi Yoneyama e Cairo L. Júnior Nascimento. *Inteligência artificial em controle e automação*, volume 1. Edgard Blucher LTDA, São Paulo, 2000. 20

[17] Christos H. Papadimitriou. *Combinatorial optimization: algorithms and complexity*, volume 1. Prentice-Hall, Massachusets, 1982. 20, 23

[18] Stuart Russell e Peter Norvig. *Artificial Inteligence - A Modern Approach*, volume 3. Prentice Hall, New Jersey, 2010. 24, 30

[19] R. Eberhart e R. Dobbins. *Neural Networks PC Tools - A Practical Guide*, volume 1. Academic Press, San Diego, 1990. 25

[20] Michael J. A. Berry e Gordon Linoff. *Data Mining Techniques: for marketing, sales and customer support*, volume 1. Wiley Computer Publishing, USA, 1997. 32

[21] Ian H. Witten, Eibe Frank, e Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*, volume 3. Morgan Kaufmann, USA, 2011. 32

[22] Maria Madalena Dias e Roberto Carlos dos Santos Pacheco. Uma metodologia para o desenvolvimento de sistemas de descoberta de conhecimento descoberta de conhecimento, *Acta Scientiarum - Universidade Estadual de Maringá-Eduem*, 27, Maringá, Paraná, Brasil. `http://periodicos.uem.br/ojs/index.php/ActaSciTechnol/article/viewFile/1500/857`, 2005. [Online; acessado 10-12-2016]. 33

[23] Thomas H. Harrison. *Intranet Data Warehouse*, volume 1. Bekerley Brasil, São Paulo, 1998. 34

[24] Saulo Barbará Oliveira, Jorge de Abreu Soares, e Ana Lúcia Borba. *JCL e Utilitários do Sistema z/OS*, volume 1. Editora Ciência Moderna, Rio de Janeiro, 2011. 35

[25] Gary DeWard Brown. *Advanced ANSI COBOL with Structured Programming*, volume 1. John Wiley & Sons, Inc., USA, 1992. 36

[26] Nilo Ney Coutinho Menezes. *Introdução à Programação com Python. Ideal para quem nunca teve contato com programação*, volume 1. Novatec, São Paulo, 2010. 36

[27] K. Jarrod Millman e Michael Aivazis. Python for Scientists and Engineers. *Computing in Science & Engineering*, 13(undefined):9–12, 2011. 36

[28] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermüller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron C. Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Melanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian J. Goodfellow, Matthew Graham, Çaglar Gülçehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrançois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Joseph Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph P. Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, e Ying Zhang. Theano: A Python

framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688, 2016. 36

[29] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015. [Online; acessado 11-10-2016]. 37

[30] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007. 37

[31] Artur Filipe Ewald Wuerges e José Alonso Borba. Redes Neurais, Lógica Nebulosa e Algoritmos Genéticos: Aplicações e Possibilidades em Finanças e Contabilidade, *Revista de Gestão da Tecnologia e Sistemas de Informação*, 7(1). `http://www.scielo.br/pdf/jistm/v7n1/08.pdf`, 2010. [Online; acessado 14-12-2016]. 37

[32] Maria Terra Mello. Aplicação de redes neurais artificiais no processo de precificação de ações, Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação), *Universidade Federal de Pelotas*, Pelotas, Brasil. `http://www.ufpel.tche.br/prg/sisbi/bibct/acervo/info/2004/mono_marilia.pdf`, 2004. [Online; acessado 20-12-2016]. 37

[33] Carlos Henrique Dias. Um novo algoritmo genético para a otimização de carteiras de investimento com restrições de cardinalidade, Dissertação (Mestre em Matematica Aplicada), *Universidade Estadual de Campinas*, Campinas, Brasil. `http://www.bibliotecadigital.unicamp.br/document/?code=vtls000438801`, 2008. [Online; acessado 20-12-2016]. 37

[34] Luiz Paulo Rodrigues de Freitas Parreiras. MODELO GENÉTICO-NEURAL DE GESTÃO DE CARTEIRAS DE AÇÕES, Trabalho de Formatura (Engenheiro de Produção), *Escola Politécnica da Universidade de São Paulo*, São Paulo, Brazil. `http://pro.poli.usp.br/trabalho-de-formatura/modelo-genetico-neural-de-gestao-de-carteiras-de-acoes/`, 2003. [Online; acessado 20-12-2016]. 37

[35] Fernando Rafael Stahnke. USO DE DATA MINING NO MERCADO FINANCEIRO, Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação), *Centro Universitário FEEVALE*, Novo Hamburgo, Brasil. `http://tconline.feevale.br/tc/files/0001_1749.pdf`, 2008. [Online; acessado 10-12-2016]. 38

[36] Giuliano Padilha Lorenzoni. UMA INVESTIGAÇÃO ESTATÍSTICA SOBRE ANÁLISE TÉCNICA, Dissertação (Mestrado em Engenharia Elétrica), *Pontifícia Universidade Católica do Rio de Janeiro*, Rio de Janeiro, Brasil. `http://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=9192@1`, 2006. [Online; acessado 20-12-2016]. 38

[37] Marcelo Berenstein. Uso de Mineração de Dados na Bolsa de Valores, Monografia (Bacharel em Ciência da Computação), *Universidade do Vale do Itajaí*, Itajaí, Brasil. `http://siaibib01.univali.br/pdf/MarceloBerenstein.pdf`, 2010. [Online; acessado 20-12-2016]. 38

[38] Luiz Henrique Debei Herling, Marcus Vinícius Andrade Lima, Gilberto de Oliveira Moritz, e Pedro Henrique Marangoni. Testando a Hipótese de Eficiência de Mercado por Meio de Redes Neurais Artificias: um estudo de caso com as dez ações principais do IBOVESPA do primeiro quadrimestre de 2011, *Future Studies Research Journal: Trends and Strategies*, 4(2). `https://revistafuture.org/FSRJ/article/viewFile/103/182`, 2012. [Online; acessado 20-12-2016]. 38

[39] Roberto Pontes. *Inteligência Artificial Nos Investimentos*, volume 2. Clube de Autores, Rio de Janeiro, 2011. 38

[40] Ciniro A. L. Nametala, Alexandre Pimenta, Adriano César M. Pereira, e Eduardo G. Carrano. Uma estratégia automatizada de investimento por meio de redes neurais artificiais e preditores econométricos, *Simpósio Brasileiro de Sistemas de Informação*. `http://www.lbd.dcc.ufmg.br/colecoes/sbsi/2016/021.pdf`, 2016. [Online; acessado 26-02-2017]. 39

[41] Ilya Sutskever, James Martens, George Dahl, e Geoffrey Hinton. On the importance of initialization and momentum in deep learning, *Journal of Machine Learning Research*, 28, USA. `http://jmlr.org/proceedings/papers/v28/sutskever13.pdf`, 2016. [Online; acessado 10-12-2016]. 49

[42] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, e Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, *CoRR*, abs/1609.04836 . `http://arxiv.org/abs/1609.04836`, 2016. [Online; acessado 03-10-2016]. 50

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, e Rusian Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Learning Research*, 15, Espanha. `https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf`, 2014. [Online; acessado 28-11-2016]. 55