

TRABALHO DE GRADUAÇÃO

PREVISÃO DE VALORES DE AÇÕES
UTILIZANDO *DEEP LEARNING*

Pedro Henrique Hecksher Faber

Brasília, Julho de 2016



ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

**PREVISÃO DE VALORES DE AÇÕES
UTILIZANDO *DEEP LEARNING***

Pedro Henrique Hecksher Faber

*Relatório submetido como requisito parcial de obtenção
de grau de Engenheiro de Controle e Automação*

Banca Examinadora

Prof. Alexandre Romariz, ENE/UnB

Orientador

Prof. Alexandre Zaghetto, ENE/UnB

Examinador interno

Prof. Luís Figueredo, ENE/UnB

Examinador interno

Brasília, Julho de 2016

FICHA CATALOGRÁFICA

PEDRO HENRIQUE HECKSHER FABER

Pr visão d valor s d açõ s utilizando *deep-learning*,

[Distrito F d ral] 2016.

69p., 297 mm (FT/UnB, Eng nh iro, Control Automação, 2016

Trabalho d Graduação – Univ rsidad d Brasília.Facultad d T cnologia.

1. r d n ural convolucional

2.p rc ptron multicamada

3. m rcado financ iro

4.*deep learning*

I. M catrônica/FT/UnB

II. Pr visão d Valor s d Açõ s Utilizando *Deep Learning*

REFERÊNCIA BIBLIOGRÁFICA

FABER, PEDRO HENRIQUE HECKSHER, (2016). Pr visão d valor s d açõ s utilizando *deep-learning*. Trabalho d Graduação m Eng nharia d Control Automação, Publicação FT.TG-*n*º07, Facultad d T cnologia, Univ rsidad d Brasília, Brasília, DF, 69p.

CESSÃO DE DIREITOS

AUTOR: P dro H nriqu H cksh r Fab r

TÍTULO DO TRABALHO DE GRADUAÇÃO: Pr visão d valor s d açõ s utilizando *deep-learning*.

GRAU: Eng nh iro

ANO: 2016

É conc dida à Univ rsidad d Brasília p rmissão para r produzir cópias d st Trabalho d Graduação para mpr star ou v nd r tais cópias som nt para propósitos acadêmicos ci ntíficos. O autor r s rva outros dir itos d publicação n nhuma part d ss Trabalho d Graduação pod s r r produzida s m autorização por scrito do autor.

P dro H nriqu H cksh r Fab r

SHIS QL 16, Conj. 1 , nº 2, Lago Sul.

71640-215 Brasília – DF – Brasil.

Dedicatória

Eu dedico esse trabalho à você, minha irmã. Em breve será a sua vez de fazer essa jornada e traçar os seus próprios caminhos. Que durante todo esse trajeto você seja sempre a pessoa brilhante que é hoje e que suas escolhas levem à realização de todos os seus sonhos mas, principalmente, que você seja sempre feliz.

Pedro Henrique Hecksher Faber

Agradecimentos

Agradeço aos meus colegas de curso, que dividiram cada noite virada, cada trabalho extenuante, cada tensão antes de prova. Agradeço à todos os membros do LARA, que me acolheram no laboratório de braços abertos e me ajudaram intensamente durante essa reta final. Agradeço à todos os meus amigos, dentro e fora do curso. Eles me deram motivo para sorrir e continuar mesmo nas integrais mais complexas, nos espaços de estados mais difíceis e nas transformadas Z que menos faziam sentido. Um agradecimento especial aos meus amigos Eric, De Hong, Daniel, Teixeira, Jessé, Caio, Carioca e Cris, cuja amizade e companheirismo foi essencial durante toda essa jornada. Agradeço à todos os professores espetaculares que tive durante o curso. Devo fazer uma menção especial ao Prof. Romariz, oráculo da inteligência artificial, pela orientação e ajuda durante todo o projeto.

Agradeço ao meu pai e a minha mãe por todo apoio e educação e pela formação da pessoa que sou hoje mas, principalmente, por todo amor que sempre me deram.

Pedro Henrique Hecksher Faber

RESUMO

O mercado financeiro é considerado um dos mais complexos e voláteis sistemas criados pelo homem. Devido a grande quantidade de entradas suas complexas interações, é difícil para um humano analisar, prever, com acurácia, valores futuros das ações. Entretanto, é possível aplicar sistemas inteligentes modelos estatísticos para estimar esses valores futuros. Esse projeto utiliza duas diferentes redes neurais artificiais para fazer a previsão de valores futuros das ações PETR4. As ANNs implementadas são uma rede perceptron multicamada e uma rede neural convolucional profunda. A entrada de ambas as redes é um banco de dados de internet por tempo dos usuários da Google sobre transacionados e o histórico dos valores de mercado das ações listadas. As séries temporais usadas como entrada foram segmentadas com 94% sendo dados de treinamento e 6% sendo dados de validação. A rede neural convolucional profunda foi capaz de prever se as ações iriam subir ou descer 59.81% dos casos nos dados de validação. Comparando com a DCNN, o perceptron multicamada obteve um resultado melhor. Ele previu 68.22% dos dados de validação corretamente. Como esperado, as duas técnicas obtiveram resultados melhores que um classificador ingênuo, que teve uma taxa de acerto de apenas 50.72%.

Palavras Chave : rede neural convolucional, perceptron multicamada, mercado financeiro, *deep learning*

ABSTRACT

The financial market is considered one of the most complex and challenging man-made systems. Owing to its numerous input variables and complex interactions, it is difficult for man to analyze and predict, with accuracy, future stock values. It is, however, possible to apply intelligent systems and statistical models to estimate these future values. This project uses two different artificial neural networks for estimating the future value of PETR4 stocks. The artificial neural networks implemented were a multilayer perceptron and a deep convolutional neural network. The input for these systems is a database of internet over time of Google users on certain topics and the historical market values of the selected stock. The time series used as inputs were segmented with 94% as training data and 6% validation data. The convolutional neural network was able to predict whether the stock was going to increase or decrease in the next day for 59.81% of the validation data tested. Compared to the DCNN, the multilayer perceptron was able to achieve a better result, accurately predicting 68.22% of the validation datasets. As expected, both techniques obtained better results than a naïve classifier, which was able to predict only 50.72% of the future states.

Keywords: convolutional neural network, multilayer perceptron, stock market prediction, deep learning

SUMÁRIO

1	INTRODUÇÃO	1
1.1	CONTEXTUALIZAÇÃO	1
1.1.1	TIPOS DE ANÁLISE DE ATIVOS	2
1.1.2	TÉCNICAS DE APRENDIZADO DE MÁQUINA	4
1.1.3	TRABALHOS NA ÁREA	6
1.2	DEFINIÇÃO DO PROBLEMA	7
1.3	OBJETIVOS DO PROJETO.....	7
1.4	RESULTADOS OBTIDOS	8
2	FUNDAMENTOS	9
2.1	PERCEPTRON SIMPLES.....	9
2.1.1	ESTRUTURA DO PERCEPTRON	9
2.1.2	CONVERGÊNCIA DO PERCEPTRON.....	10
2.2	PERCEPTRON MULTICAMADA.....	11
2.2.1	ESTRUTURA DO PERCEPTRON MULTICAMADA.....	11
2.2.2	FUNCIONAMENTO DO PERCEPTRON MULTICAMADA	12
2.3	REDE NEURAL CONVOLUCIONAL.....	13
2.3.1	ESTRUTURA BÁSICA DAS REDES NEURAI CONVOLUCIONAIS.....	13
2.3.2	FUNCIONAMENTO DA REDE NEURAL CONVOLUCIONAL	14
2.4	USO DE ENTROPIA COMO FUNÇÃO DE ERRO	17
3	DESENVOLVIMENTO	19
3.1	AQUISIÇÃO DE DADOS	19
3.1.1	AQUISIÇÃO DE DADOS DO GOOGLE TRENDS	19
3.1.2	AQUISIÇÃO DE DADOS DA BOLSA DE VALORES	22
3.2	IMPLEMENTAÇÃO DAS REDES NEURAI	22
3.2.1	PERCEPTRON MULTICAMADA	23
3.2.2	REDE NEURAL CONVOLUCIONAL	26
4	RESULTADOS.....	30
4.1	PERCEPTRON MULTICAMADA.....	30
4.1.1	DISCUSSÃO SOBRE OS RESULTADOS DO PERCEPTRON MULTICAMADA	31
4.2	REDE NEURAL CONVOLUCIONAL.....	32

4.3	COMPARAÇÃO DOS RESULTADOS	33
5	CONCLUSÕES	36
	REFERÊNCIAS BIBLIOGRÁFICAS	37
	ANEXOS	39
I	TEOREMA DA CONVERGÊNCIA DO PERCEPTRON	40
II	BACK-PROPAGATION	43
III	CORRELAÇÃO CRUZADA ENTRE COTAÇÃO E POPULARIDADE NO GOOGLE TRENDS	47

LISTA DE FIGURAS

1.1	Gráfico da cotação da Petrobras nos últimos dois anos	2
1.2	Exemplo do gráfico <i>candlestick</i> ⁰	3
1.3	Exemplo de uma rede perceptron multicamada	5
1.4	Ganho médio acima do mercado por mês [1]	7
2.1	Perceptron simples	9
2.2	Exemplo de uma rede neural convolucional	14
3.1	Gráfico de popularidade do termo <i>deep learning</i> nos últimos anos	19
3.2	Gráfico exemplo de dados não ajustados	21
3.3	Gráfico exemplo de dados ajustados	21
3.4	Exemplo de cotação após normalização preenchimento com 0	23
3.5	Cotação da Petrobras com interpolação durante um período de 5 meses	25
3.6	Erro quadrático médio nos dados de treinamento, por época, para um η igual a 1.	26
3.7	Erro quadrático médio nos dados de treinamento, por época, para um η igual a 0.5. .	26
3.8	Disposição da CNN utilizada	27
4.1	Gráfico da queda do EQM para 80 épocas	31
4.2	Queda no EQM por época de treinamento da CNN	33
4.3	Queda no EQM por época de treinamento da CNN	33
4.4	Variação percentual do valor da ação PETR4 em relação ao dia anterior para todo o banco de dados	34
II.1	Gráfico que representa a propagação do sinal em um neurônio j	43
III.1	Correlação cruzada entre contatos "Ajuste Fiscal"	47
III.2	Correlação cruzada entre contatos "Corrupção"	48
III.3	Correlação cruzada entre contatos "Dívida"	48
III.4	Correlação cruzada entre contatos "Dilma"	49
III.5	Correlação cruzada entre contatos "Gasolina"	49
III.6	Correlação cruzada entre contatos "Imposto"	50
III.7	Correlação cruzada entre contatos "Índice Bovespa"	50
III.8	Correlação cruzada entre contatos "Lava Jato"	51
III.9	Correlação cruzada entre contatos "Mercado Financeiro"	51
III.10	Correlação cruzada entre contatos "Odebrecht"	52

III.11	Corr	lação	cruzada	ntr	contaçõ	s	"OGX"	52
III.12	Corr	lação	cruzada	ntr	contaçõ	s	"Ouro"	53
III.13	Corr	lação	cruzada	ntr	contaçõ	s	"PETR4"	53
III.14	Corr	lação	cruzada	ntr	contaçõ	s	"P trobras"	54
III.15	Corr	lação	cruzada	ntr	contaçõ	s	"Plataforma d P tról o"	54
III.16	Corr	lação	cruzada	ntr	contaçõ	s	"Pré Sal"	55
III.17	Corr	lação	cruzada	ntr	contaçõ	s	"PSDB"	55
III.18	Corr	lação	cruzada	ntr	contaçõ	s	"PT"	56
III.19	Corr	lação	cruzada	ntr	contaçõ	s	"S lic"	56
III.20	Corr	lação	cruzada	ntr	contaçõ	s	"T rouro Dir to"	57

LISTA DE SÍMBOLOS

Símbolos Latinos

e	Erro na saída
d	Valor desejado da saída
x	Entrada de uma rede neural artificial
w	Peso de uma conexão sináptica entre neurônios
s	Soma das entradas multiplicadas pelos pesos
y	Saída de um neurônio
l	Camada de entrada
L	Camada de saída
\mathbf{x}	Vetor ou matriz da entrada
\mathbf{w}	Vetor ou matriz dos pesos entre camadas
\mathcal{E}	Função de erro
\mathcal{S}	Entropia

Símbolos Gregos

η	Taxa de aprendizagem
σ	Função de ativação do neurônio
δ	Gradiente local
α	Constante de <i>momentum</i>

Siglas

ANN	Rede Neural Artificial - <i>Artificial Neural Network</i>
CNN	Rede Neural Convolucional - <i>Convolutional Neural Network</i>
EQM	Erro Quadrático Médio
DCNN	Rede Neural Convolucional Profunda - <i>Deep Convolutional Neural Network</i>
PETR4	Ação da Probabilidades com Prioridade na Distribuição de Dividendos

Capítulo 1

Introdução

Fazer previsões no mercado financeiro é um problema de grande complexidade. Existem diversas variáveis envolvidas na determinação dos valores das ações. Nesse projeto propõe-se desenvolver duas redes neurais aplicadas a esse problema. Elas devem ser capazes de detectar informações relevantes em um conjunto de dados de entrada e prever estados futuros do mercado.

1.1 Contextualização

Eugênio Fama, ganhador do Prêmio Nobel de Economia de 2013, publicou, em Maio de 1970, um artigo intitulado “Efficient Capital Markets: A Review of Theory and Empirical Work”. Esse trabalho propôs um conceito que passou a ser amplamente utilizado no mercado financeiro, que é a hipótese do mercado eficiente (EMH) [2][3]. A EMH diz que o preço dos ativos reflete em toda a informação disponível, isso implica em um agente não conseguir obter, consistentemente, retornos acima da média do mercado devido ao fato do preço dos ativos reagir às notícias.

Como nem toda informação é disponível para todos, ineficiências no mercado acabam ocorrendo. Métodos computacionais podem correlacionar dados extraídas das informações não triviais, que poderiam passar despercebidas por um analista técnico. Obter essas informações implica em uma vantagem competitiva para quem as possui pois, não estando disponível para todos o mercado, não haveria uma reação eficiente, acarretando, para o investidor detentor das mesmas, um ganho médio acima do zero.

Quando uma notícia significativa se torna pública, ela implica numa mudança de comportamento dos investidores. Essa mudança de comportamento é um fator humano que, por sua natureza, é caótico. Consistentemente, os valores das ações variam dependendo das especulações geradas em resposta à notícia. Na Figura 1.1, por exemplo, vê-se uma queda na cotação PETR3, da Petrobras, após outubro de 2014, que corresponde ao escândalo do Petrobrás e aos resultados da eleição. O mercado, portanto, tem variações caóticas quase imprevisíveis. Análises convencionais tentam aprovar ineficiências no mercado que independentemente se alinham com as

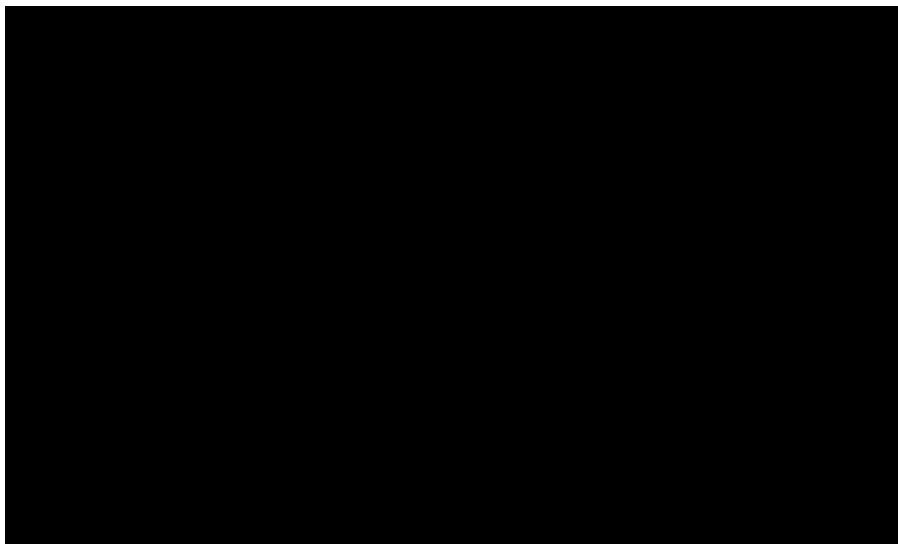


Figura 1.1: Gráfico da cotação da Petrobras nos últimos dois anos

speculações geradas por notícias.

Nos mercados atuais, técnicas convencionais, como análise de gráficos *candlesticks*, têm perdido espaço para técnicas mais inovadoras. Essa mudança tem acontecido devido ao aumento dos investidores, que conseguiram, aumentando a velocidade de resposta do mercado a novas informações, tornam mais difícil encontrar informações não vistas por outros investidores.

Quando há uma nova informação disponível no mercado, a procura por temas correlacionados feita em sites de pesquisa como o Google aumenta, assim como o número de publicações sobre o assunto em redes sociais como Twitter e Facebook. A variação na popularidade do tema comum antes de a resposta do mercado. Essa correlação entre procura e resposta é sutil e pode não ser simplesmente observada.

1.1.1 Tipos de Análise de Ativos

Nessa seção, as principais técnicas de análise de ativos serão abordadas.

1.1.1.1 Análise Fundamentalista

O primeiro método é a análise fundamentalista. Ele consiste em estudar a situação financeira da empresa, como balanço patrimonial, demonstração do resultado do exercício, fluxo de caixa, etc. Esses dados, apresentados em relatórios, são chamados de demonstrações financeiras.

Analistas fundamentalistas utilizam as demonstrações financeiras para estimar um valor de mercado para as empresas. Sabendo-se esse valor estimado, é possível estimar um valor para a ação da empresa em especulação. Caso esse valor esteja mais baixo que o valor de mercado das ações, é um indicativo de que está num bom momento de vender, caso esteja acima do valor de mercado, significa que está num bom momento de comprar. Entretanto, esse não é o único



Figura 1.2: Ex mplo d gráfico *candlestick*⁰

critério para a análise fundamentalista. Também se leva em conta a perspectiva de crescimento do mercado na área do ativo que está sendo comprado, no histórico da empresa, na sua credibilidade. A análise é feita a partir da empresa mesmo, normalmente para um investimento de médio a longo prazo.

A análise fundamentalista pode ser feita também a partir de commodities ou de moedas, baseando-se no mercado daquele produto ou da economia do próprio país.

1.1.1.2 Análise Técnica

A análise técnica pura segue a hipótese de que toda, ou quase toda a informação necessária para saber valores futuros de ativos está contida no próprio mercado. Analistas técnicos focam em determinar o valor futuro de ações baseando-se em uma análise numérica das tendências do mercado. Modelos matemáticos como média móvel, média móvel exponencial, bandas de Bollinger, análise de Fourier são utilizados nesse tipo de análise.

Essa metodologia de especulação surgiu no século XVII com alguns conceitos primitivos algumas máximas a serem seguidas [1]. No século seguinte, analistas japoneses desenvolveram o gráfico *candlestick* que contém as informações de abertura, fechamento (ou quotação). Esse gráfico contém informações de abertura, fechamento, máximo e mínimo em cada *candle*. Normalmente utiliza-se esse tipo de gráfico como base para outras análises técnicas. Na Figura 1.2, pode-se ver um gráfico *candlestick*, com duas médias móveis exponenciais plotadas com períodos diferentes.

1.1.1.3 Análise Computacional

O tradicional método frequentemente utilizado originou-se da análise técnica, que é a utilização de métodos computacionais para auxiliar ou conduzir a análise. Atualmente, com a disseminação da computação, é incomum ver analistas técnicos trabalhar mesmo com o auxílio de softwares e algoritmos computacionais.

Dados cálculos de médias móveis simples até aplicações de filtros de Kalman *machine lear-*

⁰Gráfico extraído do site www.bussoladoinvestidor.com.br no dia 21/06/16

ning, a análise técnica foi revolucionada pelo advento da computação. *Algotraders* (analistas que desenvolvem algoritmos para comprar e vender ativos automaticamente) utilizam essas diversas técnicas para tentar ter alguma vantagem no mercado.

1.1.2 Técnicas de Aprendizado de Máquina

Aprendizado de máquina é uma área que trabalha com otimizações matemáticas com o objetivo de reconhecer padrões. Em termos gerais, técnicas de aprendizado de máquina tentam modelar um sistema ou prever resultados futuros com informações de saídas ou estados passados.

Existem diversos tipos de aprendizado de máquina, desenvolvidos para diferentes objetivos. Algumas podem surgir através de inspirações na natureza no funcionamento do cérebro humano, como algoritmos genéticos e redes neurais artificiais, outras vêm da generalização de modelos estatísticos, como as cadeias de Markov. Nessa seção será abordada, de forma sucinta, algumas técnicas.

1.1.2.1 Redes Neurais Artificiais(ANN)

As redes neurais artificiais surgiram através da observação do sistema nervoso humano como resolver problemas. Em redes artificiais, os “neurônios” são unidades de processamento com entradas, saídas e interconectando, dessa forma mimetizando sistemas nervosos.

Redes neurais artificiais possuem os seguintes pré-requisitos [4]:

- Unidades de processamento, chamadas de neurônios.
- Interconexões com diferentes pesos entre as unidades de processamento. Essas conexões definem como a ativação na saída de um neurônio leva à entrada de um próximo.
- Regras de ativação que agem nos sinais de entrada para produzir uma saída.
- Padrões de aprendizado que ajustam os pesos das entradas e saídas do sistema.

Os neurônios podem ser organizados de diversas formas mas, geralmente, são organizados em camadas. Normalmente existe uma camada de entrada, na qual os dados entram na rede, existe uma ou mais camadas ocultas, na qual os dados são processados, uma camada de saída, onde os dados são manipulados para ficar no seu formato de saída. A Figura 1.1.2.1 mostra uma representação de três tipos de camadas. Os neurônios h_1 a h_{k_1} representam a primeira camada oculta, g_1 a g_{k_2} representam a segunda camada oculta, f_1 a f_2 representam a camada de saída. k_1 e k_2 são os números de neurônios da primeira e segunda camada escondida, respectivamente.

Como dito anteriormente, existem múltiplos tipos de ANNs, para diferentes propósitos. Redes *feed-forward* são as mais comuns, especialmente os perceptrons e perceptrons multicamadas. Elas foram as primeiras a serem desenvolvidas e são de fácil implementação. Nesse tipo de rede o

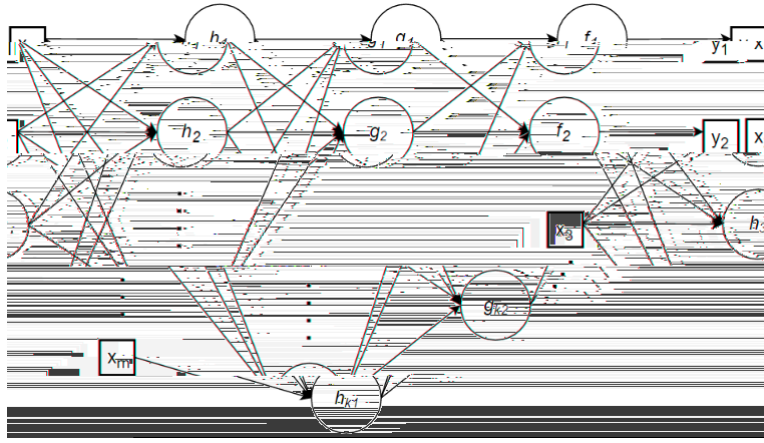


Figura 1.3: Ex mplo d uma r d p rc ptron multicamada

movimento dos dados é unidirecional. Ele é recebido na entrada, passa pelas camadas escondidas e sai pela saída. Não existe um *feedback* interno na rede durante a etapa.

O processo de aprendizado de todas as ANNs pode ser com dados durante a execução, de forma *online*, ou com dados capturados anteriormente. A técnica mais comum de treinamento é chamada de *back-propagation*. Durante esse tipo de treinamento, para cada dado de entrada o computador calcula o erro entre a saída e o valor esperado real. Após o cálculo do erro, o sistema propaga esse erro para as camadas anteriores, ajustando os pesos entre as camadas. Seguindo esse processo, novos parâmetros são definidos para as entradas dos neurônios seguindo uma taxa de aprendizado arbitrária, que pode ser tanto estática, quanto dinâmica, de forma a aproximar a saída do valor real esperado[2].

Repetindo esses passos para diferentes entradas no sistema, os pesos se ajustam de forma a tentar encontrar o menor erro de saída. Para uma base de dados extensa o erro tende a aproximar de um valor mínimo, local ou absoluto, para aquele banco de dados usado no treinamento. Caso o erro não chegue a um valor mínimo, treinamos a rede múltiplas vezes em repetições chamadas de épocas.

1.1.2.2 Deep Learning

Deep learning é uma área da ciência da computação que utiliza algoritmos de aprendizado de máquina para modelar sistemas que requerem um alto nível de abstração. Utilizando múltiplas camadas de processamento, normalmente se apresenta em forma de estruturas mais complexas que as *perceptron* multicamada. Algoritmos de aprendizado profundo têm sido usados em diversos campos e têm produzido alguns dos melhores resultados dentre as possíveis soluções com *machine learning*[5].

Redes de aprendizado profundo, assim como redes neurais artificiais comuns podem ser utilizadas para as seguintes três classes de problema:

- Treinamento supervisionado: Neste tipo de aprendizado existe um conjunto de dados de

tr inam nto qu contém as saídas sp radas para div rsas possív is ntradas, assim a r d varia s us parâmetros a fim d ncontrar uma estrutura qu ncontr as saídas mais próximas dos valor s sp rados.

- Tr inam nto não sup rvisionado: N st tipo d tr inam nto não xist (ou não é usado) um valor d saída d s jado.
- Sist ma híbrido: Possui part do sist ma s ndo sup rvisionado part não, normalm nt são mais compl xos d s r m impl m ntados.

D ntr as técnicas mais impl m ntadas d *deep learning*, val m ncionar uma: *convolutional neural networks*. Ess método t v um grand aum nto m sua popularidad , m sp cial para proc s-sam nto d imag ns, d vido à qualidad dos r sultados qu têm sido publicados com ssa técnica. Para d t cção d caract r s scritos a mão l itura d t xto, por x mplo, ssa é a t cnologia *state-of-the-art* [6].

1.1.3 Trabalhos na Área

Durant a última década, vários p squisador s instituição s t ntaram pr v r valor s d ativos no m rcado utilizando r d s n urais artificiais.

Uma r l vant p squisa foi conduzida na Nanyang T chnological Univ rsity [7]. A maioria dos studos na ár a d pr visão d valor s d ação s, t ntam pr v r os valor s um ciclo a fr nt , contudo, ssa p squisa t ntou aplicar um algoritmo qu fiz ss uma pr visão *multi-step* m qu s t ntava pr v r a situação do m rcado vários stados a fr nt . D vido à compl xidad do probl ma, os r sultados foram fracos para pr visão s *multi-step*. Obtiv ram, ntr tanto, r sultados qu os p squisador s consid raram satisfatórios para pr dição s im diatas, com uma taxa d ac rto d 64.7% [7].

Outro r sultado int r ssant foi obtido por um grupo d student s d doutorado d Harvard. El s inv stiam no índic s manal S&P 500, s guindo as s guint s r gras: S hou v r um aum nto pr visto, l s inv stiriam todo o dinh iro m S&P 500; s hou v ss um d crécimo pr visto, l s r mov riam todo o dinh iro inv stido salvá-lo para mais tard .

O dif r ncial d ss proj to foi o uso d *big-data* para id ntificar os fator s com maior significância estatística no m rcado. O grupo r uniu mais d 7.000 conjuntos d dados d assuntos r lacionados a m rcado financ iro D pois d usar um classificador Bay siano a fim d liminar os dados statisticam nt não significativos, o núm ro d conjuntos d dados foi r duzido para 20. Aplicando st s 20 conjuntos como ntradas m um p rc ptron multicamada, com um tr inam nto m *back-propagation* l s cons guiram uma pr cisão g ral d 63,64%. Com uma pr cisão para aum ntos d 79,37%, para qu das d 36,11%. O tr inam nto incluiu dados d 1999-2010, os utilizou os dados d 2011 2012 para t st . O mod lo obt v , m média, um lucro 2.39% acima do cr scim nto do m rcado m nsalm nt [1].

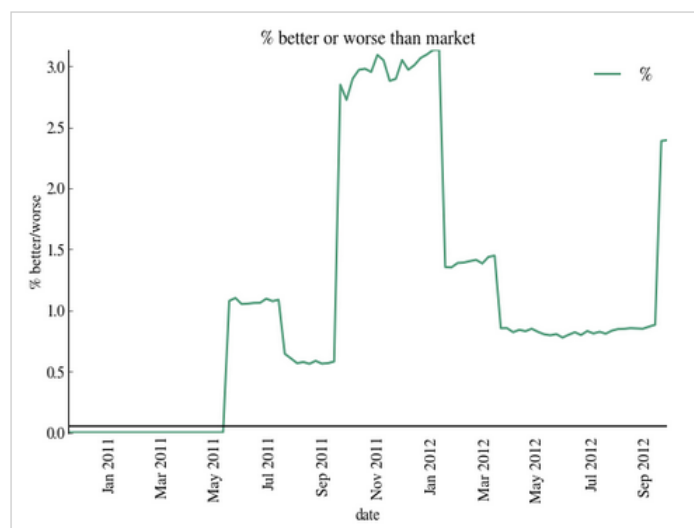


Figura 1.4: Ganho médio acima do mercado por mês [1]

Existiram, além dos mencionados, diversas outras pesquisas com bons resultados em prever valores no mercado financeiro [8, 9, 10, 11].

1.2 Definição do problema

O mercado financeiro é um sistema não linear de grande complexidade. Mesmo as modelagens mais robustas têm dificuldade de acertar previsões. Essa complexidade é intrínseca ao sistema, que depende de inúmeras variáveis. Para fazer previsões acertar constantemente, um investidor teria que ter todas as informações envolvidas no mercado: todos os relatórios financeiros de cada empresa, mesmo os dados não publicados; quanto cada outro investidor vai investir em qual ativo; os dados macroeconômicos de cada país, incluindo os não divulgados ao público; além de diversos outros fatores como, fenômenos naturais e interações humanas. Como é, em termos práticos, impossível ter acesso a todas essas informações, a modelagem com o objetivo de fazer previsões obter lucro, fica restrita às informações acessíveis.

1.3 Objetivos do projeto

Este trabalho tem como objetivo desenvolver um sistema inteligente que seja capaz de prever se o preço de um ativo vai subir ou descer no dia seguinte. A rede neural artificial desenvolvida, sistematicamente, obter um lucro médio acima do mercado. O ativo selecionado para este trabalho foi a ação PETR4, da Petrobras. O sistema proposto desenvolver como entradas dados de *big data*, disponibilizados pelo Google Trends: dados de mercado, desenvolvida capaz de utilizar esses dados para correlacionar popularidade, na plataforma de pesquisa Google, de temas pré-selecionados arbitrariamente com as ações da Petrobras.

1.4 Resultados obtidos

A taxa de acertos obtida para a rede neural convolucional foi de 59.81% para os dados de validação 64.60% para os dados de treinamento. Para a rede perceptron multicamada desenvolvida a taxa de acerto foi de 68.22% para os dados de validação 75.50% para os dados de treinamento. As duas redes obtiveram resultados significativamente superiores a um classificador ingênuo, que teve, para os dados de validação, uma taxa de acerto de 50.72%.

Executando simulações de treinamento para o perceptron multicamada, em um período de 107 dias o rendimento médio é de 21.44%. A ação selecionada obtém um crescimento de apenas 1.85% para o mesmo período.

Capítulo 2

Fundamentos

"We've arranged a global civilization in which most crucial elements profoundly depend on science and technology." - Carl Sagan

Este capítulo contém as fundamentações teóricas necessárias para a compreensão do desenvolvimento do projeto.

2.1 Perceptron Simples

2.1.1 Estrutura do perceptron

O perceptron é o modelo mais básico de uma rede neural. A sua estrutura básica pode ser vista na Figura 2.1.

Sua funcionalidade consiste em receber entradas, x_i , multiplicá-las pelos seus respectivos pesos, ω_i , somar todos os resultados da multiplicação. Além das entradas do sistema, também soma-se um bias, que tem seu valor determinado como 1 também é multiplicado por um peso específico, b , segundo a Equação 2.1.

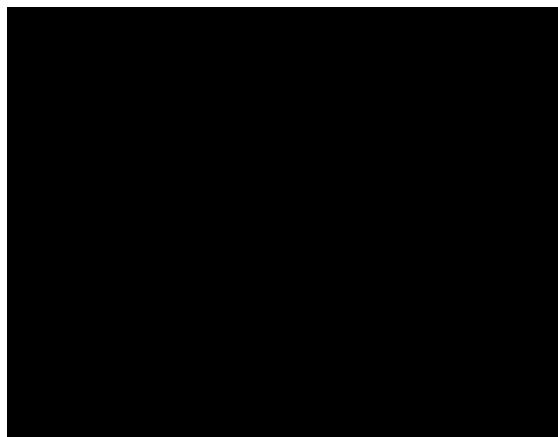


Figura 2.1: Perceptron simples

$$s = \sum_{i=1}^m \omega_i x_i + b \quad (2.1)$$

A soma das entradas ponderadas pelos pesos passa por uma função de ativação, σ , resultando na sua saída, y . As funções de ativação utilizadas neste trabalho são as funções logística tangente hiperbólica, dadas pelas Equações (2.2)–(2.3), respectivamente.

$$y = \sigma(s) = \frac{1}{1 + e^{-s}} \quad (2.2)$$

$$\sigma(s) = \tanh(s) \quad (2.3)$$

2.1.2 Convergência do perceptron

Suponha-se que, em um espaço \mathbb{R}^3 existam duas classes lineares separáveis, como uma maçã e uma tigela, dispostas lado a lado sobre uma mesa. Neste cenário é fácil delimitar um plano que separe os dois objetos. Agora, assumindo as mesmas classes, ou objetos, só que com a maçã dentro da tigela, não é mais possível separá-las com um plano. É necessário uma superfície mais complexa. Este exercício mental pode ser generalizado para qualquer espaço \mathbb{R}^k , sendo $k \in \mathbb{N}^*$. Para esse espaço, podem existir duas classes, \mathcal{C}_1 – \mathcal{C}_2 , que entre si são lineares separáveis, possuindo um hiperplano que as separe, mas ao aproximá-las no espaço, elas perdem essa propriedade.

Um perceptron não é capaz de separar duas classes \mathcal{C}_1 – \mathcal{C}_2 caso elas não sejam lineares separáveis. Portanto, para um perceptron simples, será assumido que \mathcal{C}_1 – \mathcal{C}_2 contém pelo menos um hiperplano que as separem em \mathbb{R}^k .

Sejam os dados de entrada da rede representados através de um vetor

$$\mathbf{x}(n) = \begin{bmatrix} 1, & x_1(n), & x_2(n), & x_3(n), & \dots, & x_m(n) \end{bmatrix}^T \quad (2.4)$$

, de forma correspondente, os pesos

$$\mathbf{w}(n) = \begin{bmatrix} b(n), & w_1(n), & w_2(n), & w_3(n), & \dots, & w_m(n) \end{bmatrix}^T \quad (2.5)$$

Utilizando as Equações (2.4)–(2.5) podemos representar a Equação (2.1) da seguinte forma

$$s = \mathbf{w}^T(n)\mathbf{x}(n) \quad (2.6)$$

O conjunto de dados que serão usados como entrada no perceptron são: \mathcal{H}_1 o subespaço que contém todos os vetores de treinamento $\mathbf{x}_1(1), \mathbf{x}_1(2), \dots$, pertencentes à classe \mathcal{C}_1 – \mathcal{H}_2 o subespaço que contém todos os vetores $\mathbf{x}_2(1), \mathbf{x}_2(2), \dots$, pertencentes à classe \mathcal{C}_2 . $\mathcal{H}_1 \cup \mathcal{H}_2$ representa todo o espaço de entradas \mathcal{H} .

Para o espaço de entradas \mathcal{H} o treinamento envolve ajustar os pesos \mathbf{w} para que seja possível afirmar que

- $\mathbf{w}^T \mathbf{x} > 0$ para cada vetor de entrada \mathbf{x} pertencente à classe \mathcal{C}_1
- $\mathbf{w}^T \mathbf{x} \leq 0$ para cada vetor de entrada \mathbf{x} pertencente à classe \mathcal{C}_2

A prova do teorema que demonstra que o perceptron converge (veja o Apêndice I), demonstra que é possível encontrar uma solução \mathbf{w}_0 que satisfaz completamente as classes \mathcal{C}_1 e \mathcal{C}_2 .

Para demonstração no Apêndice I, o algoritmo de adaptação dos pesos para o perceptron simples é:

1. Se o membro n do conjunto de treinamento $\mathbf{x}(n)$ foi corretamente classificado pelo vetor de pesos $\mathbf{w}(n)$, nenhuma correção é feita:

$$\mathbf{w}(n+1) = \mathbf{w}(n) \quad (2.7)$$

2. Caso contrário, os pesos $\mathbf{w}(n)$ são ajustados de acordo com a seguinte regra

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)x(n) \quad \text{se } \mathbf{w}^T \mathbf{x} > 0 \quad \mathbf{x} \text{ pertencente à classe } \mathcal{C}_2 \quad (2.8)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)x(n) \quad \text{se } \mathbf{w}^T \mathbf{x} \leq 0 \quad \mathbf{x} \text{ pertencente à classe } \mathcal{C}_1 \quad (2.9)$$

Em que η é o fator de aprendizagem está definido entre 0 e 1. O valor de η para qualquer resultado não é importante pois, simplesmente alterando o número de iterações necessárias para separar as classes, não altera a separabilidade delas mesmas.

2.2 Perceptron Multicamada

Retornando à analogia da maçã da tigela, para um espaço \mathbb{R}^3 , foi demonstrado que o perceptron simples consegue separar os dois objetos, desde que eles sejam linearmente separáveis. O perceptron simples consegue limitar um plano no qual a maçã está de um lado e a tigela está do outro. Em um cenário em que a maçã e a tigela estão em uma posição que nenhum plano consiga separar os dois, como quando a maçã está dentro da tigela, o perceptron falha em separar os dois. O perceptron multicamada, neste tanto, é uma modificação do perceptron simples que consegue fazer a separação de duas classes, ou objetos, que não são linearmente separáveis. Quando a maçã está dentro da tigela, por exemplo. Novamente essa analogia pode ser generalizada para um espaço \mathbb{R}^k , sendo $k \in \mathbb{N}^*$. [12]

2.2.1 Estrutura do Perceptron Multicamada

O perceptron multicamada é uma estrutura ordenada, visto que é formada por camadas de perceptrons simples. A Figura 1.3 contém um exemplo de uma rede desse tipo, cada neurônio contido na Figura 1.3 é representado pela Figura 2.1.

O perceptron multicamada é organizado em três tipos de camada diferentes:

1. CAMADA DE ENTRADA: que é composta pelos dados de entrada de cada iteração.
2. CAMADA ESCONDIDA: que são camadas cujas saídas são entradas de próximas camadas.
3. CAMADA DE SAÍDA: em que sua saída é a saída do sistema em si.

O perceptron multicamada necessariamente contém pelo menos uma camada de entrada, uma camada de saída. Podendo haver qualquer número de camadas escondidas.

Os perceptrons que não estão na saída do sistema, ou seja, neurônios de camadas escondidas, têm a saída de suas funções de ativação como entrada de perceptrons da próxima camada. Essa estrutura, embora é fácil compreender, contém uma dificuldade intrínseca: não existe uma forma direta, direta também, o erro de neurônios das camadas escondidas. Cada neurônio, é portanto projetado para executar duas tarefas:

1. Calcular o sinal de saída, que é expressado através de uma função não linear em relação ao sinal de entrada $\mathbf{x}(n)$ aos pesos $\mathbf{w}(n)$, também chamados de pesos sinápticos.
2. Computar uma estimativa do vetor gradiente do erro em relação aos pesos da entrada do neurônio, que é necessário para a propagação do erro para trás.

2.2.2 Funcionamento do Perceptron Multicamada

Com essa estrutura definida, o funcionamento da rede, para uma iteração, ocorre da seguinte forma:

1. INICIALIZAÇÃO: É o processo no qual se definem os parâmetros iniciais da rede.
2. APRESENTAÇÃO DOS DADOS DE TREINAMENTO: A primeira camada da rede é alimentada com os sinais de entrada.
3. FEED-FORWARD: É processada a propagação dos sinais para frente, conhecida como *feed-forward*. Ela funciona assim como o perceptron simples, para o neurônio j na camada l a soma das entradas é

$$s_j^{(l)}(n) = \sum_i w_{ji}^{(l)}(n) y_i^{(l-1)} \quad (2.10)$$

onde $y_i^{(l-1)}$ é a saída da camada anterior. Aplicando uma função de ativação σ a saída do neurônio fica

$$y_j^{(l)} = \sigma_j(s_j^{(l)}(n)) \quad (2.11)$$

se o neurônio for da primeira camada escondida

$$y_j^{(1)} = \sigma_j\left(\sum_i w_{ji}^{(1)}(n) \mathbf{x}_i(n)\right) \quad (2.12)$$

Se o neurônio j pertence à camada de saída (i.e., $l = L$ — L é a profundidade da rede),

$$y_j^{(L)} = o_j \quad (2.13)$$

o sinal de erro computado na última camada é

$$e_j(n) = d_j(n) - o_j(n) \quad (2.14)$$

no qual $d_j(n)$ é o valor desejado para a saída do neurônio j na última camada.

4. BACK-PROPAGATION: É a propagação para trás do erro pelas camadas, computada os gradientes locais $\delta_j^{(l)}$

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)} \sigma'_j(s_j^{(L)}(n)) & \text{para um neurônio } j \text{ na camada de saída } L \\ \sigma'_j(s_j^{(L)}(n)) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) & \text{para um neurônio } j \text{ em uma camada escondida } l. \end{cases} \quad (2.15)$$

Na qual σ' denota a derivada da função de ativação. Após o cálculo do erro, os pesos são ajustados de acordo com a seguinte atualização da Regra de Delta

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha [\Delta w_{ji}^{(l)}(n-1)] + \eta \delta_j^{(l)}(n) y_i^{(l-1)}(n) \quad (2.16)$$

onde η é a taxa de aprendizagem — α é a constante de *momentum*.

Os passos 3–4 são repetidos para novas épocas até que critérios predefinidos de parada sejam cumpridos.

A demonstração do algoritmo de *back-propagation* está no Apêndice II.

2.3 Rede Neural Convolucional

Mais uma vez voltando à maçã da tigela, a rede neural convolucional (CNN) faz a separação de dois objetos de uma forma diferente das redes de perceptrons simples multicamadas. Esse tipo de rede é capaz, assim como o perceptron multicamada de classificar objetos não lineares, mas ao invés de limitar uma superfície — que serve para os dois objetos, a CNN utiliza as *features* das entradas para identificar a qual classe pertencem. No caso da maçã da tigela, por exemplo, a rede aprenderia a identificar propriedades que as distanciam uma da outra, como cor, formato, posição no espaço, etc.

2.3.1 Estrutura Básica das Redes Neurais Convolucionais

As redes neurais convolucionais são divididas em camadas. Os dados que entram na rede passam por um processo de *feed-forward* — o erro é propagado para trás por *back-propagation*, assim como o perceptron multicamada. A estrutura das camadas — a forma como o *feed-forward* e o *back-propagation* acontecem, no entanto, é fundamentalmente diferente.

A estrutura básica da rede pode ser vista na imagem 2.2.



Figura 2.2: Ex mplo d uma r d n ural convolucional

Ess tipo d r d é dividido ntr quatro tipos d camada:

1. CAMADA DE ENTRADA: Assim como outras r d s, ssa é uma camada qu contém os dados d ntrada.
2. CAMADA DE CONVOLUÇÃO: A camada d convolução é composta por um núm ro arbitrário d grad s r tangular s d n urônios, chamadas d *feature maps*. A ntrada d cada n urônio d uma grad f_l sp cífica r c b os dados d uma s ção da malha ant rior. Os p sos das ntradas d cada n urônio são compartilhados ntr todos os n urônios d uma m sma grad . Ess s p sos só são dif r nt s ntr os *feature maps*, cada um possuindo sua própria matriz d p sos. A camada d convolução t m ss nom porqu , como cada n urônio d d stino é uma composição d uma malha d n urônios ant rior, la é uma convolução spacial da sua ntrada os p sos são o filtro da convolução.
3. CAMADA DE POOLING: A camada d *pooling* é mais simpl s qu a camada d convolução. N ssa camada, s çõ s r tangular s da imag m são subamostradas d forma qu saia ap nas uma única saída d ss proc sso. Essa subamostra comum nt é f ita a partir da s l ção do valor máximo da s ção.
4. CAMADA COMPLETAMENTE CONECTADA: Essa camada s apr s nta no final da r d , normalm nt após div rsas camadas int rcaladas d convolução *pooling*. Essa camada é similar ao p rc ptron multicamada: todos as saídas d cada n urônio da camada ant rior são con ctados a p rc ptrons d saída.

2.3.2 Funcionamento da Rede Neural Convolucional

As r d s n urais convolucionais normalm nt são aplicadas para o proc ssam nto d imag ns, como d t cção d obj to r conh cim nto d padrão s. Como a ntrada é, cons qu nt m nt ,

sempre uma matriz, os filtros de convolução as regiões de *pooling* normalmente são submatrizes quadradas, representados por tamanhos $m \times m$ e $k \times k$, respectivamente. Como no projeto foram utilizadas múltiplas séries temporais, as demonstrações a seguir são feitas para tamanhos genéricos $m_1 \times m_2$ e $k_1 \times k_2$.

2.3.2.1 Propagação dos dados na camada convolucional

Suponha-se que exista uma camada de neurônios l , de tamanho $N_1 \times N_2 \forall (N_1, N_2) \in \mathbb{N}$, seguida de uma camada convolucional. Aplicando-se um filtro w de tamanho $m_1 \times m_2$ a camada convolucional vai ter um tamanho de $(N_1 - m_1 + 1) \times (N_2 - m_2 + 1)$. A soma das entradas de cada neurônio de l será, então

$$x_{ij}^l = \sum_{a=0}^{m_1-1} \sum_{b=0}^{m_2-1} w_{ab} y_{(i+a)(j+b)}^{l-1} \quad (2.17)$$

Com essa soma, pode-se calcular a função de ativação do neurônio

$$y_{ij}^l = \sigma(x_{ij}^l) \quad (2.18)$$

2.3.2.2 Propagação dos dados na camada de *pooling*

A camada de *pooling* é, em geral, simples. Existem mais de um tipo de camada de *pooling*, contudo, nessa fundamentação, só será abordado o *max-pooling*.

Para aplicar o *max-pooling*, basta selecionar a matriz ou vetor de entrada menor de tamanho $k_1 \times k_2$. Para cada uma dessas submatrizes de entrada, será selecionado o neurônio de maior valor para passar o valor para a próxima camada. Assim, um neurônio da saída influencia a próxima camada a cada iteração. Com um *pooling* para cada saída $k_1 \times k_2$, a matriz é reduzida a um tamanho $\frac{N_1}{k_1} \times \frac{N_2}{k_2}$.

2.3.2.3 Propagação dos dados na camada completamente conectada

Essa camada, presente no final da CNN, é como um perceptron multicamada normal. Todos os dados da última camada, convolucional ou de *pooling*, são conectados à perceptrons que computam a saída do sistema.

2.3.2.4 *Backpropagation* na camada convolucional

Pelo que foi visto na fundamentação das outras ANNs neste trabalho, para o ajuste dos pesos das conexões de cada camada, é necessário calcular o gradiente de cada peso.

Assumindo uma função de erro \mathcal{E} para uma camada l de convolução, o cálculo do erro da camada $l - 1$ será baseado na derivada parcial do custo \mathcal{E} em relação a cada saída de cada

o urônio, $\partial \mathcal{E} / \partial y_{ij}^l$. Primeiro, calcula-se o componente do gradiente de cada peso aplicando a regra da cadeia em cima da soma da entrada de todos os neurônios em que os pesos ocorrem. É necessário somar todos os casos x_{ij}^l em que w_{ab} ocorre, já que há um compartilhamento do peso.

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial w_{ab}} &= \sum_{i=0}^{N_1-m_1} \sum_{j=0}^{N_2-m_2} \frac{\partial \mathcal{E}}{\partial x_{ij}^l} \frac{\partial x_{ij}^l}{\partial w_{ab}} \\ &= \sum_{i=0}^{N_1-m_1} \sum_{j=0}^{N_2-m_2} \frac{\partial \mathcal{E}}{\partial x_{ij}^l} y_{(i+a)(j+b)}^{l-1} \end{aligned} \quad (2.19)$$

A substituição

$$\frac{\partial x_{ij}^l}{\partial w_{ab}} = y_{(i+a)(j+b)}^{l-1} \quad (2.20)$$

pod-se refazer tirando a derivada parcial da Equação 2.17 em relação a w_{ab} . Expandindo a expressão $\partial \mathcal{E} / \partial x_{ij}^l$ pela regra da cadeia,

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial x_{ij}^l} &= \frac{\partial \mathcal{E}}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \\ &= \frac{\partial \mathcal{E}}{\partial y_{ij}^l} \frac{\partial}{\partial x_{ij}^l} (\sigma(x_{ij}^l)) \\ &= \frac{\partial \mathcal{E}}{\partial y_{ij}^l} \sigma'(x_{ij}^l) \end{aligned} \quad (2.21)$$

Como já se sabe o valor de $\partial \mathcal{E} / \partial y_{ij}^l$, para calcular o gradiente da função de erro, $\partial \mathcal{E} / \partial x_{ij}^l$, basta calcular a derivada da função de ativação $\sigma'(x)$.

A propagação dos erros para trás pode ser mostrada através de mais um uso da regra da cadeia.

$$\frac{\partial \mathcal{E}}{\partial x_{ij}^{l-1}} = \sum_{a=0}^{m_1-1} \sum_{b=0}^{m_2-1} \frac{\partial \mathcal{E}}{\partial x_{(i-a)(j-b)}^l} \frac{\partial x_{(i-a)(j-b)}^l}{\partial y_{ij}^{l-1}} \quad (2.22)$$

A partir da Equação 2.17 pode-se inferir

$$w_{ab} = \frac{\partial x_{(i-a)(j-b)}^l}{\partial y_{ij}^{l-1}} \quad (2.23)$$

Utilizando esse resultado substituindo em 2.22 chega-se ao seguinte

$$\frac{\partial \mathcal{E}}{\partial x_{ij}^{l-1}} = \sum_{a=0}^{m_1-1} \sum_{b=0}^{m_2-1} \frac{\partial x_{(i-a)(j-b)}^l}{\partial y_{ij}^{l-1}} \quad (2.24)$$

Esses resultados, não tanto, só fazem sentido para entradas que estão m_1 pontos de distância das laterais esquerda da matriz de entrada m_2 pontos do topo da mesma matriz. Para resolver esse problema, computacionalmente se acrescenta um *padding* de zeros nas entradas.

2.4 Uso de Entropia como Função de Erro

Comumente, a função de erro utilizada no treinamento de ANNs é

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \quad (2.25)$$

Entretanto, podemos substituí-la pela função de entropia cruzada do sistema

$$\mathcal{J} = -\frac{1}{n} \sum_x [d \ln y + (1-d) \ln(1-y)] \quad (2.26)$$

com d sendo o valor desejado y a saída do sistema. Essa escolha ocorre devido ao fato da propagação de erro na entropia cruzada não depender da derivada da função de ativação, ou seja, não ocorre uma diminuição da aprendizagem quando o gradiente tende a zero. Não é trivial a conclusão que essa fórmula pode ser utilizada como uma função de erro. São duas propriedades que levam a essa conclusão.

A primeira é que a função não é negativa, ou seja $\mathcal{J} \geq 0$, isso é evidente pois o resultado do somatório é sempre negativo já que ambos os logaritmos estão entre 0 e 1. A segunda propriedade é que o valor da entropia cruzada decresce monotonicamente de acordo com o decrescimento do módulo do erro.

Na Equação 2.26, substituindo y por $\sigma(s)$ e aplicando a regra da cadeia temos

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial w_j} &= -\frac{1}{n} \sum_x \left(\frac{d}{\sigma(s)} - \frac{(1-d)}{1-\sigma(s)} \right) \frac{\partial \sigma}{\partial w_j} \\ &= -\frac{1}{n} \sum_x \left(\frac{d}{\sigma(s)} - \frac{(1-d)}{1-\sigma(s)} \right) \sigma'(s) x_j \end{aligned} \quad (2.27)$$

que pode ser simplificada a

$$\frac{\partial \mathcal{J}}{\partial w_j} = \frac{1}{n} \sum_x \frac{\sigma'(s) x_j}{\sigma(s)(1-\sigma(s))} (\sigma(s) - d) \quad (2.28)$$

Para uma função de ativação logística

$$\sigma(s) = \frac{1}{1 + e^{-s}} \quad (2.29)$$

A derivada da função é simples, como visto a seguir

$$\begin{aligned} \frac{d\sigma(s)}{ds} &= \frac{e^{-s}}{(1 + e^{-s})^2} \\ &= \left(\frac{1 + e^{-s} - 1}{1 + e^{-s}} \right) \left(\frac{1}{1 + e^{-s}} \right) \\ &= (1 - \sigma(s)) \sigma(s) \\ \sigma'(s) &= \sigma(s)(1 - \sigma(s)) \end{aligned} \quad (2.30)$$

Qu , substituindo na Equação 2.28 encontra-se [13]

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{n} \sum_x x_j (\sigma(s) - d) \quad (2.31)$$

Capítulo 3

Desenvolvimento

"If a machine is expected to be infallible, it cannot also be intelligent." - Alan Turing

O projeto consiste em utilizar dados do mercado de popularidade de determinados tópicos no sistema de busca da Google Inc. para estimar aumentos ou quedas nos valores de ações da Petrobras S. A.

O trabalho realizado pode ser dividido em duas etapas principais: a primeira é a aquisição de dados da plataforma Google Trends e dados de ações da Petrobras e a segunda etapa é a programação/aplicação dos dados na rede.

3.1 Aquisição de Dados

3.1.1 Aquisição de dados do Google Trends

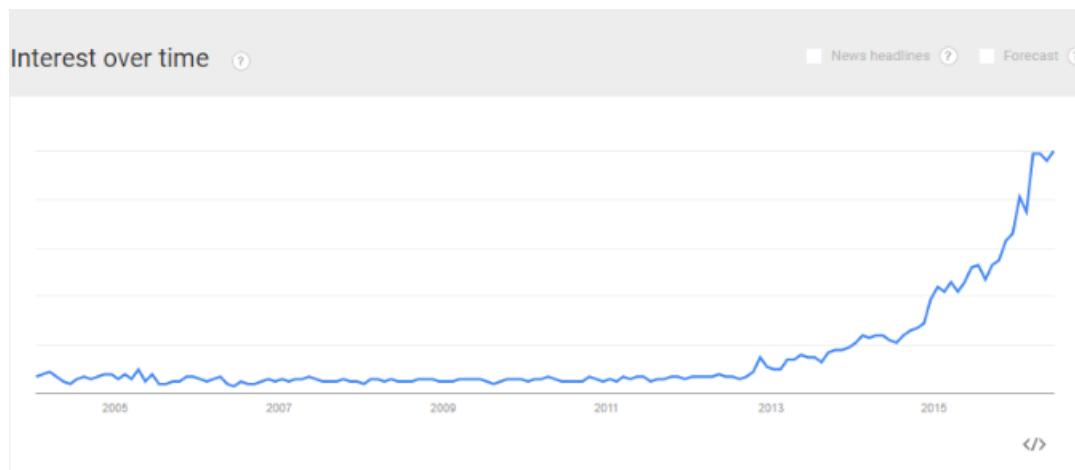


Figura 3.1: Gráfico de popularidade do termo *deep learning* nos últimos anos

O Google Trends é uma plataforma online fornecida pela Google Inc. que oferece dados sobre a popularidade de pesquisas no buscador da empresa. Os dados são oferecidos através de um gráfico, visível online, ou através de uma tabela, na extensão .csv disponível para download. Para obter

os dados, basta selecionar o período para pesquisar o tema de interesse. Um exemplo está disponível na Figura 3.1, aonde o tema pesquisado foi *Deep Learning* o período foi de Janeiro de 2004 até Junho de 2016.

Os dados disponíveis contêm a informação normalizada em uma escala de zero a cem para o período de interesse selecionado, sendo sempre com o dia de maior número de pesquisas, ou seja, os dados representam uma porcentagem em relação ao dia de maior valor.

Foram escolhidos arbitrariamente os temas de interesse, totalizando vinte:

- “ajust fiscal”
- “corrupção”
- “dólar”
- “Dilma”
- “gasolina”
- “imposto”
- “índice Bovespa”
- “lava jato”
- “mercado financeiro”
- “Odebrecht”
- “OGX”
- “ouro”
- “PETR4”
- “Petrobras”
- “plataforma de petróleo”
- “pré sal”
- “PSDB”
- “PT”
- “Selic”
- “tesouro direto”

A escala temporal do gráfico de período selecionado, sendo possível ter uma precisão de até um minuto para tópicos populares em períodos recorrentes curtos (menos de uma hora). Para a realização do trabalho, é de interesse dados fornecidos diariamente. Dados com um período diário só estão disponíveis caso o período selecionado seja de três meses ou menos.

Cada um contendo os dados diários normalizados de acordo com o período de Janeiro de 2011 até Maio de 2016. Existem, portanto, uma necessidade de capturar diversos períodos de dados, de três meses cada, para cada um dos temas de pesquisa uni-los.

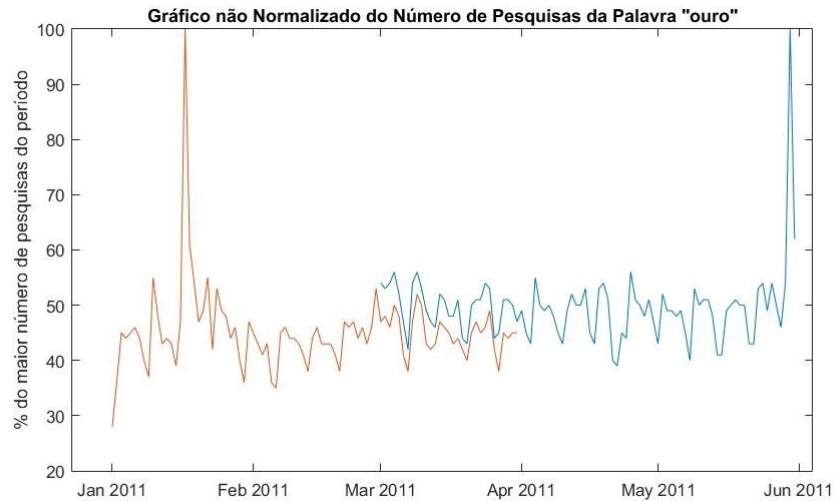


Figura 3.2: Gráfico exemplo de dados não ajustados

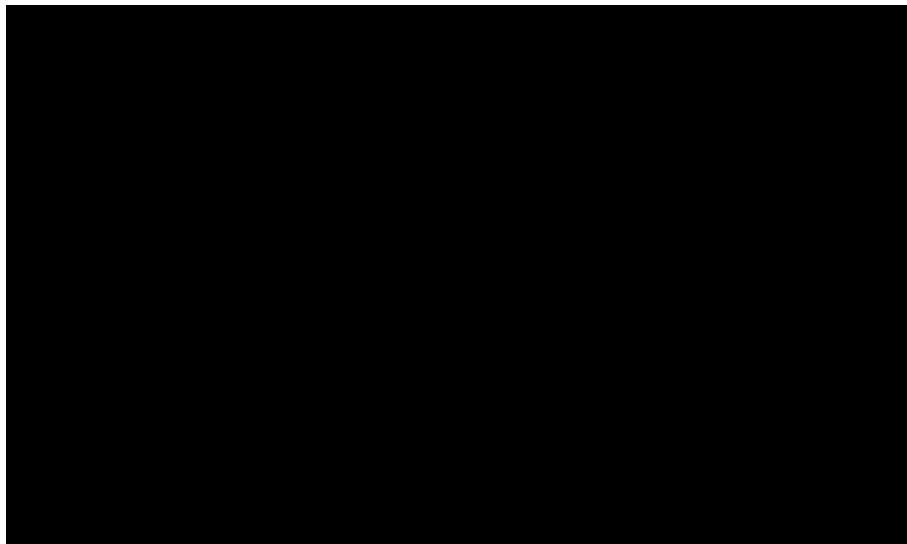


Figura 3.3: Gráfico exemplo de dados ajustados

Como cada conjunto de dados está normalizado para o período que lhe é pertinente, os dados foram capturados com uma intenção intrínseca no domínio do tempo. Assim, sabendo os valores finais de cada conjunto de dados comparando-os com os primeiros dados do conjunto seguinte, é possível fazer uma nova normalização, novamente de acordo com o conjunto de dados.

concatenados. Na Figura 3.2 pode-se ver um exemplo de um gráfico não ajustado. Nota-se que a parte laranja do gráfico está deslocada para cima em relação à azul. Após realizar o ajuste dos dados pode-se observar como fica o resultado na Figura 3.3.

Após esse pré-processamento dos dados, a série temporal final, para cada tópico, se inicia em Janeiro de 2011 e termina em Março de 2016. Para todo esse período, considerando as intervalos entre o limite máximo de três meses para cada aquisição de dados, são 31 arquivos no total, somando um total de 620 arquivos. Para capturar todos esses dados processá-los, foram necessários dois *scripts* diferentes, um para capturar os dados do Google Trends, outro para importar os arquivos para o software Matlab automaticamente uni-los e normalizá-los.

3.1.2 Aquisição de dados da bolsa de valores

Além dos dados do Google Trends, para fazer estimativas em cima da bolsa de valores é importante ter o histórico daquela ação específica. Os dados utilizados foram os referentes à PETR4, que são ações da Petrobras S. A. com preferência na distribuição de dividendos. As informações contidas no histórico diário são os valores de abertura, os valores de fechamento, ou cotação, os máximos e mínimos, o volume diário. Esses dados estão disponíveis em diversos sites.

Para esse trabalho, foram utilizados dados extraídos da plataforma de dados econômicos da UOL, pois além de oferecer o mesmo banco de dados distribuído pela BMF&Bovespa, mas de forma organizada, em uma planilha para download. Os dados desse histórico, assim como os históricos finais já pré-processados do Google Trends, começam em Janeiro de 2011 e vão até Março de 2016. Esses dados também precisam ser normalizados. Os valores de volume, por exemplo, são muito superiores a todos os outros valores.

Após a normalização, para implementação nos algoritmos de aprendizado de máquina com os dados do Google Trends, as séries da bolsa de valores não podem ter dados faltando. Os dados financeiros são disponibilizados, normalmente, como uma série temporal com dados faltando nas datas em que o preço está fechado. Assim foi necessário um *script* que detecta datas faltando nos históricos, como finais de semana feriados, inserindo essas datas nas séries, inserindo o valor 0 para usar respectivos valores de mercado, visto na Figura 3.4. Posteriormente, durante o pré-processamento, esses valores iguais a 0 são substituídos por outros valores de interesse.

3.2 Implementação das Redes Neurais

Foram utilizadas duas técnicas de redes neurais artificiais para tentar estimar o preço de valor das ações vão subir, ou se elas vão descer. A primeira técnica utilizada foi um perceptron multicamada, a segunda, uma rede neural convolucional. Para ambos os casos, as entradas das redes foram as séries temporais extraídas do Google Trends e os históricos de valores de mercado, somando um total de 25 séries temporais, cinco das quais são os dados de mercado e o resto do Google Trends. Por fim, além das redes, também foi utilizado um estimador ingênuo.

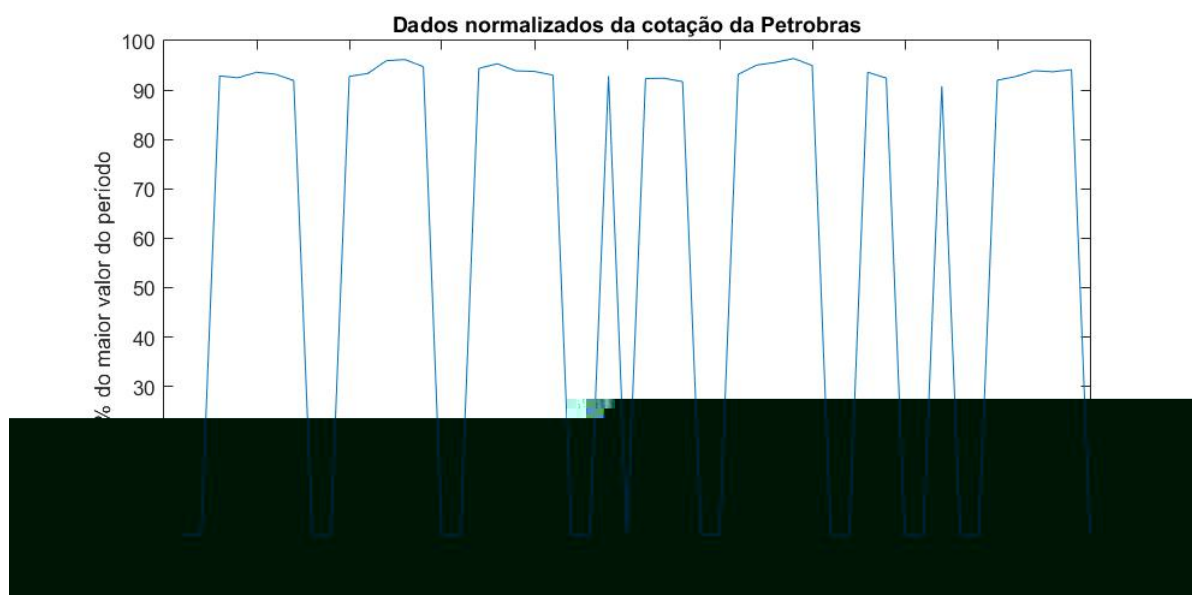


Figura 3.4: Ex mplo d cotação após normalização pr nchimento com 0

3.2.1 Perceptron Multicamada

O p rc ptron multicamada impl m ntado possui uma camada d ntrada qu contém os dados d stados ant rior s da séri multivariada. Possui duas camadas scondidas uma camada d saída. Todas as camadas são compl tam nt con ctadas, ou s ja, cada um d s us n urônios r c b dados d todos os n urônios da camada ant rior t m sua saída como ntrada d todos os n urônios da próxima camada, com xc ção da prim ira da última, qu não têm camada ant rior n m próxima, r sp ctivam nt .

Os dados d ntrada, como dito ant riorm nt , são as séri s t mporais adquiridas. Para stimar o valor futuro, ins rir todo o histórico prévio àqu l valor, além d s r p sado computacionalm nt , é impraticáv l. N c ssitaria da ins rção d dados na matriz d p sos a cada it ração já qu o núm ro d ntradas aum nta. Isso implicaria nos dados s r m tratados d forma irr gular: os prim iros p sos s riam tr inados durant mais it ração s qu os últimos. Foi d cidido criar um p ríodo fixo, prévio ao stado qu s qu r stimar, m qu todos os dados d todas as séri s s rviriam d ntrada.

Ess p ríodo foi d t rminado através da corr lação cruzada ntr as séri s do Googl Tr nds o valor das açõ s. A corr lação cruzada r pr s nta a corr lação ntr duas séri s discr tas no t mpo. Ela m d a similaridad ntr uma séri d int r ss x (no caso o valor das açõ s da P trobras) copias d uma séri y (os dados do Googl Tr nds) d slocados no t mpo. Pod -s t star a corr lação d slocada para fr nt ou para trás no t mpo, com um d slocamento máximo igual ao tamanho das próprias séri s. Como as séri s t m 1907 valor s, t stando todas as corr laçõ s possív is, obtém-s 3814 valor s d saída, para cada um dos -1907 à 1907 d slocamentos no t mpo. O Apêndic III contém os gráficos das corr laçõ s cruzadas.

A d t rminação foi arbitrária após uma anális gráfica dos dados, não havia grand corr lação ntr o valor a s r stimado valor s mais distant s qu 150 pontos, a fr nt ou atrás no t mpo.

Considerando essa informação, foi selecionado um período de 150 dias para servir de entrada para a rede, para cada estimação. Para o dia 701, por exemplo, a entrada seria do dia 550 ao dia 700. Como são 25 séries, 150 dias de cada uma, para cada estimação entram no sistema, portanto, um total de 3750 valores de entrada.

Para essa quantidade de entradas foram encontradas duas principais dificuldades. A primeira, mais evidente, é a demora no processamento, que leva horas para cada conjunto de épocas treinadas. A segunda é o *overfitting*. A taxa de acerto nos dados de treinamento chega a valores próximos de 100%. Já nos arquivos de validação, o resultado ficava, com consistência, próximo de 50% de acerto, o que, para um sistema com saída binária, é o mesmo resultado esperado de uma decisão a partir de uma moeda não viciada. É um resultado insatisfatório.

Foram testados vários outros valores possíveis para o tamanho de entrada. Foi notória uma melhoria no resultado com a diminuição do número de dias de entrada. Seguindo essa tendência, os melhores resultados obtidos foram para uma entrada de apenas os últimos dois dias antes do dia de interesse.

Outro fator importante no tratamento da entrada é um pré-processamento dos dados. Todos os valores de entrada são normalizados para um valor relativo entre 0 e 1. Como o acesso da população à internet aumentou nos últimos anos, é esperado que a média das pesquisas das últimas tenha aumentado também. Consequentemente, os dados distribuídos de maneira cíclica contêm uma informação que é difícil para a rede aprender, pois a entrada para cada dado não representa todo o histórico, mas apenas os últimos dias. A entrada deve conter apenas a informação da variação de popularidade de um tópico, não a popularidade absoluta. Todos os dados são ajustados segundo as equações 3.1 e 3.2.

$$\Delta V(t) = V(t) - V(t - 1) \quad (3.1)$$

$$\Delta V(0) = 0 \quad (3.2)$$

Onde $V(t)$ é o valor de cada série na data t , $\Delta V(t)$ é o valor da variação na data. Os dados de entrada $V(t)$ são substituídos por $\Delta V(t)$.

O último ajuste na entrada foi testar e mover algumas das séries temporais para a esquerda do número de acertos da saída do sistema. Percebemos que o resultado melhorava ao retirar as séries relevantes aos seguintes tópicos:

- "Dígitos"
- "Corrupção"
- "Ajuste Fiscal"
- "Mercado Financeiro"
- "Índice Bovespa".

A rede utilizada possui entrada, duas camadas escondidas e uma camada de saída. O primeiro parâmetro a determinar é o número de neurônios na camada de saída. Esse valor deve corresponder à quantidade de dados de saída desejados para cada iteração.

Inicialmente, foram testados três neurônios de saída para a rede, uma para caso a cotação tivesse subido ou tivesse permanecido igual, uma saída para caso a cotação tenha diminuído em relação ao dia anterior e uma terceira saída, que deveria identificar quando os próximos valores do mercado seriam zero, que são os casos de finais de semana e feriados. O neurônio que tivesse o maior valor de saída é considerado ganhador daquela iteração, atribuindo o valor de 1 em sua respectiva posição em um vetor de três bits. Esse vetor é comparado com a *label* daquela iteração, caso fossem iguais, é considerado um acerto.

Essa configuração inicial não resultou em uma boa taxa de acertos, para 50 neurônios em cada uma das camadas escondidas com os dados de entrada já processados, obtivemos taxas de acerto próximas de 33%, o que é esperado para uma saída aleatória em um sistema neurônio. Consequentemente, essa configuração de saída foi descartada logo no início dos testes e foi substituída por uma saída binária, com dois neurônios. O primeiro para caso a cotação tenha subido ou se mantido estável, o segundo para caso a cotação tenha caído. Para ser possível ter uma saída com dois neurônios, os dados que foram definidos como zero foram substituídos pelo último valor não zero. Fazendo uma interpolação simples, como visto na Figura 3.5. Os valores iniciais das matrizes de pesos de cada camada são randomizados segundo uma distribuição uniforme, entre -1 e 1.

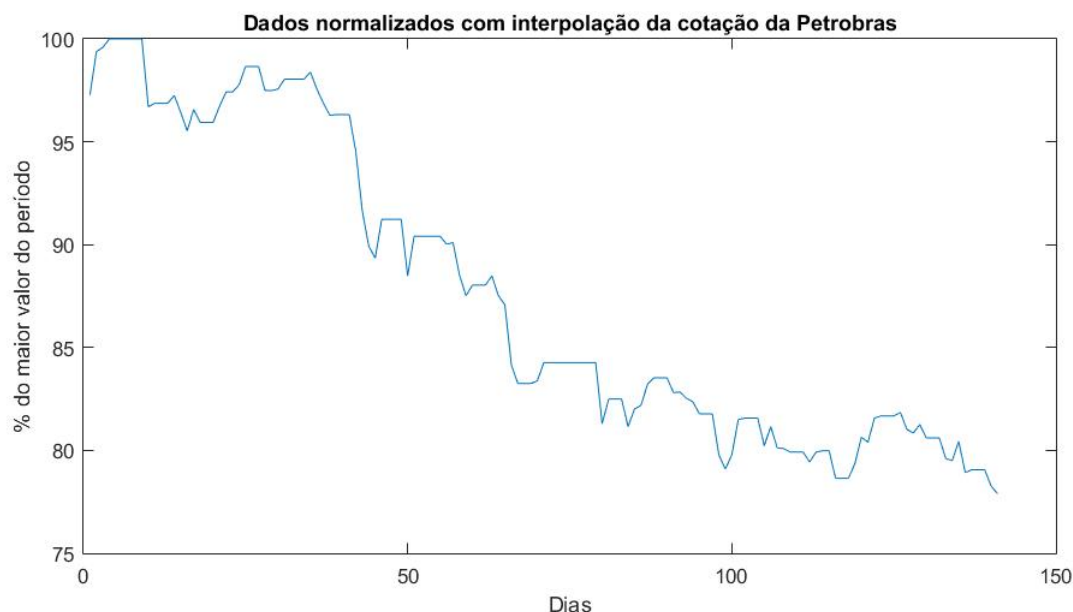


Figura 3.5: Cotação da Petrobras com interpolação durante um período de 150 dias

Adicionalmente, outros parâmetros da rede foram testados, em sua maioria, de forma empírica. Variamos os parâmetros até encontrar uma composição cujo resultado fosse satisfatório.

O número de neurônios nas camadas escondidas foi definido como 50. Foram testadas uma quantidade de neurônios de 20 a 100 variando de dez em dez, para cada uma das camadas in-

dividualmente, o melhor resultado obtido foi para 50 neurônios em cada camada. A taxa de aprendizagem, η , também foi determinada empiricamente. Comçando em um, foram testados diferentes valores para η de forma decrescente até encontrar um valor no qual há a queda mais rápida do erro quadrático médio sem que haja oscilação nos valores finais. As Figuras 3.6 e 3.7 representam as variações dos erros quadráticos médios para um η igual a 1 e 0.5, respectivamente.

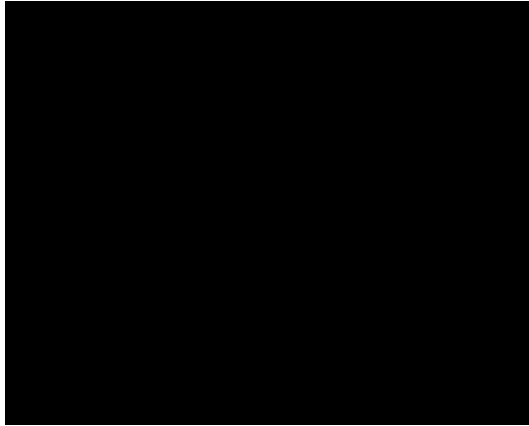


Figura 3.6: Erro quadrático médio nos dados de treinamento, por época, para um η igual a 1.



Figura 3.7: Erro quadrático médio nos dados de treinamento, por época, para um η igual a 0.5.

Para um η de 0.5, na Figura 3.7, são notórias a monotonicidade e a suavidade na queda do erro quadrático médio, para η igual a 1, como o da Figura 3.6, essas propriedades não são presentes. Os resultados se comportaram de acordo com o esperado em relação ao η . Quanto maior, mais rápida a queda, contudo mais instáveis são os resultados; quanto menor, mais devagar é a queda, mas mais estáveis são os valores finais. Após diversas experimentações, foi perceptível que, para esse determinado sistema, qualquer η menor ou igual 0.5 já encontrava estabilidade em valores finais resultava em taxas de acerto similares. O valor escolhido para η foi, consequentemente, 0.5.

Além dos diferentes valores iniciais para η , também foi testado um η variável, com um valor mínimo para cada iteração. Diferentes taxas de queda foram testadas para η , mas nenhuma resultou em diferenças estatisticamente significantes em relação a um η fixo.

Foram testados dois diferentes tipos de função de ativação: a função logística e a tangente hiperbólica. Nesse trabalho, a função logística teve resultados melhores que os da função tangente hiperbólica.

3.2.2 Rede Neural Convolutional

A CNN desenvolvida é composta em 10 camadas. A primeira é a camada de entrada que, como sempre, contém os dados de entrada do sistema. As próximas 4 camadas de convolução e 4 de *pooling*, dispostas de forma intercalada, comçando por uma convolução seguida de *pooling*. Por fim, vem uma camada de saída completamente conectada.

Essa estrutura é mais simples, além de ter uma maior complexidade de implementação do que o perceptron multicamada, requer também mais capacidade computacional. Para treinamentos com o mesmo número de épocas para o mesmo banco de dados, o tempo de execução do treinamento dessa rede é na ordem de 10^2 vezes o tempo do perceptron multicamada implementado. A Figura 3.8 mostra a estrutura da rede utilizada.

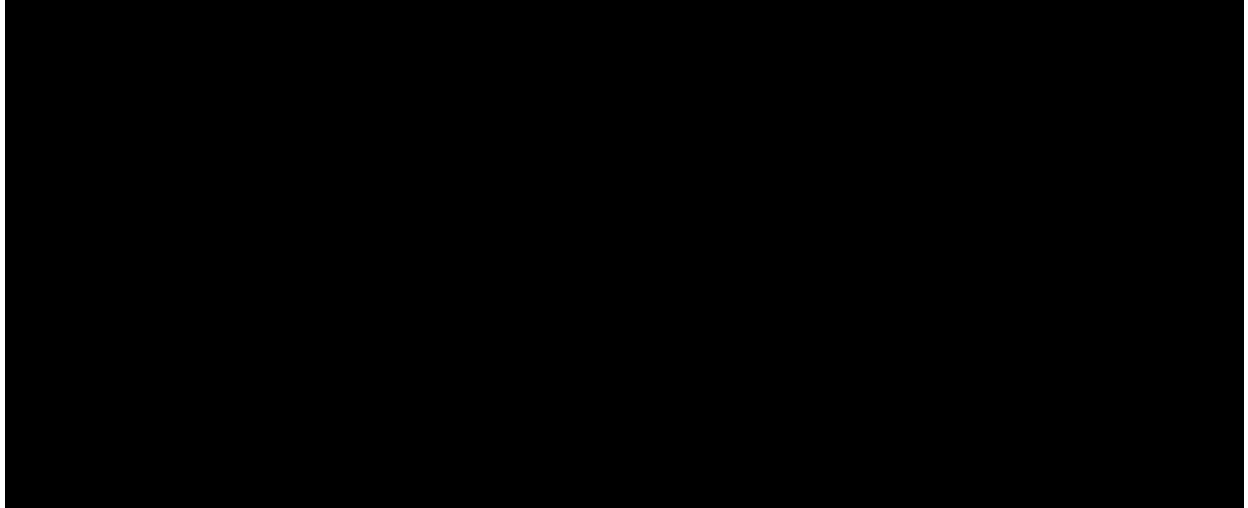


Figura 3.8: Disposição da CNN utilizada

Tratar séries temporais com CNNs é similar a problemas de classificação ou detecção de objetos em imagens. A diferença ocorre apenas na forma que os neurônios vão receber suas entradas. Uma imagem de entrada é representada através de uma matriz $N \times N$, da qual, na camada de convolução, um filtro $m \times m$ percorre a imagem fazendo a convolução de grupos retangulares de pixels para a próxima camada. No caso de uma ou mais séries temporais, o filtro é um vetor de comprimento m — o *pooling*, ao invés de ocorrer em uma área retangular, ocorre só em cima de um trecho da série que está sendo amostrada.

A disposição dos vetores de entrada da CNN implementada está em formato matricial, sendo cada linha uma das séries de interesse. As razões para a estruturação dos dados dessa forma é a possibilidade de realizar operações matriciais com todo o conjunto de entradas simultaneamente. Embora aumente a complexidade do código, isso reduz, significativamente, o tempo de execução.

Assim como outras ANNs, a rede convolucional precisa de uma detriminação prévia de suas condições iniciais. Como η , número de neurônios de saída, tamanho da janela de convolução e tamanho da janela de *pooling*.

As janelas de pesos usados como filtros na camada de convolução são gerados aleatoriamente no início de cada execução. A faixa na qual os pesos sinápticos são gerados foi definida arbitrariamente, após alguns testes, foi definida entre -0.1 e 0.1 .

As mesmas quantidades de neurônios de saída que foram testadas para o perceptron multicamada foram testadas para a camada completa contida da CNN, que também é a camada de saída. Contudo, nenhuma das duas quantidades de saída obteve resultados satisfatórios.

Inicialmente, foi decidido arbitrariamente que haveria três neurônios de saída, um para quando o valor do mercado subiu no último dia, um para quando o valor do mercado desceu no último dia e um para quando ficou estável ou é final de semana. Essa tentativa não rendeu bons resultados, a rede não aprendia a identificar quando era final de semana ficava presa em mínimos locais óbvios. Quando, no período usado para treinamento, havia mais subidas do que descidas na cotação, a rede aprendia a sempre fazer a mesma estimativa: de que o valor iria subir. O mesmo aconteceu para quando a maioria dos dados descia, a rede aprendia que o valor sempre iria descer.

A segunda tentativa foi com dois neurônios de saída, o número de neurônios que obtiveram o melhor resultado para o perceptron multicamada. Para esse número de neurônios, assim como no perceptron, foi necessário fazer uma interpolação dos dados do mercado dos finais de semana feriados. Quando era final de semana ou feriado, o dado do último dia útil era perdido, implicando em ter uma variação diária igual a zero. Os testes com duas saídas obtiveram o mesmo problema com mínimos locais que os testes com três. Quando o valor das ações subia ou descia, na média, a rede ficava presa na estimativa de que o mercado sempre iria subir, ou descer, correspondente ao mesmo.

Para resolver esse problema de mínimos locais, foram testadas três abordagens. A utilização de *momentum* foi a primeira delas. Com a inclusão de esse parâmetro, o ajuste dos pesos não dependia só do erro da iteração, mas também de uma porcentagem do ajuste da última iteração. Essa porcentagem é determinada por um valor α , como pode ser visto na equação 2.16. Dessa forma, se o gradiente descer para um mínimo local, os ajustes dos pesos teriam uma "inércia" que mantinha a variação naquela direção. Diversos valores de α foram testados e resultaram em uma aprendizagem mais rápida, mas continuaram com o mesmo problema de mínimos locais.

A segunda abordagem foi a inclusão de ruído gaussiano nos pesos a cada iteração. Essa foi a abordagem com pior resultado, a rede não ficava presa em mínimos locais, mas em contrapartida não aprendia nada. Mesmo a inclusão de pesos nos ruídos com uma ordem de grandeza estimada de 10^{-3} em relação aos pesos da rede, não havia queda perceptível no gradiente.

A última abordagem testada foi a inclusão de ruído branco gaussiano nos vetores de entrada. Inicialmente essa tentativa não teve um resultado satisfatório. Entretanto, mantendo esse ruído na entrada, após ajustes descritos futuramente, a rede foi capaz de aprender a se manter presa em mínimos locais óbvios.

O último ajuste no número de neurônios foi a redução para apenas um. Essa saída implementada é diferente das saídas com múltiplos neurônios. Ao invés de ter uma saída representativa da subida ou queda no valor da cotação, com apenas uma saída é necessário determinar um limiar arbitrário no seu valor. Para qualquer valor de saída acima desse limiar, a estimativa é de subida, para qualquer valor abaixo desse limiar, a estimativa é de descida.

Para a rede convolucional, mesmo com a inclusão de ruído com apenas uma saída, não foi possível observar bons resultados utilizando a função de erro EQM (Eq. 2.25). Essa função foi substituída pela entropia cruzada (Eq. 2.26) e obtiveram resultados significativamente melhores.

Durante todo o processo de ajuste dos parâmetros, diversos valores para η foram testados. Os melhores resultados foram para $\eta = 0.001$. Como cada neurônio pré-convolução influencia um

conjunto grande de neurônios da camada seguinte, a soma do erro propagado para trás tendo a ser grande em relação aos pesos sinápticos. Portanto, para controlar variações grandes nos pesos, é necessário um η pequeno.

Capítulo 4

Resultados

"The most exciting phrase to hear in science, the one that heralds the most discoveries, is not "Eureka!" (I found it!) but "That's funny..." - Isaac Asimov

Nos capítulos são apresentados descritos os resultados dos treinamentos realizados. Os dados foram divididos com os primeiros 94% sendo treinamento e os últimos 6% sendo validação. Essa divisão foi tomada devido ao fato de que alguns termos extraídos do Google Trends só se tornaram populares significativos para os valores das ações da Petrobras no final de 2015 início de 2016.

Antes de 2014 a correlação entre a popularidade do termo "lava jato" e o valor da ação PETR4 era insignificante, por exemplo. No final de 2015 início de 2016 a correlação entre as duas aumentou. Caso a segmentação entre treinamento e validação reversasse um período maior para validação, não haveria período algum nos dados de treinamento que a correlação entre "lava jato" e o valor da PETR4 fosse estatisticamente significativa. Nesse cenário, os dados do termo "lava jato" não seriam importantes para o treinamento, pois a ideia seria perceber que não há nenhuma correlação entre os dois. Também não seriam importantes para a validação pois a ideia seria aprender que os dados não são importantes para o mercado, o que não é verdade.

4.1 Perceptron Multicamada

Ao contrário do esperado, o perceptron multicamada foi a técnica implementada que obteve melhores resultados. A execução do perceptron também é mais rápida que a execução da CNN. O banco de dados utilizado nos treinamentos e validações são as séries temporais dos termos citados na Seção 3.1.1 os dados de mercado da Petrobras.

Durante as 10 primeiras épocas de treino, a taxa média de acerto dos dados de validação foi de 61.19%, a taxa de acerto médio no treinamento foi de 69.43% e o tempo médio de treino foi de 11.10 s. Para as 10 próximas épocas, a taxa de acerto dos dados de validação média aumentou para 65.52%, a taxa de acerto médio no treinamento foi de 73.81% e o tempo médio de treino foi de 22.18 s. Nas vinte próximas épocas, chegando a 40 épocas de treino, a taxa de acerto dos dados de validação média foi de 62.91%, a taxa de acerto médio no treinamento foi para 84.52%, o tempo continuou aumentando linearmente em relação ao número de épocas executadas. Ao totalizar 80 épocas, a taxa de acerto dos dados de validação caiu ainda mais, chegando a média foi de 62.24%, a taxa de acerto médio no treinamento foi de 93.87% e o tempo final médio de treino foi de 88.10s.

A Figura 4.1 é gráfico do erro quadrático médio (EQM) por época referente ao treino das primeiras 80 épocas.

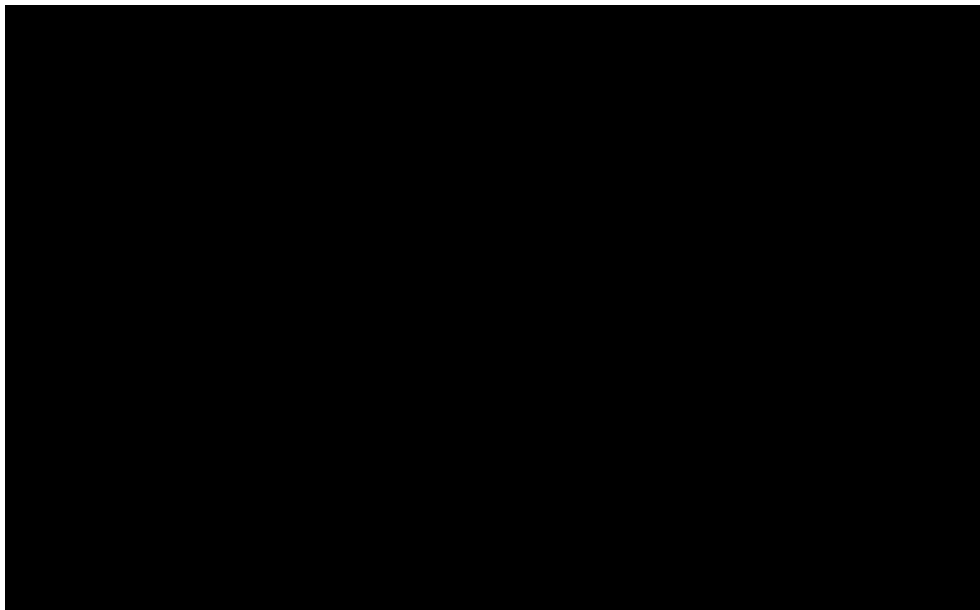


Figura 4.1: Gráfico da queda do EQM para 80 épocas

4.1.1 Discussão Sobre os Resultados do Perceptron Multicamada

Os resultados obtidos para o perceptron multicamada foram melhores do que o esperado. Os melhores desempenhos obtiveram 68.22% de acerto nos dados de validação.

Com o aumento no número de épocas, aumentou também a taxa de acerto nos dados de treinamento, monotonicamente. Esse resultado é esperado. É com base no banco de dados de treinamento que ocorrem os ajustes na rede.

Terminando a rede por 80 épocas, como visto na Figura 4.1, obtém-se uma taxa de acerto média de quase 94%. A taxa de acerto média para um treinamento de 20 épocas é de apenas 73.81%. Essa diferença, somada com uma queda na quantidade média de acertos nos dados de validação para esse período de épocas, caracteriza um caso de *overfitting*. O número de acertos para os dados

de treinamento aumentava com o número de épocas. Entretanto, o número de acertos da validação aumentava apenas no começo após algumas épocas, com isso a cair devido ao *overfitting* do sistema.

Usando os parâmetros selecionados durante o desenvolvimento, os melhores resultados de validação para a rede perceptron multicamada ocorriam, normalmente, por volta de 20 épocas. Número de épocas menores que 20 normalmente resultam em treinamentos *underfitted*, mostrando que a rede não aprendeu tudo o que poderia para fazer estimativas sobre os dados de validação. Treinamentos com dezessete épocas a mais que 20 implicavam em *overfitting*. O treinamento ajustava os pesos tão bem para o período de treinamento do modo que as previsões só funcionavam para ele.

Um parâmetro que influencia significativamente no resultado valioso mencionado é o número de dias passados que foram usados como entrada. A estimativa inicial de 150 dados não funcionou pois o impacto que as notícias têm no valor da ação provava-lhe não gerarmos resultados duradouros por período. Embora exista a correlação entre os dados, como visto no Apêndice III, a correlação mais forte é nos dias imediatamente após a publicação da notícia. Um período longo de 150 dias contém, junto com informações importantes para o modo, informações que são apenas ruído para a rede, dificultando o aprendizado. O melhor resultado obtido foi para um número de dias anteriores de entrada igual a 2.

4.2 Rede Neural Convolucional

Era esperado que a CNN implementada obtivesse resultados melhores que o perceptron multicamada. É um tipo de rede que tem obtido resultados estado da arte em trabalhos envolvendo processamento de imagens, por exemplo [14]. A razão do uso dessa técnica era a expectativa de que, assim como a rede funciona para detecção de *features* em imagens, ela fosse capaz de detectar *features* nas séries temporais correlacionadas. Embora tenham obtido ótimos resultados, redes neurais convolucionais, por serem profundas, têm muitas iterações na propagação dos dados em cada camada, têm uma complexidade computacional maior que perceptrons multicamada. A rede implementada, para um treinamento de uma época com todo o banco de dados, demora consistentemente mais de 7 minutos, sendo que o tempo de execução é centenas de vezes o tempo de treinamento de uma época do perceptron multicamada.

A CNN, ao contrário do que era esperado, teve um resultado pior do que o perceptron. A rede convolucional desenvolvida obtinha, para um treinamento com 200 épocas, uma taxa de acerto para os dados de validação de 59.81%, para os dados de treinamento, de 64.60%. Além de ter um resultado pior, o tempo de execução, como dito anteriormente, é mais extenso. O treinamento para 200 épocas demorou dois dias, como pode ser visto no gráfico da Figura 4.2, o erro quadrático médio ainda estava aumentando. É possível para um treinamento mais longo, com mais épocas, a taxa de acerto da CNN aumentasse, em troca de mais tempo de execução.

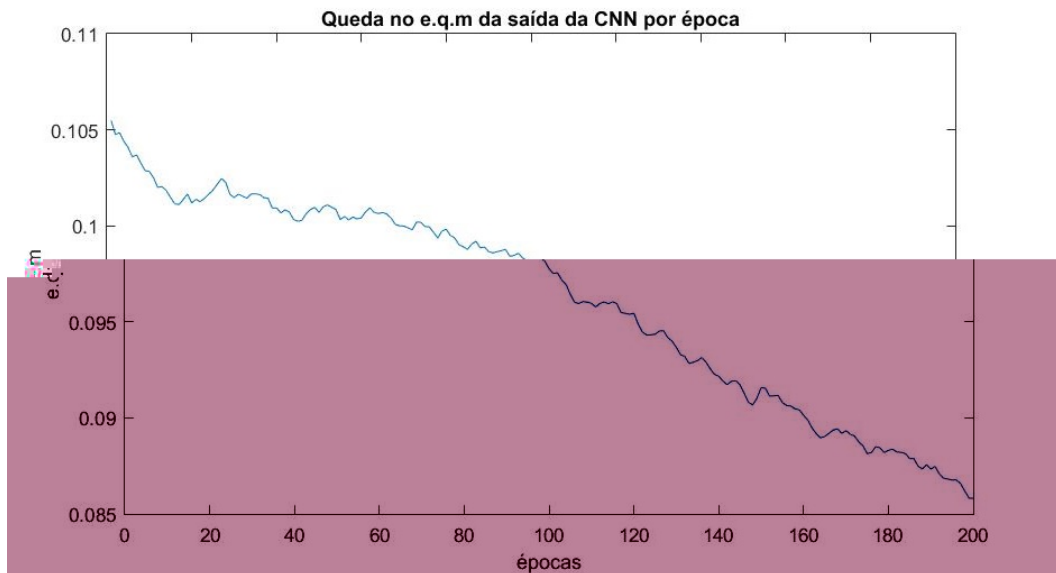


Figura 4.2: Qu da no EQM por época d tr inam nto da CNN

Além d uma simulação com 200 épocas, também foi r alizada uma simulação com 500. A x cução para 500 épocas durou 4 dias. Entr tanto, não houv estabilização do rro quadrático médio. Ao concluir o tr inam nto, o rro ainda stava m qu da. Para ssa quantidade d épocas tr inadas, a taxa d ac rto para o arquivo d validação foi a m sma qu para 200 épocas, 59.81%.

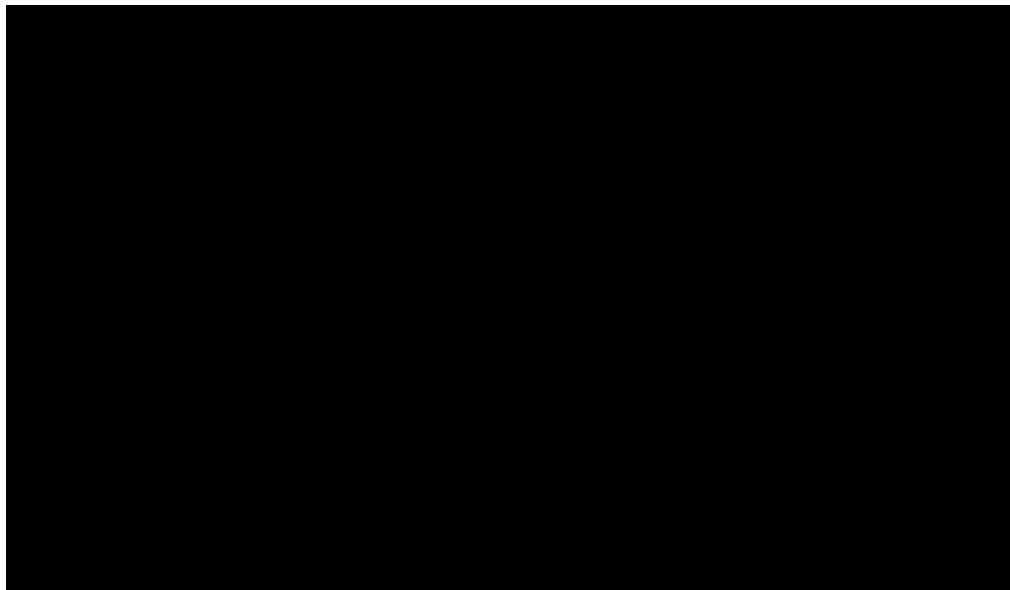


Figura 4.3: Qu da no EQM por época d tr inam nto da CNN

4.3 Comparação dos Resultados

Para analisar o r sultado das r d s n urais artificiais impl m ntadas, além d ssas duas técnicas foi t stado um classificador ingênuo nos dados d validação. Ess simpl s classificador stima o

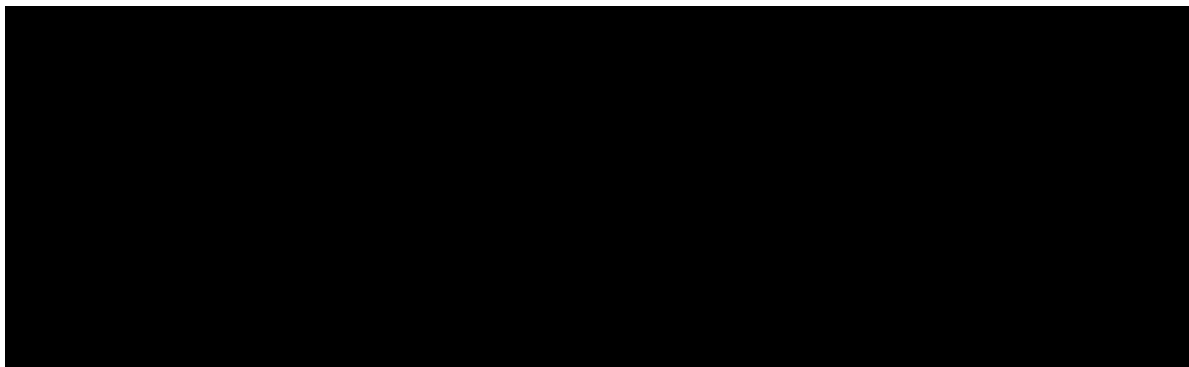


Figura 4.4: Variação percentual do valor da ação PETR4 em relação ao dia anterior para todo o banco de dados

estado de um dia basando-se no dia anterior a ele, caso o valor da ação tenha subido, ele estimará uma subida, caso tenha descido, ele estimará uma descida. O gráfico da Figura 4.4 mostra a variação diária do valor da ação. Todo dia que o valor no gráfico for positivo, significa que o valor da ação aumentou, toda vez que for negativo, significa que o valor da ação diminuiu. Através desses dados é fácil ver o resultado do classificador ingênuo.

Os dados de validação são os últimos 107 dias da série temporal vista na Figura 4.4. O classificador ingênuo teve uma taxa de acerto de 50.72% para esses dados. Ambas as redes neurais artificiais implementadas foram capazes de lidar com o conjunto de dados de entrada encontrando um resultado melhor que o classificador ingênuo. A melhor taxa de acerto do perceptron multicamada foi de aproximadamente 68.22%. O resultado obtido pela CNN foi de uma taxa de acerto para o arquivo de validação igual a 59.81%. É notório que as duas redes, para esse período, não só lidaram com os dados fornecidos como também obtiveram resultados acima de um classificador não treinado.

Comparando os resultados do perceptron multicamada da CNN é evidente que o perceptron multicamada obtiveram resultados melhores. Essa afirmativa, no entanto, não pode ser generalizada. Ela só pode ser feita para essas redes, com seus parâmetros específicos com esse banco de dados.

Como, para esse problema específico, o perceptron multicamada obtiveram resultados melhores, foram realizadas simulações de investimento utilizando-o como critério. As simulações executadas seguem as estimativas do perceptron para o período de validação de acordo com as seguintes regras:

1. Inicia-se a simulação com R\$ 1000.
2. Caso a rede previja que o valor vai subir toda a quantia de valores investida, caso ela já esteja aplicada, a aplicação de valores é mantida.
3. Caso a rede previja que vai haver uma queda, toda a quantia de valores é desaplicada, caso ela já esteja, nada de valores é feito.
4. Caso a rede acerte uma subida, soma-se ao valor total a porcentagem que subiu.
5. Caso a rede acerte uma descida, nada de valores é feito ao valor total.

6. Caso a rendeira esteja em uma descida mas houve uma subida, nada deve ser feito ao valor total.
7. Caso a rendeira esteja em uma subida mas houve uma descida, subtraia-se do valor total a porcentagem da queda.

Essa simulação foi realizada 10 vezes obtendo, na média, um lucro de 21.44% em cima dos 107 dias usados como validação. Nas 10 simulações realizadas, o pior rendimento foi de 12.44% e o melhor, de 31.35%. Nesse mesmo período, a ação PETR4 valorizou 1.85%.

Capítulo 5

Conclusões

“Mathematics has given economics rigor, but alas, also mortis.” - Robert Heilbroner

Este trabalho apresentou a aplicação de duas técnicas de redes neurais artificiais para a previsão de movimentos financeiros. Foi explicado o processo de seleção de dados de entrada no sistema, assim como uma breve contextualização do problema. Apresentou a estrutura básica das redes implementadas, bem como uma breve fundamentação matemática. Foram abordados o desenvolvimento das duas redes neurais artificiais aplicadas ao processo de seleção de parâmetros. Por fim foram apresentados e comparados os resultados das estimativas.

A rede perceptron multicamada, um sistema inteligente capaz de fazer estimativas não lineares foi a primeira rede a ser implementada. Ela foi capaz de encontrar resultados satisfatórios, em desacordo com as expectativas, melhor do que os resultados da CNN implementada. A melhor taxa de acerto obtida foi de 68.22% para dados de validação.

A outra rede desenvolvida foi uma rede neural convolucional profunda, mais complexa e mais pesada computacionalmente que a outra técnica aplicada. Esperava-se que o resultado dessa ANN fosse melhor que o perceptron multicamada. Entretanto o resultado ficou aquém do esperado, acertando apenas 59.81% dos arquivos de validação.

As duas redes implementadas obtiveram resultados acima de um classificador ingênuo, que acertou apenas 50.72% das estimativas durante o período de validação. Foram realizadas simulações de investimentos para o perceptron multicamada. Essas simulações obtiveram um lucro médio, em um período de 107 dias, de 21.44%.

Com trabalhos futuros propõe-se aplicar outras técnicas de sistemas inteligentes para previsão de mercado. Também propõe-se utilizar um banco de dados mais vasto com dados mais diversos, como dados de redes sociais. Outro objetivo futuro é de implementar esse algoritmo para operações de *day-trade* não só operações diárias.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] SUN, J. R. J. S. K. A neural network application: Predicting the stock market. 2014.
- [2] RUSSELL, I. *The Delta Rule*. [S.l.: s.n.], 1996.
- [3] SNOWBERG, E.; WOLFERS, J.; ZITZEWITZ, E. *Prediction markets for economic forecasting*. [S.l.], 2012.
- [4] RUSSELL, I. *Neural Networks Module*. [S.l.: s.n.], 1996.
- [5] DENG, L.; YU, D. Deep learning. *Signal Processing*, Cit ser, v. 7, p. 3–4, 2014.
- [6] CIRESAN, D. et al. Deep, big, simple neural networks for handwritten digit recognition. *Neural computation*, MIT Press, v. 22, n. 12, p. 3207–3220, 2010.
- [7] DONG, G.; FATALIYEV, K.; WANG, L. On-step and multi-step ahead stock prediction using backpropagation neural networks. In: IEEE. *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on*. [S.l.], 2013. p. 1–5.
- [8] PHUA, P. K. H.; MING, D.; LIN, W. Neural network with genetic algorithms for stocks prediction. In: *Fifth Conference of the Association of Asian-Pacific Operations Research Societies, 5th-7th July, Singapore*. [S.l.: s.n.], 2000.
- [9] JUANG, J.; CHEN, C.-Y.; YANG, C.-F. *Proceedings of the 2nd International Conference on Intelligent Technologies and Engineering Systems (ICITES2013)*. [S.l.]: Springer Science & Business, 2014.
- [10] MIZUNO, H. et al. Application of neural network to technical analysis of stock market prediction. *Studies in Informatic and control*, v. 7, n. 3, p. 111–120, 1998.
- [11] KAR, A. Stock prediction using artificial neural networks. *Dept of Computer Science and Engineering, IIT Kanpur*.
- [12] CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, Springer, v. 5, n. 4, p. 455–455, 1992.
- [13] BENGIO, I. G. Y.; COURVILLE, A. Deep learning. Book in preparation for MIT Press. 2016. Disponível em: <<http://www.deeplearningbook.org>>.

- [14] TAIGMAN, Y. et al. DeepFace: Closing the gap to human-level performance in face verification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 1701–1708.
- [15] HAYKIN, S. S. et al. *Neural networks and learning machines*. [S.l.]: Pearson Upper Saddle River, NJ, USA:, 2009.

APÊNDICES

I. TEOREMA DA CONVERGÊNCIA DO PERCEPTRON

A prova para a o algoritmo de convergência do perceptron será apresentada para as seguintes condições:

- Os vetores de entradas são dados por:

$$\mathbf{x}(n) = \begin{bmatrix} b, & x_1(n), & x_2(n), & x_3(n), & \dots, & x_m(n) \end{bmatrix}^T; \quad (\text{I.1})$$

- Os vetores de pesos são:

$$\mathbf{w}(n) = \begin{bmatrix} w_b(n), & w_1(n), & w_2(n), & w_3(n), & \dots, & w_m(n) \end{bmatrix}^T; \quad (\text{I.2})$$

- As classes a serem separadas são \mathcal{C}_1 e \mathcal{C}_2 e as paradas da seguinte forma:

$$\mathbf{w}^T \mathbf{x} > 0 \text{ para cada vetor de entrada } \mathbf{x} \text{ pertencente à classe } \mathcal{C}_1,$$

$$\mathbf{w}^T \mathbf{x} \leq 0 \text{ para cada vetor de entrada } \mathbf{x} \text{ pertencente à classe } \mathcal{C}_2;$$

- Condição inicial de $\mathbf{w}(0) = 0$.

Suponha que $\mathbf{w}(n)^T \mathbf{x}(n) < 0$ para $n = 1, 2, \dots$, que o vetor $\mathbf{x}(n)$ pertence ao subespaço \mathcal{H}_1 . O perceptron classifica incorretamente os vetores $\mathbf{x}_1(1), \mathbf{x}_1(2), \dots$, pois viola a primeira suposição sobre a separação das classes. Então, para a constante $\eta = 1$ podemos usar a Equação (2.9) para atualizar

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mathbf{x}(n) \text{ para } \mathbf{x}(n) \text{ pertencente à classe } \mathcal{C}_1 \quad (\text{I.3})$$

Usando a condição inicial $\mathbf{w}(0) = 0$ e resolvendo a equação para $\mathbf{w}(n+1)$, obtêm-se o seguinte resultado:

$$\mathbf{w}(n+1) = \mathbf{x}(1) + \mathbf{x}(2) + \dots + \mathbf{x}(n) \quad (\text{I.4})$$

Como é assumido que as classes \mathcal{C}_1 e \mathcal{C}_2 são linearmente separáveis, existe uma solução para \mathbf{w} para qual $\mathbf{w}^T \mathbf{x} > 0$ para os vetores $\mathbf{x}(1), \mathbf{x}(2), \dots$, pertencentes ao subespaço. Para uma solução \mathbf{w}_0 podemos definir um número positivo α como

$$\alpha = \min_{\mathbf{x}(n) \in \mathcal{H}_1} \mathbf{w}_0^T \mathbf{x}(n) \quad (\text{I.5})$$

Assim, multiplicando a Equação (I.4) pelo vetor \mathbf{w}_0^T encontramos

$$\mathbf{w}_0^T \mathbf{w}(n+1) = \mathbf{w}_0^T \mathbf{x}(1) + \mathbf{w}_0^T \mathbf{x}(2) + \dots + \mathbf{w}_0^T \mathbf{x}(n) \quad (\text{I.6})$$

Pela definição dada na Equação (I.5), temos

$$\mathbf{w}_0^T \mathbf{w}(n+1) \geq n\alpha \quad (\text{I.7})$$

Aplicando a desigualdade de Cauchy-Schwarz para \mathbf{w}_0 e $\mathbf{w}(n+1)$ chegamos ao seguinte resultado

$$\|\mathbf{w}_0^T\|^2 \|\mathbf{w}(n+1)\|^2 \geq n^2 \alpha^2 \quad (\text{I.8})$$

que é equivalente à

$$\|\mathbf{w}(n+1)\|^2 \geq \frac{n^2 \alpha^2}{\|\mathbf{w}_0\|^2} \quad (\text{I.9})$$

Essa relação agora deve ser guardada. Retornando à Equação (I.3), reescrevemos a seguinte forma

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}(k) \quad \text{para } k = 1, \dots, n, \quad \mathbf{x}(k) \in \mathcal{H}_1 \quad (\text{I.10})$$

Tirando a norma Euclidiana de ambos os lados da equação, obtemos

$$\|\mathbf{w}(k+1)\|^2 = \|\mathbf{w}(k)\|^2 + \|\mathbf{x}(k)\|^2 + 2\mathbf{w}^T(k)\mathbf{x}(k) \quad (\text{I.11})$$

Como $\mathbf{w}^T(k)\mathbf{x}(k) \leq 0$, encontramos, a partir da Equação (I.11), que

$$\|\mathbf{w}(k+1)\|^2 - \|\mathbf{w}(k)\|^2 \leq \|\mathbf{x}(k)\|^2 \quad k = 1, \dots, n \quad (\text{I.12})$$

Definindo

$$\beta = \max_{\mathbf{x}(k) \in \mathcal{H}_1} \|\mathbf{x}(k)\|^2 \quad (\text{I.13})$$

usando as condições iniciais $\mathbf{w}(0) = 0$, encontramos a seguinte inequação

$$\|\mathbf{w}(n+1)\|^2 \leq \sum_{k=1}^n \|\mathbf{x}(k)\|^2 \leq n\beta \quad (\text{I.14})$$

Analisando a Inequação I.14, percebemos que deve haver um número suficiente e grande n para que não haja conflito com a Inequação I.9. Assim, havendo, portanto, um valor máximo para n de forma a satisfazer ambas as inequações. Esse valor é a solução da equação

$$\frac{n_{max}^2 \alpha^2}{\|\mathbf{w}_0\|^2} = n_{max} \beta \quad (\text{I.15})$$

Ess e resultado prova que s $\eta = 1$, $\mathbf{w}(0) = \mathbf{0}$ s xistir uma soluo \mathbf{w}_0 que consiga s parar \mathcal{C}_1 \mathcal{C}_2 , xist um limit n_{max} it rao s para a adaptao dos pesos do perceptron para o sub-espao \mathcal{H}_1 .

A mesma demonstrao utilizada para \mathcal{H}_1 tambm  vlida para \mathcal{H}_2 .

Considerando que $\eta(n)$ o menor inteiro na qual a seguinte desigualdade  verdadeira:

$$\eta(n) \mathbf{x}^T(n) \mathbf{x}(n) > |\mathbf{w}^T(n) \mathbf{x}(n)| \quad (\text{I.16})$$

Ess procedimento encontra que, se o produto interno $\mathbf{w}^T(n) \mathbf{x}(n)$ possui um sinal incorreto, ento $\mathbf{w}^T(n+1) \mathbf{x}(n)$ na iterao $n+1$ ter o sinal correto. Dessa forma, caso $\mathbf{w}^T(n) \mathbf{x}(n)$ tenha um sinal incorreto em n , poder-se- modificar a sequncia de treinamento para que $\mathbf{x}(n+1) = \mathbf{x}(n)$. Cada padro de entrada  apresentado repetidamente at que ele s classificado corretamente.

Uma condio inicial $\mathbf{w}(0)$ diferente do zero apenas implica variao no nmero de iterao necessrias para que haja a convergncia. A variao depend das condies iniciais no nulas se aproximam ou se distanciam da soluo \mathbf{w}_0 .

Para o prximo passo ser utilizada uma simples funo sinal para a ativao do perceptron:

$$\text{sgn}(v) = \begin{cases} +1 & \text{se } v \geq 0 \\ -1 & \text{se } v < 0 \end{cases} \quad (\text{I.17})$$

A resposta do perceptron poder ser escrita como

$$y(n) = \text{sgn}[\mathbf{w}^T(n) \mathbf{x}(n)] \quad (\text{I.18})$$

Para calcular o erro  necessrio um vetor que represente o valor desejado

$$d(n) = \begin{cases} +1 & \text{se } x(n) \text{ pertence a class } \mathcal{C}_1 \\ -1 & \text{se } x(n) \text{ pertence a class } \mathcal{C}_2 \end{cases} \quad (\text{I.19})$$

O erro do sistema  portanto

$$\text{erro} = d(n) - y(n) \quad (\text{I.20})$$

Por fim, a adaptao dos pesos $\mathbf{w}(n)$ poder ser concatenada na forma da seguinte regra de atualizao para correo do erro[15]

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)] \mathbf{x}(n) \quad (\text{I.21})$$

II. BACK-PROPAGATION

Considere a Figura II.1, que representa um neurônio j sendo alimentado por um conjunto de sinais oriundos de uma camada de neurônios à sua esquerda.

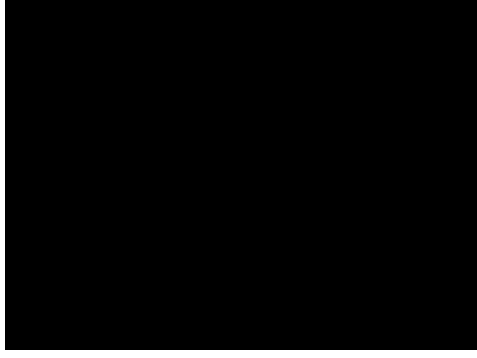


Figura II.1: Gráfico que representa a propagação do sinal em um neurônio j

O campo local produzido na entrada da função de ativação do neurônio j é

$$s_j(n) = \sum_{i=0}^m w_{ji}(n) y_i(n) \quad (\text{II.1})$$

onde m é o número total de entradas (exceto o *bias*) aplicado ao neurônio m .

O algoritmo de *back-propagation* aplica uma correção $\Delta w_{ji}(n)$ ao ajuste $w_{ji}(n)$, proporcional à derivada parcial $\partial \mathcal{E} / \partial w_{ji}(n)$. De acordo com a regra da cadeia, podemos expressar o gradiente como

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial s_j(n)} \frac{\partial s_j(n)}{\partial w_{ji}(n)} \quad (\text{II.2})$$

A derivada parcial $\partial \mathcal{E} / \partial w_{ji}(n)$ representa a sensibilidade do sistema de treinamento à direção da busca no espaço de pesos para o peso sináptico w_{ji} .

Somando todos os erros na camada de saída da rede, temos o erro total instantâneo do sistema.

$$\mathcal{E}(n) = \sum_{j \in C} \mathcal{E}_j(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \quad (\text{II.3})$$

Em que C inclui todos os neurônios da camada de saída.

Diferenciando os dois lados da equação II.3 em relação a $e_j(n)$ encontramos

$$\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} = e_j(n) \quad (\text{II.4})$$

Para $y_j(n) = o_j(n)$, diferenciando os dois lados da equação 2.14 encontra-se

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = -1 \quad (\text{II.5})$$

Em seguida, diferenciando a equação 2.11 em relação à $s_j(n)$, temos

$$\frac{\partial y_j(n)}{\partial s_j(n)} = \sigma'_j(s_j(n)) \quad (\text{II.6})$$

Por fim, diferenciando a equação II.1 em relação à $w_{ji}(n)$ chegamos

$$\frac{\partial s_j(n)}{\partial w_{ji}(n)} = y_i(n) \quad (\text{II.7})$$

Juntando as equações II.7 e II.4 na equação II.2, encontra-se a expressão

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n) \sigma'_j(s_j(n)) y_i(n) \quad (\text{II.8})$$

O ajuste $\Delta w_{ji}(n)$ aplicado em $w_{ji}(n)$ é definida por

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} \quad (\text{II.9})$$

com η sendo o parâmetro de aprendizagem no algoritmo de *back-propagation*. O sinal negativo na equação II.9 representa a queda no gradiente, significando uma mudança de pesos em direção ao menor valor de $\mathcal{E}(n)$.

Substituindo a equação II.8 na equação II.9:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad (\text{II.10})$$

onde o gradiente local $\delta_j(n)$ é definida por

$$\begin{aligned} \delta_j(n) &= \frac{\partial \mathcal{E}(n)}{\partial s_j(n)} \\ &= \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial s_j(n)} \\ &= e_j(n) \sigma'_j(s_j(n)) \end{aligned} \quad (\text{II.11})$$

Agora encontrar o gradiente local da última camada é trivial, sabendo o erro $e_j(n)$ basta aplicar a equação II.11.

Para determinar o gradiente das camadas escondidas, a tarefa é um pouco mais complexa. Um neurônio não pertencente à camada de saída não tem uma saída específica. Portanto, o sinal de erro de um neurônio escondido é determinado propagando-se para trás o sinal de erro de todos os neurônios dos quais ele está conectado. Esse processo é recursivo, logo, um neurônio da

camada $[l]$ vai ter como m o sinal de erro, influência de todos os erros da camada $[l+1]$, o sinal de erro de cada neurônio dessa camada terá, por sua vez, influência por parte de cada neurônio da camada $[l+2]$ assim sucessivamente.

Para poder definir como essa propagação ocorre, primeiro será utilizada a seguinte definição do gradiente $\delta_j(n)$ para um neurônio escondido j

$$\begin{aligned}\delta_j(n) &= -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial s_j(n)} \\ &= -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \sigma'_j(s_j(n))\end{aligned}\tag{II.12}$$

Utilizando a equação II.3 para um neurônio escondido j seguido de um outro neurônio, da camada de saída, k , de tal forma que j está presente na camada l e k está presente na camada L , temos

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)\tag{II.13}$$

Definindo a equação II.13 em relação ao sinal $y_j(n)$, aplicando a regra da cadeia, resulta-se em

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial y_i(n)} &= \sum_k e_k \frac{\partial e_k(n)}{\partial y_i(n)} \\ &= \sum_k e_k \frac{\partial e_k(n)}{\partial s_i(n)} \frac{\partial s_k(n)}{\partial y_i(n)}\end{aligned}\tag{II.14}$$

Para o neurônio de saída k

$$\begin{aligned}e_k(n) &= d_k - y_k(n) \\ &= d_k(n) - \sigma_k(s_k(n))\end{aligned}\tag{II.15}$$

Assim,

$$\frac{\partial e_k(n)}{\partial s_k(n)} = -\sigma'_k(s_k(n))\tag{II.16}$$

Também é notável que, para qualquer neurônio k , o campo local é

$$s_k(n) = \sum_{j=0}^m w_{kj}(n) y_j(n)\tag{II.17}$$

em que m é o total de entradas, excluindo o bias, aplicado ao neurônio k . Definindo a equação II.17 em relação a $y_j(n)$, temos

$$\frac{\partial s_k(n)}{\partial y_k(n)} = w_{kj}(n) \quad (\text{II.18})$$

Substituindo as equações II.16 e II.18 na equação II.14 temos

$$\frac{\partial \mathcal{E}}{\partial y_i(n)} = - \sum_k e_k(n) \sigma'_k(s_k(n)) w_{kj}(n) \quad (\text{II.19})$$

que pode ser simplificada como

$$\frac{\partial \mathcal{E}}{\partial y_i(n)} = - \sum_k \delta_k(n) w_{kj}(n) \quad (\text{II.20})$$

A Equação (II.20) é a definição do gradiente local $\delta_k(n)$, similar à Equação (II.11), mas com o índice j substituído por k . Finalmente, utilizando a Equação (II.20) na Equação (II.12) encontramos a fórmula da *back-propagation*.

$$\delta_j(n) = \sigma'_j(s_j(n)) \sum_k \delta_k(n) w_{kj}(n), \quad \text{para um neurônio } j \text{ escondido} \quad (\text{II.21})$$

A generalização do algoritmo para a correção $\Delta w_{ji}(n)$ aplicada ao peso sináptico conectando o neurônio i ao neurônio j é

$$\begin{pmatrix} \text{correção} \\ \text{de pesos} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{parâmetro de} \\ \text{aprendizagem} \\ \eta \end{pmatrix} \times \begin{pmatrix} \text{gradiente} \\ \text{local} \\ \delta_j(n) \end{pmatrix} \times \begin{pmatrix} \text{sinal de entrada} \\ \text{do neurônio } j, \\ y_i(n) \end{pmatrix} \quad (\text{II.22})$$

na qual $\delta_j(n)$ depende se o neurônio j é um neurônio de saída ou é um neurônio escondido. Portanto, a forma geral de $\delta_j(n)$ é [15]

$$\delta_j(n) = \begin{cases} e_j(n) \sigma'_j(s_j(n)), & \text{se o neurônio } j \text{ for um neurônio da camada de saída,} \\ \sigma'_j(s_j(n)) \sum_k \delta_k(n) w_{kj}(n), & \text{se o neurônio } j \text{ for um neurônio da camada escondida.} \end{cases} \quad (\text{II.23})$$

III. CORRELAÇÃO CRUZADA ENTRE COTAÇÃO E POPULARIDADE NO GOOGLE TRENDS

Essa seção contém, listadas, os gráficos das correlações cruzadas entre as cotações do ativo PETR4 da Petrobras e o número de pesquisas dos diferentes temas utilizados no trabalho.

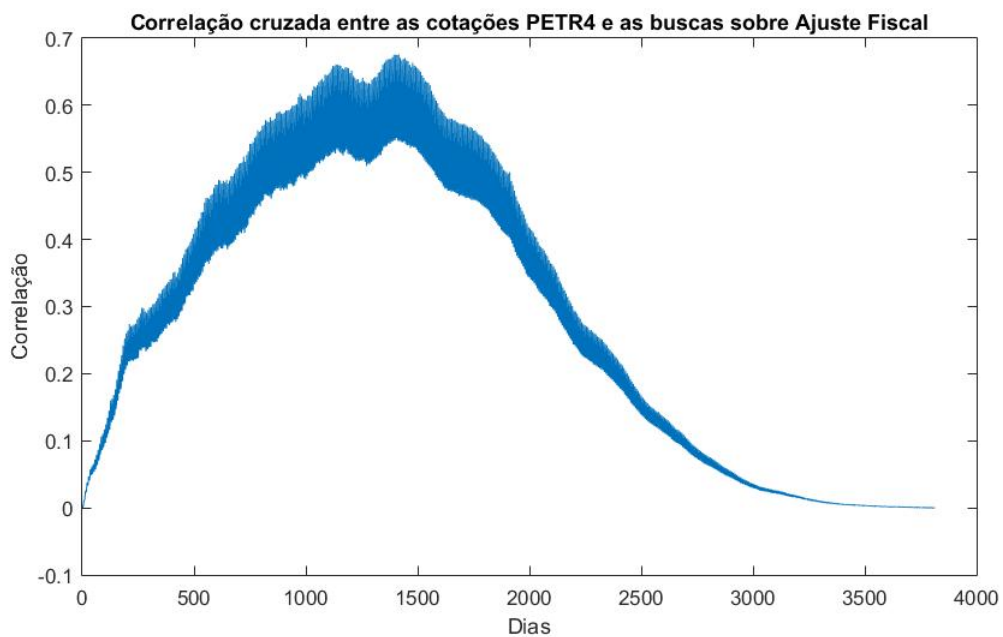


Figura III.1: Correlação cruzada entre cotações "Ajuste Fiscal"

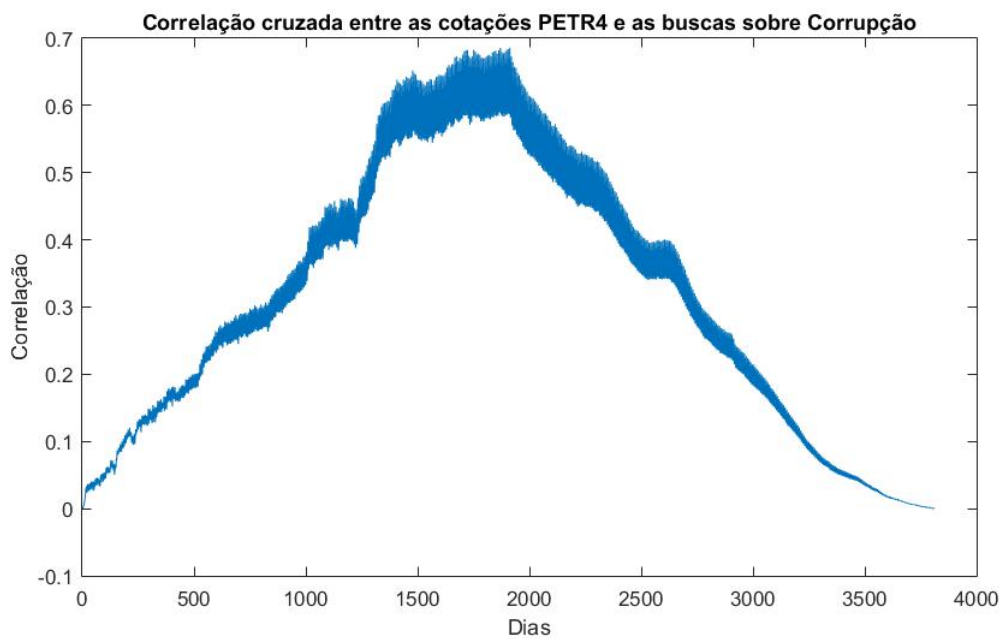


Figura III.2: Correlação cruzada entre cotações "Corrupção"

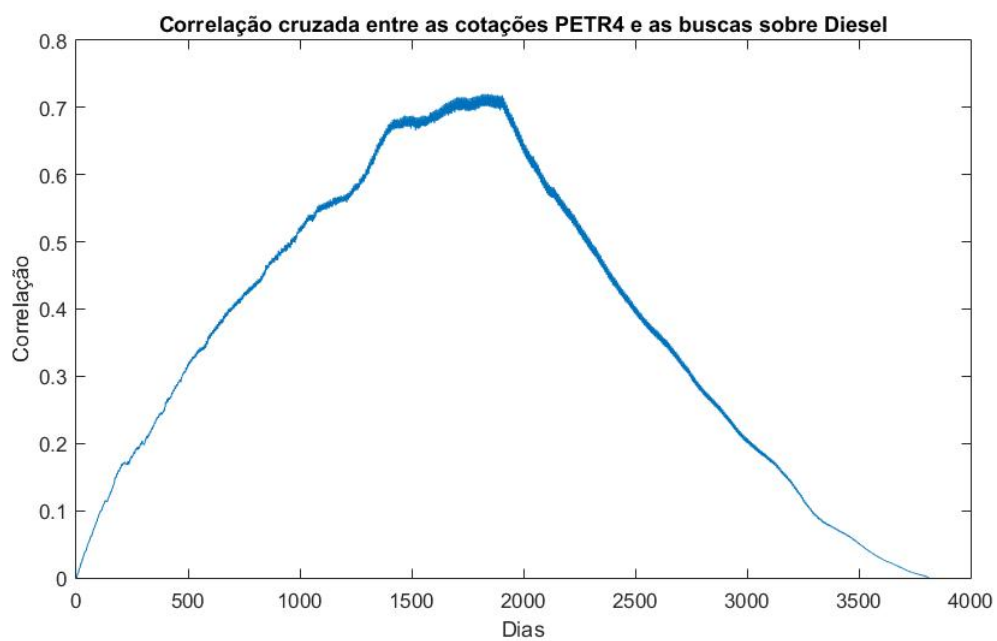


Figura III.3: Correlação cruzada entre cotações "Diesel"

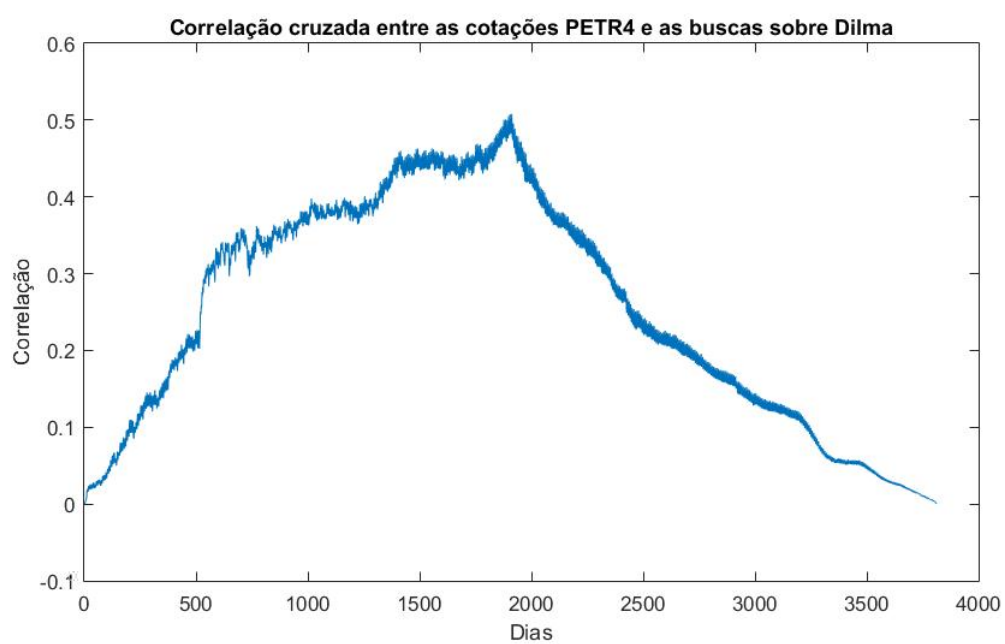


Figura III.4: Correlação cruzada entre cotações "Dilma"

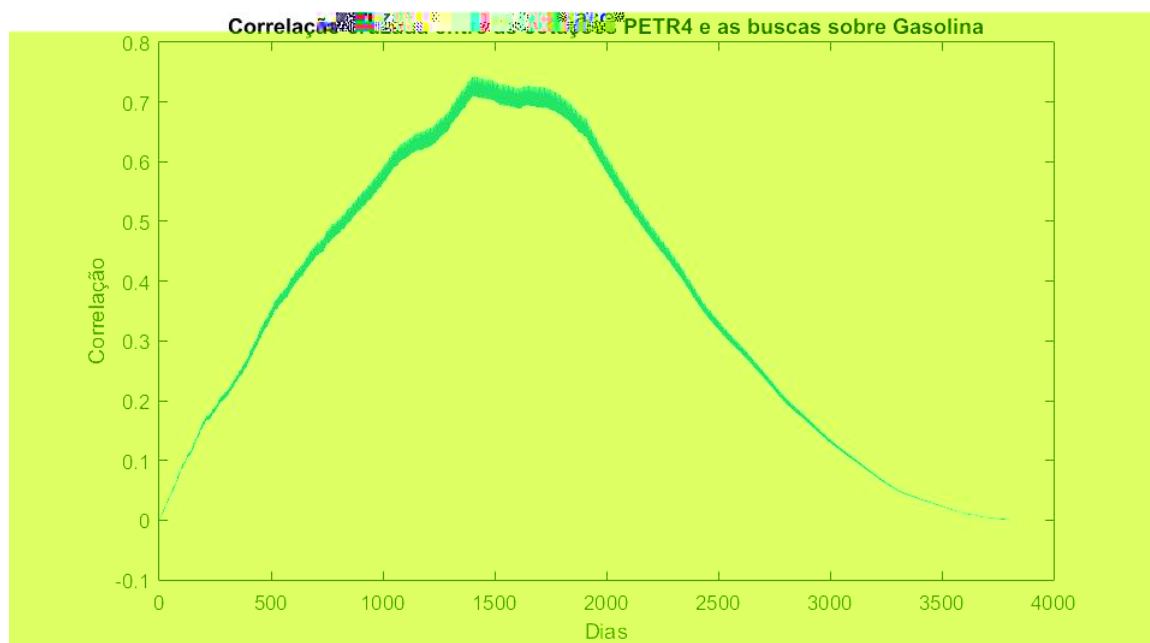


Figura III.5: Correlação cruzada entre cotações "Gasolina"

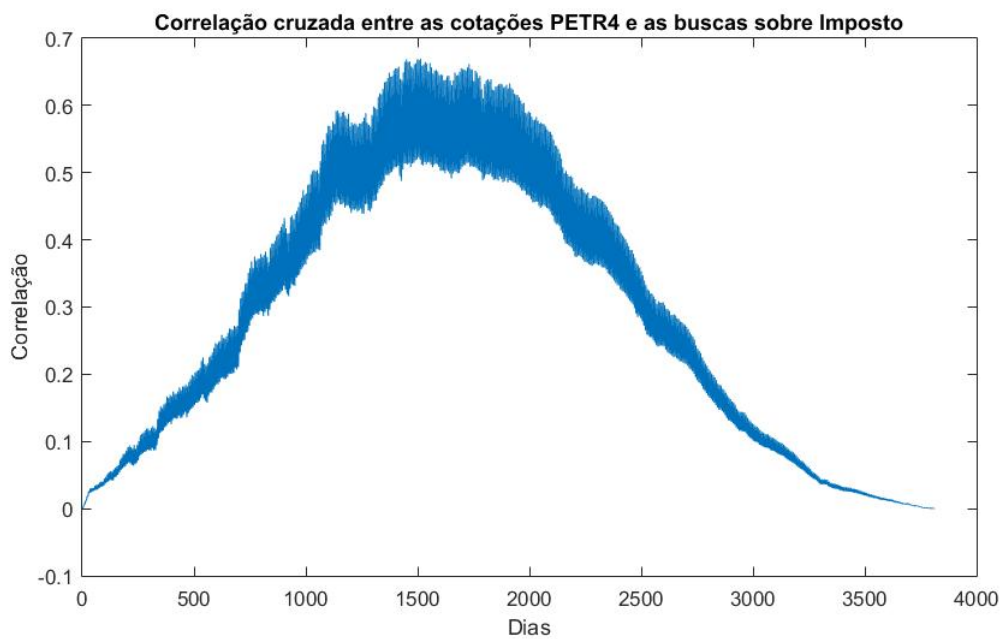


Figura III.6: Correlação cruzada entre cotações "Imposto"

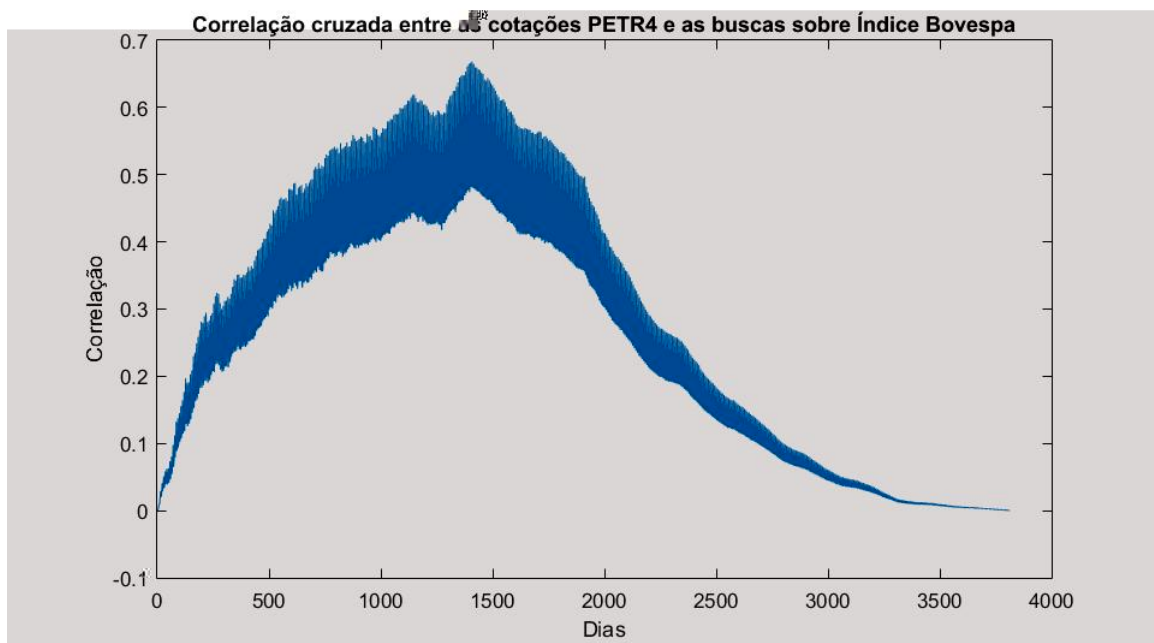


Figura III.7: Correlação cruzada entre cotações "Índice Bovespa"

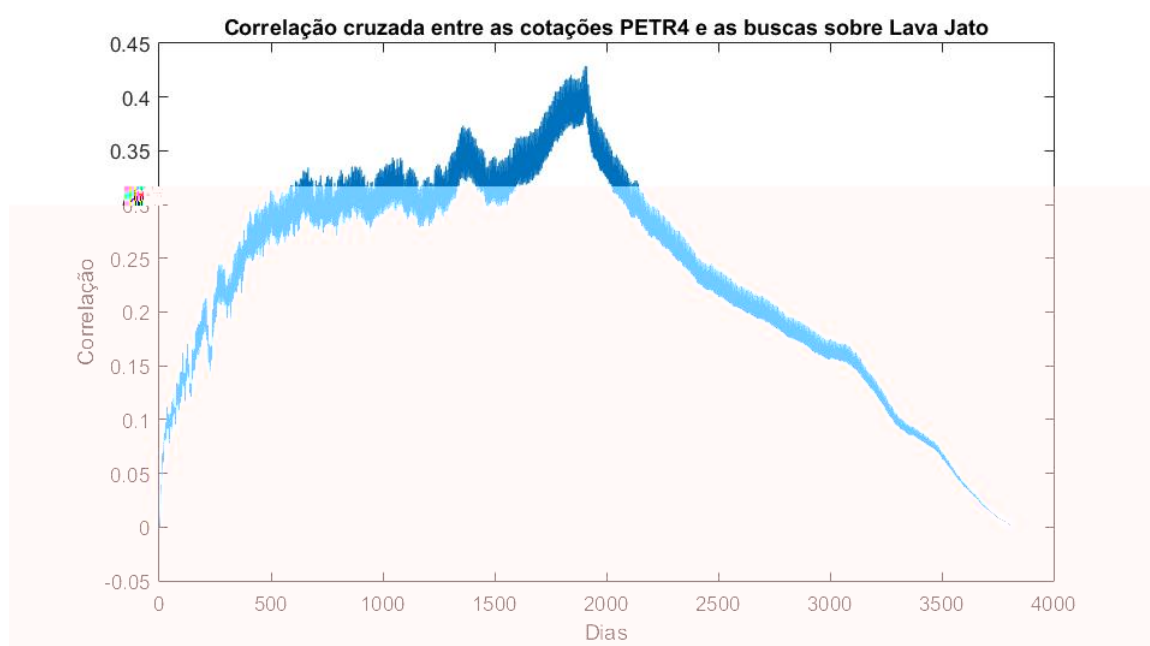


Figura III.8: Correlação cruzada entre cotações "Lava Jato"

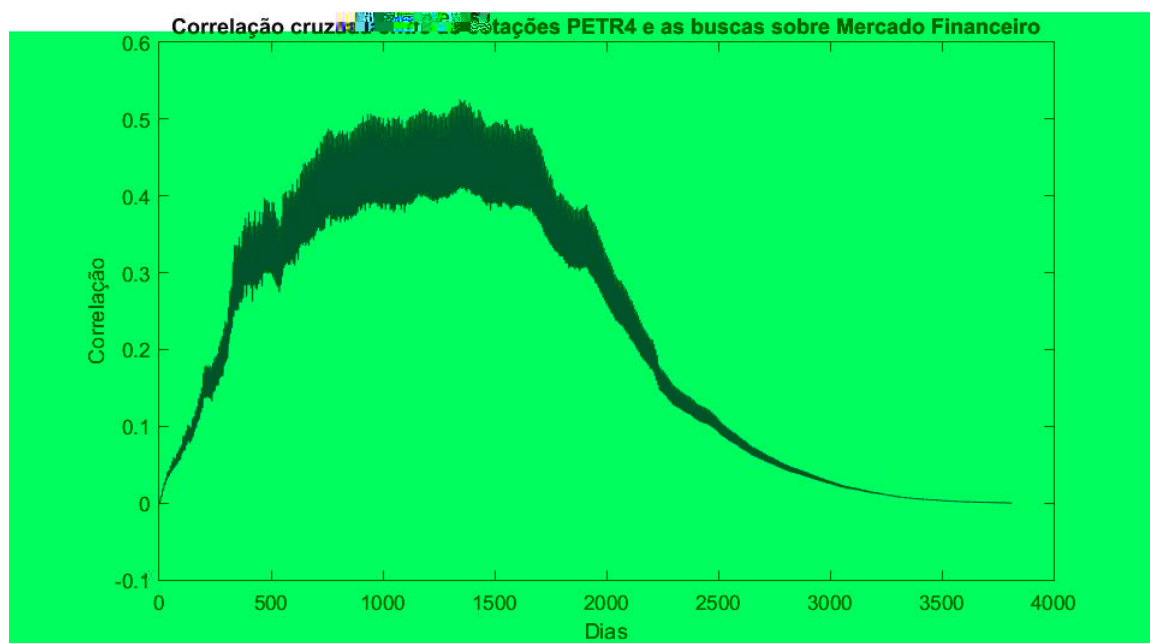


Figura III.9: Correlação cruzada entre cotações "Mercado Financeiro"

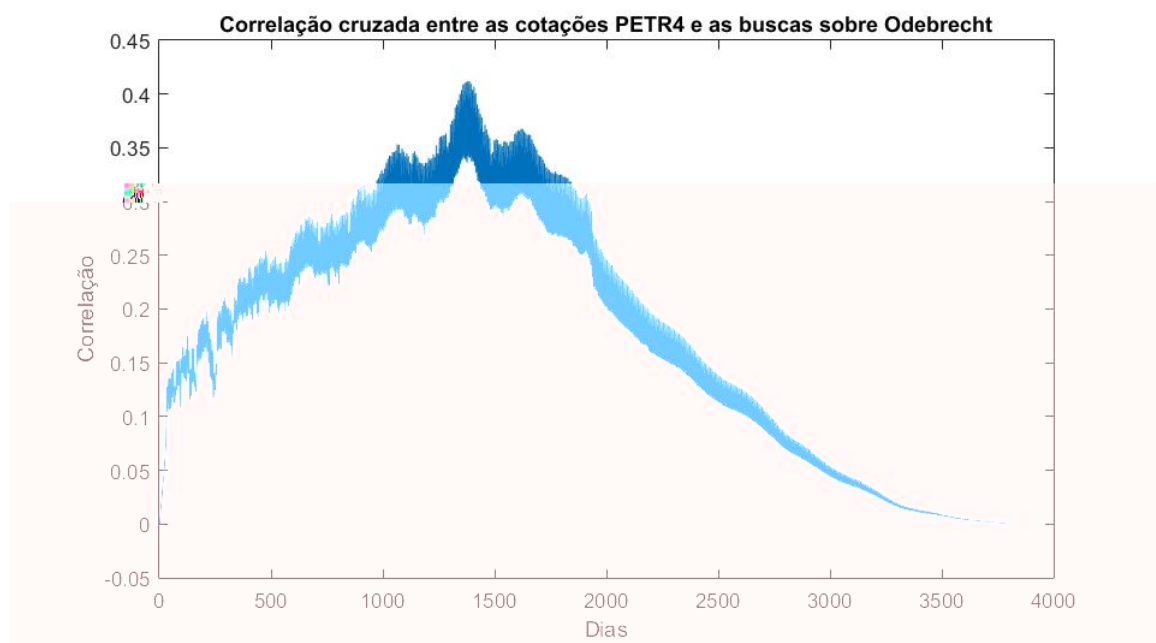


Figura III.10: Correlação cruzada entre cotações "Odebrecht"

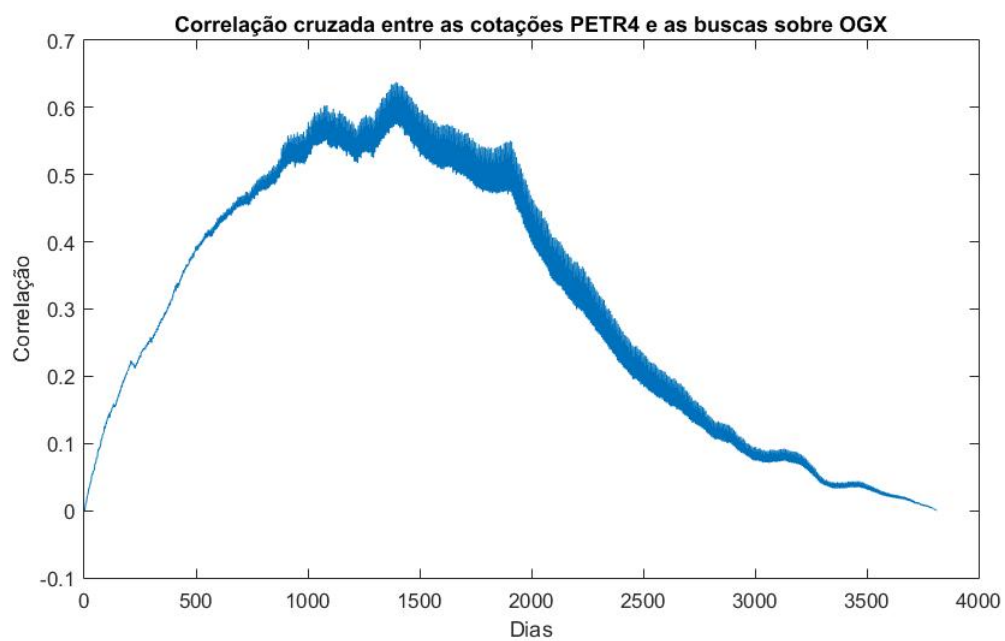


Figura III.11: Correlação cruzada entre cotações "OGX"

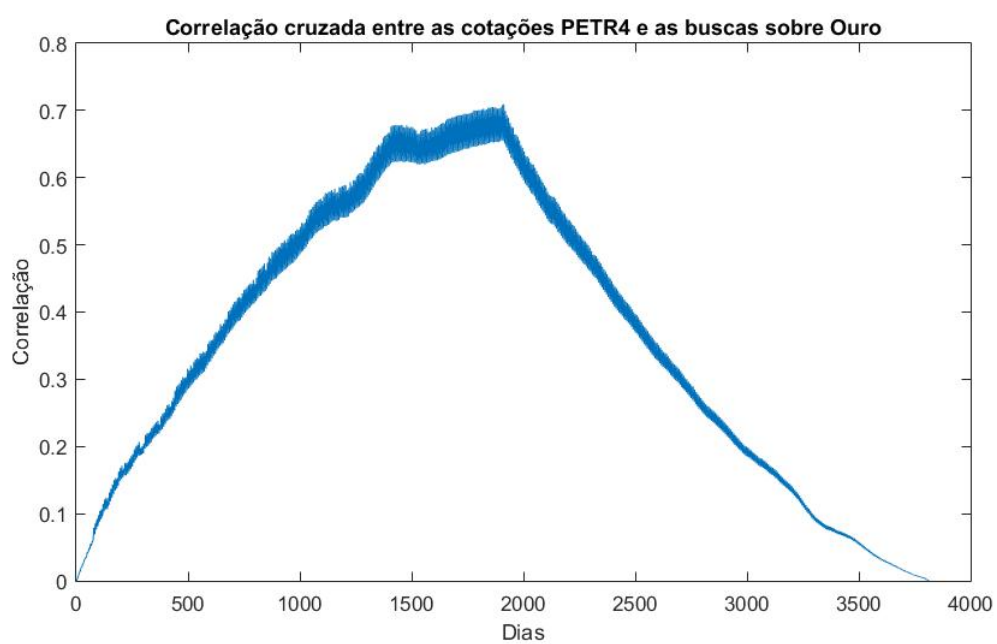


Figura III.12: Correlação cruzada entre cotações "Ouro"

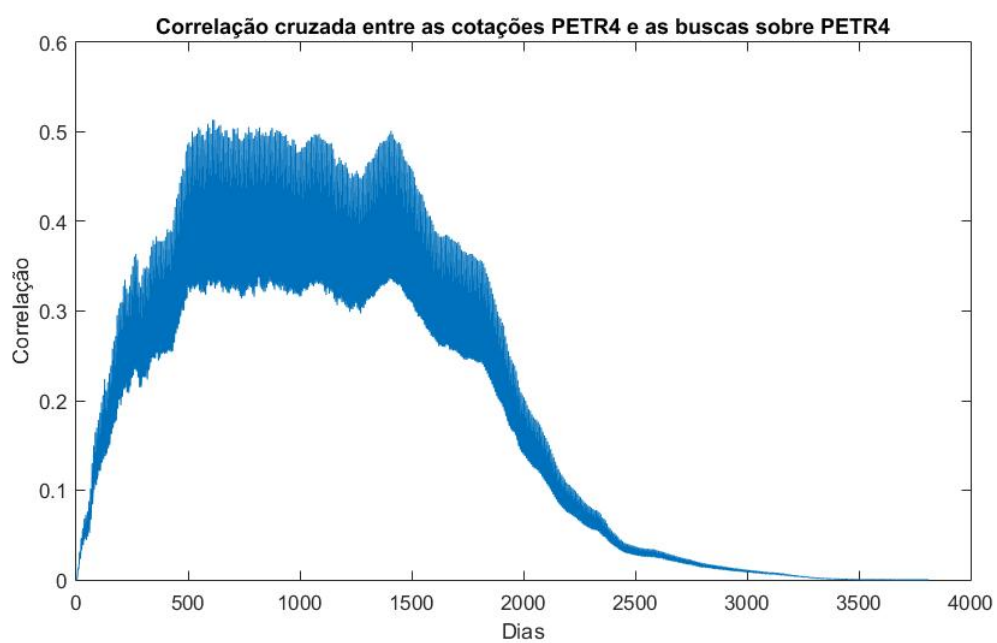


Figura III.13: Correlação cruzada entre cotações "PETR4"

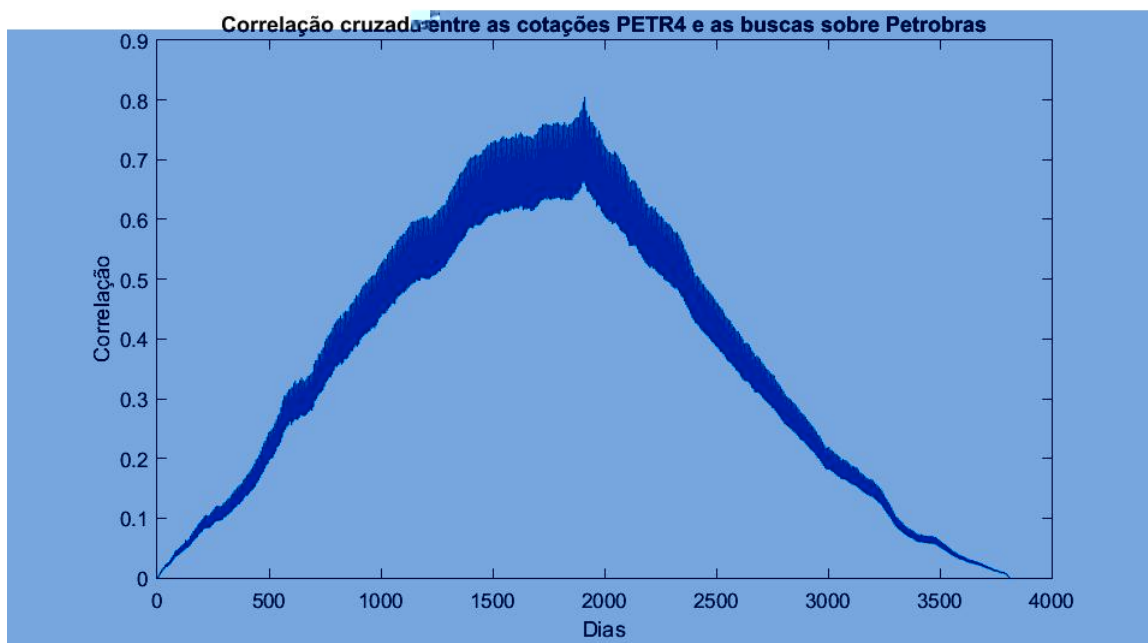


Figura III.14: Correlação cruzada entre cotações "Petrobras"

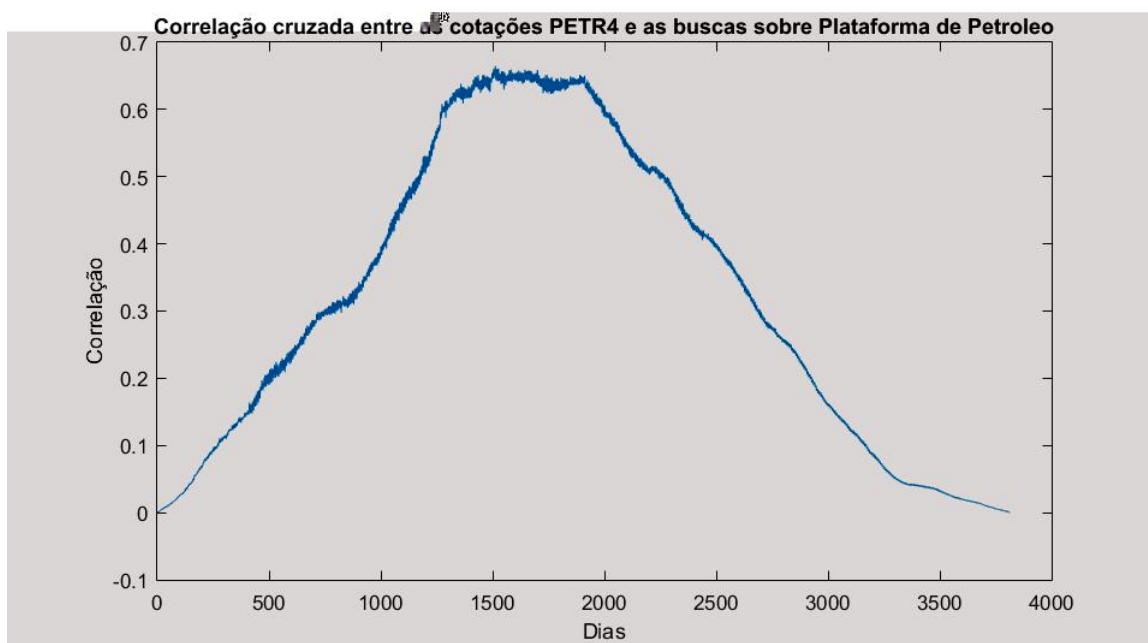


Figura III.15: Correlação cruzada entre cotações "Plataforma de Petróleo"

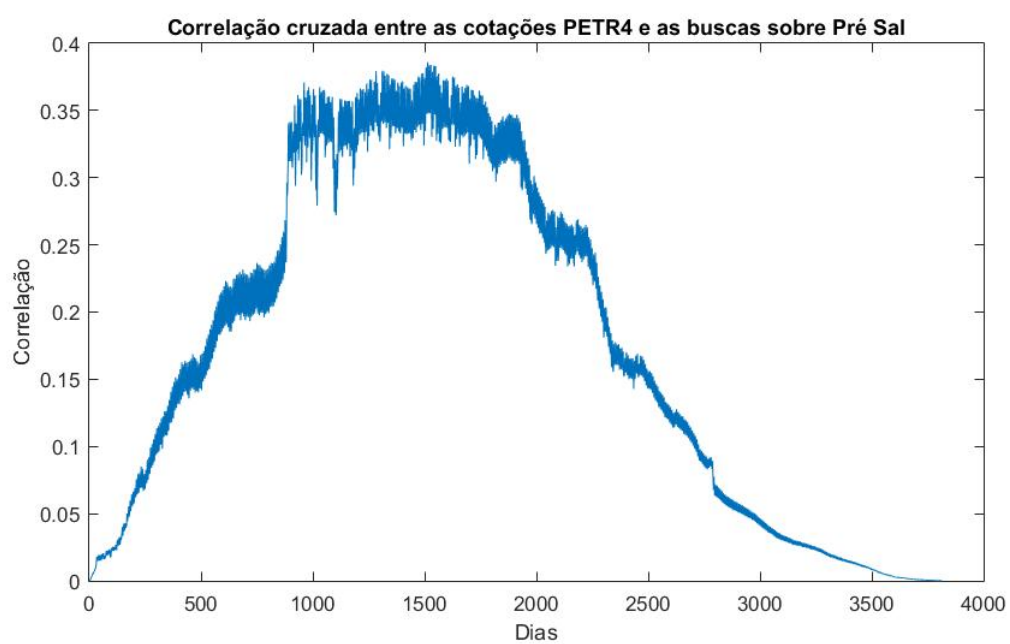


Figura III.16: Correlação cruzada entre cotações "Pré Sal"

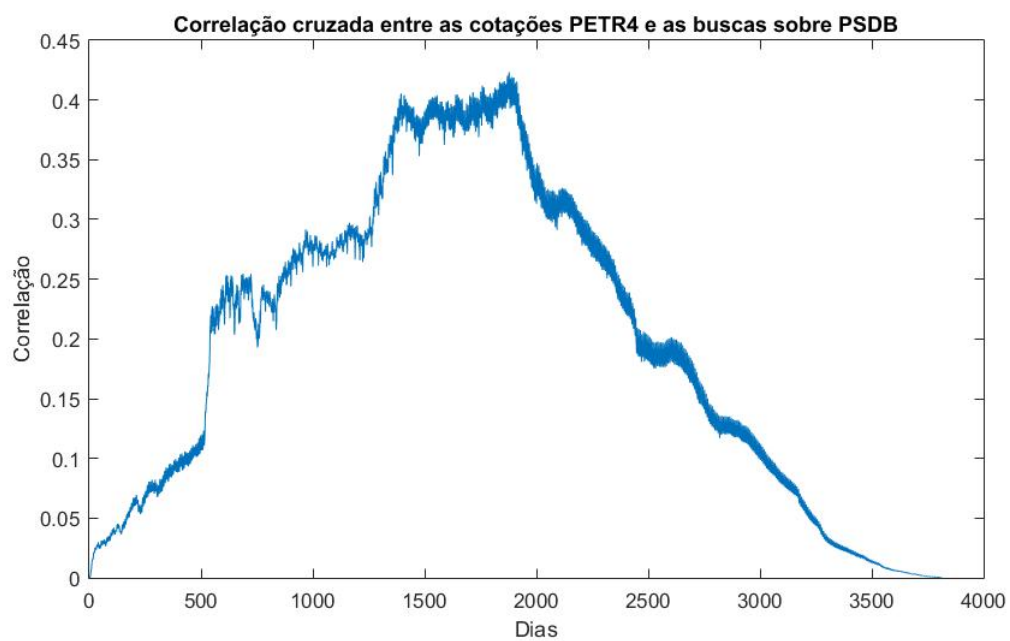


Figura III.17: Correlação cruzada entre cotações "PSDB"

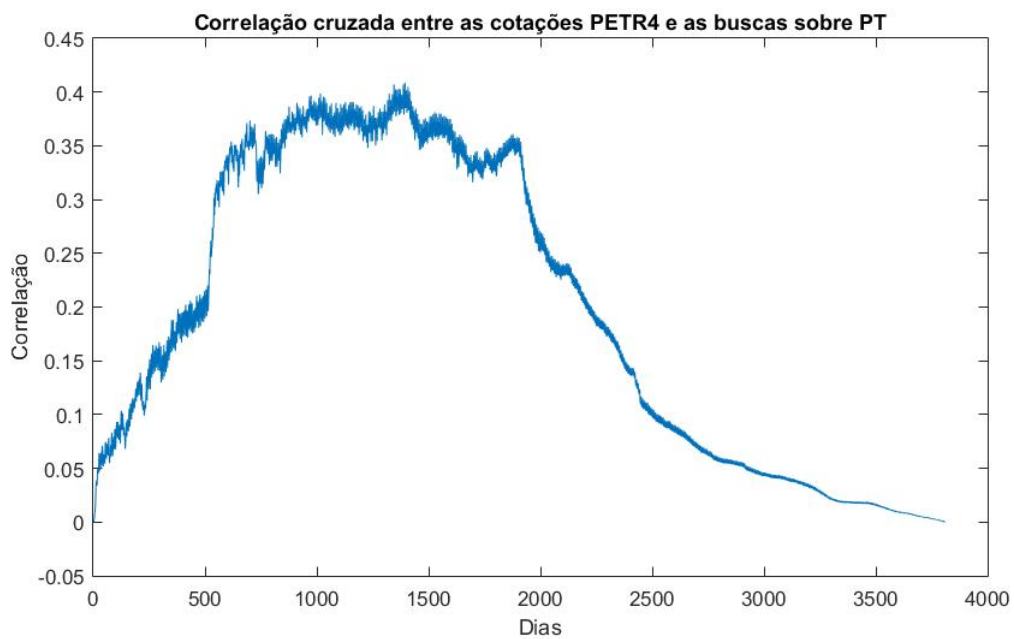


Figura III.18: Correlação cruzada entre cotações "PT"

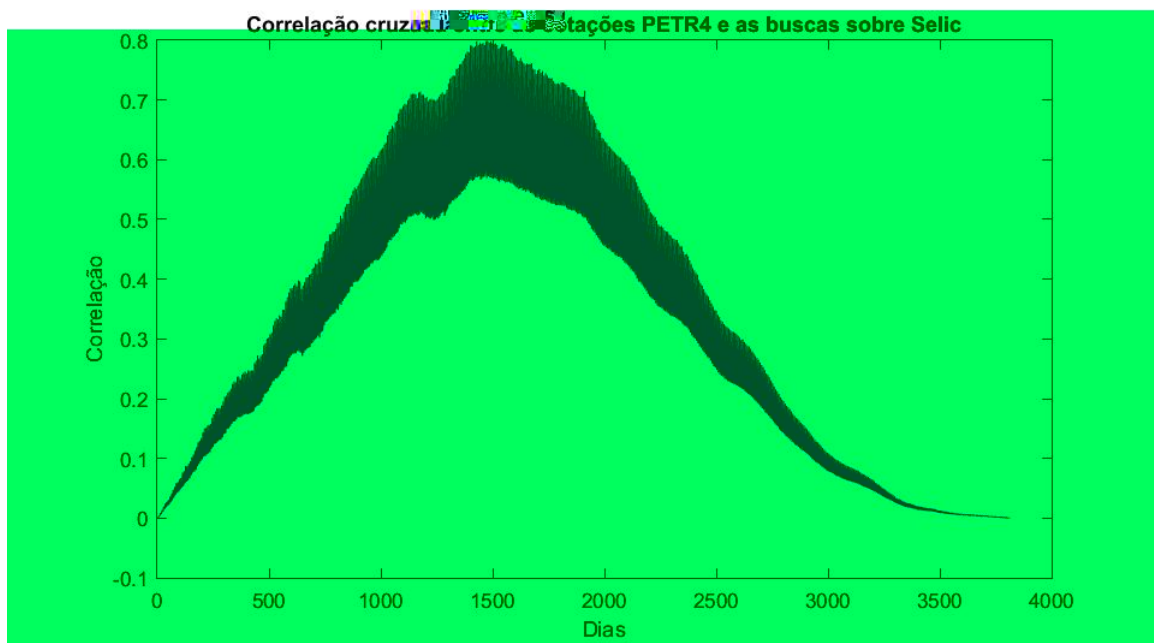


Figura III.19: Correlação cruzada entre cotações "Selic"

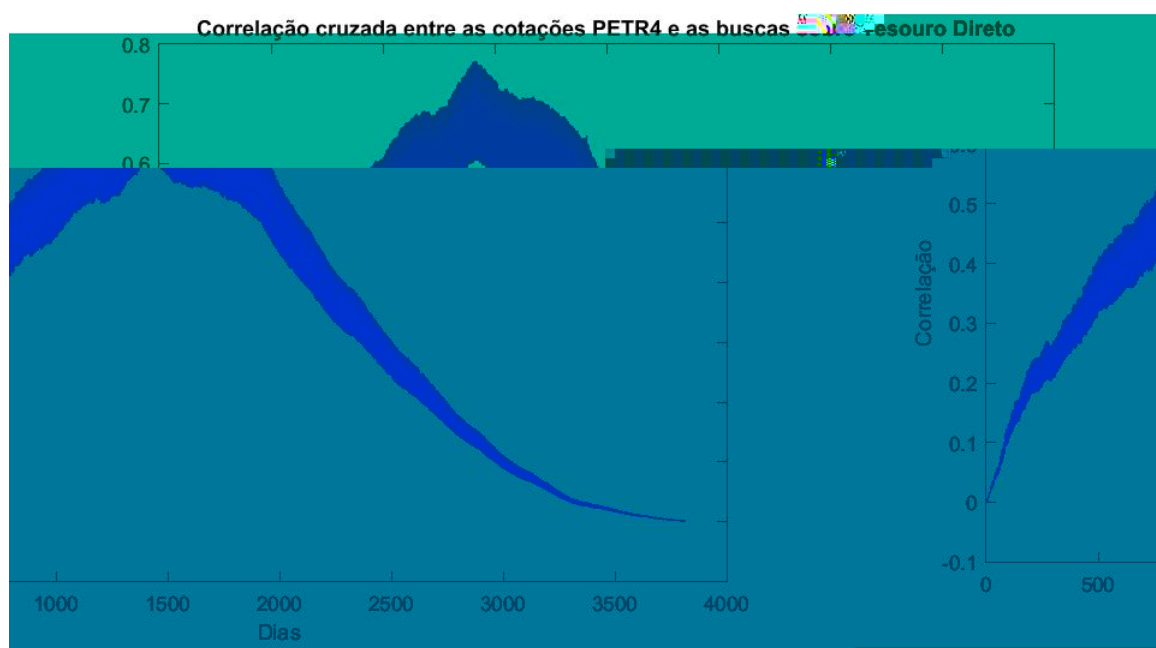


Figura III.20: Correlação cruzada entre as cotações PETR4 e as buscas de Tesouro Direto