Lucas Pimenta Vasco

Um Estudo de Redes Neurais Recorrentes no Contexto de Previsões no Mercado Financeiro

São Carlos, Brasil

2020

Lucas Pimenta Vasco

Um Estudo de Redes Neurais Recorrentes no Contexto de Previsões no Mercado Financeiro

Trabalho de conclusão do curso de graduação em engenharia de computação submetido ao corpo docente do Departamento de Computação da Universidade Federal de São Carlos como parte dos requisitos necessários para obtenção do título de engenheiro de computação.

Universidade Federal de São Carlos – UFSCar

Departamento de Computação

Engenharia de Computação

Orientador: Profa. Dra. Heloisa de Arruda Camargo

São Carlos, Brasil

2020

Lucas Pimenta Vasco
Um Estudo de Redes Neurais Recorrentes no Contexto de Previsões no Mercado Financeiro/ Lucas Pimenta Vasco. – São Carlos, Brasil, 2020-
48 p. : il. (algumas color.) ; 30 cm.

Orientador: Profa. Dra. Heloisa de Arruda Camargo

Trabalho de Conclusão de Curso – Universidade Federal de São Carlos – UFSCar
Departamento de Computação
Engenharia de Computação, 2020.

Lucas Pimenta Vasco

Um Estudo de Redes Neurais Recorrentes no Contexto de Previsões no Mercado Financeiro

Trabalho de conclusão do curso de graduação em engenharia de computação submetido ao corpo docente do Departamento de Computação da Universidade Federal de São Carlos como parte dos requisitos necessários para obtenção do título de engenheiro de computação.

São Carlos, Brasil, 01 de Dezembro de 2020:

Profa. Dra. Heloisa de Arruda Camargo
Orientador

Prof. Dr. Diego Furtado Silva
Membro da banca

Profa. Dra. Tatiane Nogueira Rios
Membro da banca

São Carlos, Brasil
2020

Dedico este trabalho aos meus pais Marco e Nanci, por todo carinho e apoio que me fizeram alcançar essa conquista, e aos meus avós, por todo amor que sempre recebi.

Agradecimentos

Agradeço primeiramente aos meus pais, Marco e Nanci, por todo amor e apoio. Graças ao seu incentivo e suporte, sempre guiando no caminho correto, tive a oportunidade de realizar esse sonho que era me tornar engenheiro.

Aos meus avós Iredil, Octávio, Dolca, Vicente, e Nair, avó de coração, que cuidam de mim desde criança e contribuíram para todas as minhas conquistas até aqui.

Aos meus irmãos Bruno e Milena, por toda parceria e amizade desde que entraram em minha vida muitos anos atrás.

À minha namorada Amanda, que foi a melhor companhia nos últimos anos desde que a conheci, ajudou a superar as adversidades e foi essencial para que eu conseguisse realizar esse trabalho.

À todos meus amigos de longa data, em especial ao Bruno e ao Pedro, companheiros para todos os momentos, não importa a distância, antes mesmo de começar a faculdade.

Aos amigos que fiz na faculdade, especialmente aos companheiros de curso Murilo, Tomas, Hugo Felipe e Alexandre, que foram parceiros em momentos de estudo e de descontração.

Aos amigos que fiz na Alemanha, que fizeram do tempo que passamos juntos inesquecível, e se tornaram uma segunda família para mim.

Aos amigos do meu estágio, que espero ter ensinado a eles tanto quanto eles ensinaram a mim, e que fizeram do estágio muito mais do que só um trabalho.

À república Voodoo e todos que lá conheci, que me receberam de portas abertas e me acolheram como família.

À minha orientadora, Heloísa de Arruda Camargo, por me auxiliar e guiar no desenvolvimento deste estudo.

Resumo

Predição de séries temporais financeiras é uma das aplicações de inteligência artificial mais estudadas por pesquisadores do mercado financeiro, tanto no mundo acadêmico quanto no corporativo. Dentro dessa área, há um grande destaque para a predição de valores de ações, principalmente devido à possibilidade de rentabilidade. Uma das técnicas mais utilizadas para fazer esse tipo de predição é a aplicação de redes neurais artificiais.

Esse trabalho é focado em uma das classes de redes neurais artificiais, as redes neurais recorrentes. São utilizados três tipos de arquitetura dessa área: o modelo de redes neurais recorrentes simples (VRNN, do inglês *Vanilla Recurrent Neural Network*), o *Long Short-Term Memory* (LSTM) e o *Gated Recurrent Unit* (GRU). Ao longo do trabalho são criados modelos de cada uma dessas arquiteturas para três ações diferentes da bolsa de valores brasileira, mais especificamente das ações do Grupo Fleury (FLRY3), da Petrobras (PETR4) e da Vale (VALE3). Os modelos criados para cada uma das ações fazem uma predição dos valores dessas ações, e os resultados são comparados entre si com a intenção de avaliar qual predição obtém os valores mais próximos dos valores reais. As medidas de erro utilizadas nesse trabalho são o erro absoluto médio e o erro quadrático médio. Ao final, concluiu-se que os modelos GRU e LSTM obtém resultados semelhantes, com um desempenho um pouco melhor do GRU, e o VRNN conseguiu detectar padrões nos dados, mas não com tanta precisão quantos os outros.

Palavras-chave: Predição de ações. Redes neurais artificiais. Redes neurais recorrentes.

Abstract

Financial time series forecasting is one of the most researched artificial intelligence applications by financial market analysts, both in the academic and corporate world. Within this area, there is a great emphasis on the prediction of stock values, mainly due to the possibility of profitability. One of the most used techniques to make this type of prediction is the application of artificial neural networks.

This work is focused on one of the classes of artificial neural networks, the recurrent neural networks. Three types of architecture in this area are used: the Vanilla Recurrent Neural Network (VRNN) model, the Long Short-Term Memory (LSTM) model and the Gated Recurrent Unit (GRU) model. Throughout this paper, models of each of these architectures are created for three different stocks of the Brazilian stock exchange, namely the stocks from Grupo Fleury (FLRY3), Petrobras (PETR4) and Vale (VALE3). The models created for each of the stocks make a prediction of the value of those stocks, and the results are compared to each other with the intention of evaluating which model predicts the values closest to the real values. The accuracy measures used in this work are the mean absolute error and the mean square error. In the end, it was concluded that the GRU and LSTM models obtain similar results, with a slightly better performance from GRU, and VRNN was able to detect patterns in the data, but not as accurately as the other two.

Key-words: Stock prediction. Artificial neural network. Recurrent neural network.

Lista de ilustrações

Figura 1 – Ilustração de um diagrama de Venn mostrando relação entre Inteligência Artificial, Aprendizado de Máquina e *Deep Learning*. Fonte: (Exastax, 2017), adaptada pelo autor. 19
Figura 2 – Representação de um neurônio artificial. Fonte: elaborada pelo autor. . 20
Figura 3 – Funções de ativação. Fonte: elaborada pelo autor. 21
Figura 4 – Sentido de propagação. Fonte: (ACADEMY, 2019). 23
Figura 5 – Rede recorrente. Fonte: (ACADEMY, 2019). 23
Figura 6 – Rede Neural Recorrente. Fonte: Hiransha et al. (2018). 24
Figura 7 – Célula LSTM. Fonte: Wikipedia contributors (2020). 25
Figura 8 – Célula GRU. Fonte: Wikipedia contributors (2020). 26

Figura 9 – Valores mínimos de erro obtidos para diferentes números de neurônios. 35
Figura 10 – Valores mínimos de erro obtidos para diferentes porcentagens de *dropout*. 35
Figura 11 – Valores mínimos de erro obtidos para diferentes otimizadores. 36
Figura 12 – Valores mínimos de erro obtidos para diferentes tamanhos de lote. . . . 36
Figura 13 – Valores mínimos de erro obtidos para diferentes funções de ativação. . . 37
Figura 14 – Valores mínimos de erro obtidos variando o número de épocas. 38
Figura 15 – Valores mínimos de erro obtidos variando o número de épocas para os dados de teste. 38
Figura 16 – Valores mínimos de erro obtidos variando o tamanho da janela de aprendizado. 39
Figura 17 – Valores mínimos de erro obtidos variando o tamanho da janela de aprendizado para os dados de teste. 39

Figura 18 – 41
Figura 19 – Valores reais e preditos da ação FLRY3 utilizando GRU 41
Figura 20 – 41
Figura 21 – Valores reais e preditos da ação PETR4 utilizando GRU 41
Figura 22 – 42
Figura 23 – Valores reais e preditos da ação VALE3 utilizando GRU 42

Lista de tabelas

Tabela 1 – Exemplo do conjunto de dados (PETR4) extraído do site Yahoo!... (2020), mostrando as 5 primeiras e 5 últimas linhas 31
Tabela 2 – Base de dados após normalização 33

Tabela 3 – Valores mínimos de EQM obtidos durante os testes para ação e tipo de rede neural 40
Tabela 4 – Valores mínimos de EAM obtidos durante os testes para ação e tipo de rede neural 40

Lista de abreviaturas e siglas

ML	Machine Learning.
RNR	Redes Neurais Recorrentes.
DL	Deep Learning.
RNA	Rede Neural Artificial.
LSTM	Long Short-Term Memory.
GRU	Gated Recurrent Unit.
VRNN	Vanilla Recurrent Neural Network.
MLP	Multi Layer Perceptron.
CNN	Convolutional Neural Network.
SVM	Support Vector Machine.
ARIMA	Autoregressive Integrated Moving Average.
IA	Inteligência Artificial.
EQM	Erro Quadrático Médio.
EAM	Erro Absoluto Médio.
CSV	Comma Separated Value.
API	Application Programming Interface.
MM	Média Móvel.

Sumário

INTRODUÇÃO **13**
Motivação **14**
Objetivos **14**

1 FUNDAMENTAÇÃO TEÓRICA **16**
1.1 Séries Temporais **16**
1.2 Inteligência artificial **17**
1.3 Aprendizado de Máquina **18**
1.4 Redes Neurais Artificiais **19**
1.4.1 Função de ativação 20
1.4.2 Processos de Aprendizado de uma Rede Neural Artificial 21
1.4.3 Arquitetura 21
1.4.4 Aprendizado Supervisionado 22
1.4.5 Aprendizado Não Supervisionado 22
1.4.6 Redes Não Recorrentes 22
1.4.7 Redes Neurais Recorrentes 23
1.4.8 Rede *Long Short-Term Memory* 25
1.4.9 Rede *Gated Recurrent Unit* 26
1.5 Medidas de Avaliação **26**
1.5.1 Erro Absoluto Médio 27
1.5.2 Erro Quadrático Médio 27

2 TRABALHOS RELACIONADOS **28**
2.1 Estudo de comparação de predições utilizando diferentes métodos de *Deep Learning* **29**
2.2 Estudo de predição do valor de ações utilizando variantes de RNR . **30**

3 DESENVOLVIMENTO **31**
3.1 Coleta de dados **31**
3.2 Tratamento de Dados **32**
3.3 Construção dos Modelos **33**
3.4 Definição dos Hiperparâmetros **34**
3.4.1 Camadas das Redes Neurais 34
3.4.2 Número de Neurônios de Cada Camada 34
3.4.3 *Dropout* 35
3.4.4 Otimizador 35

3.4.5 Tamanho do Lote 36
3.4.6 Função de Ativação 37
3.4.7 Número de Épocas 37
3.4.8 Tamanho da Janela 38

4 RESULTADOS OBTIDOS 40

5 TRABALHOS FUTUROS 43

CONCLUSÃO 44

REFERÊNCIAS 45

Índice 48

Introdução

A predição bem-sucedida de dados de séries financeiras temporais sempre foi muito visada. A previsão do preço de ações, *algorithmic trading*, precificação de ativos, alocação de portfólio, entre outras, são áreas que despertam interesse de pesquisadores e investidores, por causa da possibilidade de rentabilidade. Devido à natureza volátil e não linear de dados do mercado de ações, depara-se com técnicas de Aprendizado de Máquina (AM) como forma de modelar, visto que o cérebro humano possui características desejáveis em qualquer sistema artificial. Entre elas estão a flexibilidade para se adaptar a situações, aprender com a experiência, a capacidade para lidar com informações inconsistentes e de reconhecer e generalizar padrões. Sendo assim, para se alcançar boas predições, a criação de modelos de AM vem sendo amplamente explorada.

O valor de uma ação é o maior valor pelo qual alguém quer vender essa ação, ou o menor valor pelo qual alguém a quer comprar, o que faz da oferta e demanda um fator decisivo para a definição de seu preço. Se muitas pessoas querem comprar uma ação, seu preço aumenta devido à alta demanda, e se muitas pessoas a quiserem vender, a alta oferta faz o seu preço abaixar. O conceito é fácil de ser compreendido, o problema é compreender os fatores que influenciam na variação da oferta e demanda. Apesar de parecer ser uma tarefa muito complicada, há anos são usadas técnicas de inteligência artificial para prever o preço de ações, sendo possível encontrar exemplos até de 1994 (REFENES; ZAPRANIS; FRANCIS, 1994). Com o avanço da tecnologia nas últimas décadas, os modelos de *Deep Learning* (DL) vêm se destacando cada vez mais entre os modelos utilizados para esse propósito.

No contexto de predição do valor de ações, são feitos estudos com base em diversas classes de Redes Neurais Artificiais (RNA). Kim (2003) aplica SVM (*Support Vector Machine*, Máquina de Vetores de Suporte) para classificar a direção do índice do mercado de ações coreano (KOSPI) usando indicadores de análise técnica como variáveis preditivas. Kim e Han (2000) utilizou um algoritmo genético para discretização dos valores contínuos dos indicadores de análise técnica, com a finalidade de otimizar os pesos de uma RNA, resultando em uma melhor predição. Pai e Lin (2005) combinou SVM com um modelo *Autoregressive Integrated Moving Average* (ARIMA, Autorregressivo Integrado de Média Móvel) em um sistema híbrido com erros menores do que os obtidos pelos modelos utilizados separadamente. Chen, Cheng e Tsai (2014) utiliza lógica *fuzzy* em uma série temporal, com o objetivo de superar limitações relativas a suposições de linearidade presentes em outros modelos, como o ARIMA. Porém, os modelos de DL que se mostraram mais estudados e que obtiveram melhores resultados quando comparados a outros tipos de modelo nesse contexto são os de Redes Neurais Recorrentes (RNR) (Ryll; Seidens, 2019).

É importante também citar outros trabalhos como o de Wang, Xu e Zheng (2018) e de Iwasaki e Chen (2018), que mostram uma abordagem inovadora, utilizando métodos de interpretação de textos relacionados a investimentos em ações retirados da internet para melhorar significativamente a previsão de seus modelos. Wang, Xu e Zheng (2018) inclusive propõe um novo modelo chamado de *deep random ensemble*, que mistura múltiplos modelos de AM e obtém resultados impressionantes.

O objetivo desse projeto é focar na predição dos valores de ações de empresas registradas na B3, a bolsa de valores brasileira utilizada para câmbio de ações, utilizando RNRs e comparar os resultados obtidos com cada modelo. Os três tipos de RNRs utilizados para comparação nesse trabalho são: *Long Short-Term Memory* (LSTM, Memória de Curto e Longo Prazo), *Gated Recurrent Unit* (GRU, Unidade de Portão Recorrente), e a RNR simples, também conhecida como *Vanilla Recurrent Neural Network* (VRNN).

Motivação

Com o objetivo de ter o conhecimento das tendências de ações, visando um maior lucro e garantia, o interesse de investidores do mercado financeiro em métodos precisos de predição do valor de ações é muito grande. Conforme novas tecnologias surgem, são feitos estudos da acurácia dos diferentes tipos de modelos disponíveis, muitas vezes usando mais de uma técnica, com a finalidade de diminuir o erro dos modelos e por consequência aumentar a precisão das predições. Além disso, é possível utilizar outras técnicas para automatizar a compra e venda de ativos de acordo com esses modelos criados, facilitando ainda mais o processo.

Pelo interesse do autor em investimentos de ações no mercado financeiro, e com uma ampla gama de métodos disponíveis para predição dos valores futuros, esse trabalho apresentou uma oportunidade de testar algumas das técnicas de AM e validar a possibilidade do uso de modelos de RNR como uma opção viável de ser utilizada para investimento.

Objetivos

Este trabalho tem como objetivo realizar um estudo sobre RNRs aplicadas ao mercado de ações, treinando modelos com 3 *datasets* de ações diferentes encontrados na bolsa de valores brasileira, tais como FLRY3, PETR4 e VALE3. Para averiguar a precisão das predições, o Erro Quadrático Médio (EQM) e o Erro Absoluto Médio (EAM) de cada modelo será analisado. Assim, será possível concluir qual tipo RNR provê melhores resultados.

Adicionalmente, o trabalho propõe um estudo dos hiperparâmetros que podem ser modificados para esses tipos de RNA, identificando a combinação que traz o menor

erro durante o teste. Posteriormente, os hiperparâmetros considerados como ótimo são aplicados em todos os modelos testados.

1 Fundamentação teórica

1.1 Séries Temporais

Uma série temporal é uma sequência de valores de uma variável observadas ao longo do tempo. A ordem dos dados é fundamental e os valores vizinhos são dependentes. Logo, um dos interesses é analisar e modelar esta dependência.

A série temporal pode ser contínua, quando os valores são observados continuamente, ou discreta, quando os valores são observados em tempos específicos, normalmente equiespaçados. Essa terminologia se refere ao espaçamento dado entre as observações e não ao tipo da variável resposta. Conceitualmente, uma série temporal com n observações possui a seguinte representação: $X = x_1, x_2, ..., x_n$, onde x são valores de X_t, $\mathbf{X}$ é a série temporal e as variáveis x_1 a x_n são amostras da série no período $t = 1, 2, ..., n$.

Segundo a abordagem de componentes não observáveis, as séries temporais podem ser representadas como a combinação de quatro componentes: Tendência, Cíclica, Sazonal e Erro (MENDENHALL; REINMUTH; BEAVER, 1993).

As componentes de tendência frequentemente produzem mudanças graduais em longo prazo, podendo ser de crescimento ou decrescimento. Um exemplo é o crescimento constante na população.

As componentes cíclicas são aquelas que provocam oscilações de subida e de queda nas séries, de forma suave e repetitiva, ao longo da componente de tendência. Geralmente os efeitos cíclicos em uma série são causados por mudanças na demanda do produto, por ciclos de negócios e, em particular, pela inabilidade de se suprir as necessidades do consumidor.

As componentes sazonais em uma série são aquelas oscilações de subida e de queda que sempre ocorrem em um determinado período do ano, do mês, da semana, do dia ou horário. Por exemplo, é natural esperar que as vendas mensais de brinquedos terão um pico no mês de outubro, devido ao dia das crianças. Este padrão possivelmente se repetirá ao longo de vários anos.

A diferença essencial entre as componentes sazonais e cíclicas é que a primeira possui movimentos facilmente previsíveis, ocorrendo em intervalos regulares de tempo. Já os movimentos cíclicos tendem a ser irregulares.

A quarta componente da série, chamada de erro, apresenta flutuações de período curto, com deslocamento inexplicável e geralmente são causadas, entre outros motivos, por eventos políticos, econômicos e oscilações climáticas imprevisíveis.

Os modelos de previsão podem ser classificados em univariados, os quais têm a previsão dos valores futuros explicados somente pelos valores passados da própria série ou causais, ou os que levam em conta outras informações relevantes (covariáveis) como influentes para a previsão de uma variável resposta.

Segundo Shumway e Stoffer (2017), o principal objetivo da análise de séries temporais é desenvolver modelos matemáticos que consigam descrever uma certa amostra de dados, a fim de identificar o comportamento desses dados. Os modelos de previsão de séries temporais são capazes de definir, com certo grau de confiança, os valores futuros das séries a partir das observações passadas da própria série. O uso de modelos de previsão é importante, pois tem como caráter diminuir os riscos na tomada de decisão.

1.2 Inteligência artificial

Luger e Stubblefield (1993) definem Inteligência Artificial (IA) como o ramo da ciência da computação que trata da automação de comportamento inteligente. O surgimento dessa área foi um resultado natural da busca do ser humano pelo entendimento da forma como ele age e pensa. Com o desenvolvimento da ciência da computação e da neurociência, a IA passou a utilizar dessas duas áreas em busca da construção de entidades inteligentes.

A partir dessa definição, podemos dizer que um dos objetivos da IA, quando olhamos no contexto de Aprendizado de Máquina, é de construir e compreender inteligência, criando algo inteligente o suficiente para ter a capacidade de analisar dados e reconhecer padrões. Isto significa que, através da IA, é possível criar métodos preditivos analíticos (RUSSELL; NORVIG, 2010), que serão melhor explorados no decorrer deste trabalho.

Os princípios da IA podem ser encontrados no trabalho de Turing (1950), que propõe o chamado teste de Turing. O teste foi uma proposta para verificar o nível de inteligência de uma máquina, em que, caso o entrevistador humano não consiga diferenciar se o objeto entrevistado é uma máquina ou outro ser humano, essa máquina pode ser considerada inteligente. Segundo Russell e Norvig (2010), para uma máquina ser considerada aprovada no teste, ela precisa ter as seguintes competências:

- **Processamento de linguagem natural**, para possibilitar a comunicação;
- **Representação de conhecimento**, para armazenar o que ela aprende ou vê;
- **Raciocínio automatizado**, para usar a informação armazenada para responder perguntas e tirar novas conclusões;
- **Aprendizado de máquina**, adaptar a novas circunstâncias e detectar e extrapolar padrões.

Além dessas quatro competências, o chamado teste de Turing total, uma versão aprimorada, acrescenta ainda mais dois pontos:

- **Visão computacional**, para compreender objetos;
- **Robótica**, para manipular objetos e conseguir se movimentar.

Essas seis grandes áreas compreendem a maioria dos estudos e pesquisas de IA, fazendo o teste de Turing continuar sendo relevante mesmo décadas depois de ter sido proposto.

Nesse contexto, com o avanço da tecnologia, a capacidade computacional torna-se suficiente para grandes aplicações envolvendo Aprendizado de Máquina, que é um sub-área de Inteligência Artificial que usa algoritmos de aprendizado para construir sistemas que têm a capacidade de aprender e melhorar automaticamente com as experiências sem serem programados explicitamente.

No Aprendizado de Máquina, treinamos o algoritmo fornecendo-lhe muitos dados e permitindo que ele aprenda mais sobre as informações processadas. Conforme avançamos para a era do *Big Data*, temos um volume enorme de dados disponível para treinar essas aplicações. Visto que a capacidade humana de processamento de grandes volumes de dados parece limitada diante do potencial desses sistemas automatizados, eles podem trazem consigo a capacidade de remodelar a sociedade e a forma como ela trabalha, principalmente se tratando de situações específicas em que os algoritmos podem se especializar.

Ainda nessa linha, tem-se o conceito de Deep Learning, que procura imitar o cérebro humano e que pode ser definido como uma arquitetura de rede multi neural contendo um grande número de parâmetros e camadas.

Na figura Figura 1 é possível ver o diagrama de Venn que elucida a composição de Inteligência Artificial, Aprendizado de Máquina e *Deep Learning*.

1.3 Aprendizado de Máquina

A área de Aprendizado de Máquina está relacionada com a questão de como construir programas de computadores que aprendem e aperfeiçoam-se automaticamente através da experiência (MITCHELL, 1997). Os algoritmos de AM têm o intuito de achar padrões em um conjunto de dados e se otimizar através desses padrões. Esse tipo de algoritmo é responsável pela maior parte dos avanços e aplicações de IA utilizados no dia-a-dia, como por exemplo as recomendações dadas em diversos serviços de *streaming*, em detecção de fraudes, diagnóstico médico por imagens, *chatbots*, entre outros.

Uma grande parte de AM consiste na matemática usada em aprendizagem estatística e otimização de dados. Por isso, é muito importante a forma como os dados são

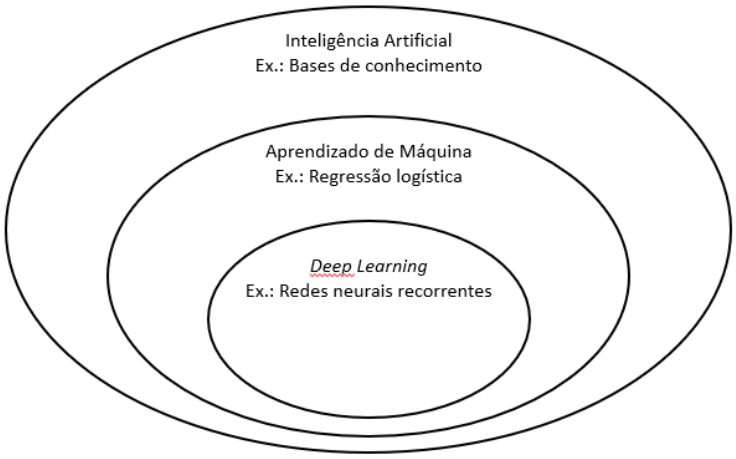


Figura 1 – Ilustração de um diagrama de Venn mostrando relação entre Inteligência Artificial, Aprendizado de Máquina e *Deep Learning*. Fonte: (Exastax, 2017), adaptada pelo autor.

representados. No contexto de predição através de redes neurais, se uma base de dados utilizada para a aprendizagem for bem estruturada, o algoritmo tende a reconhecer melhor os padrões do conjunto de dados.

1.4 Redes Neurais Artificiais

Redes Neurais Artificiais, também chamadas apenas de Redes Neurais, são uma estrutura computacional que atuam de forma semelhante à de neurônios biológicos. Os neurônios artificiais são conectados uns aos outros através de um mecanismo composto por um conjunto de pesos atribuídos (MOGHADDAM; MOGHADDAM; ESFANDYARI, 2016).

Uma RNA é projetada para identificar uma tendência básica em um conjunto de dados, e fazer uma generalização a partir dela. A maior vantagem de uma RNA, quando comparada a outras técnicas de AM, é a sua capacidade de aprender os padrões subjacentes dos dados, onde a maioria dos métodos convencionais tende a falhar (ZHANG; PATUWO; HU, 1998).

RNAs são construídas com base em uma rede de unidades de processamento que buscam simular um neurônio e que, nesse contexto, também são chamadas de neurônio. A conexão entre os neurônios são chamadas de sinapses, e cada neurônio que compõe a rede recebe informação de outro como entrada através das sinapses. Essa informação de entrada é multiplicada por um peso definido pelo construtor da rede, que é usado para ajustar a importância de diferentes resultados computacionais obtidos pelo neurônio.

O neurônio artificial, representado na Figura 2, possui 'i' entradas (x_i), e cada entrada é conectada ao neurônio através de sinapses com pesos (w_i). O neurônio soma as

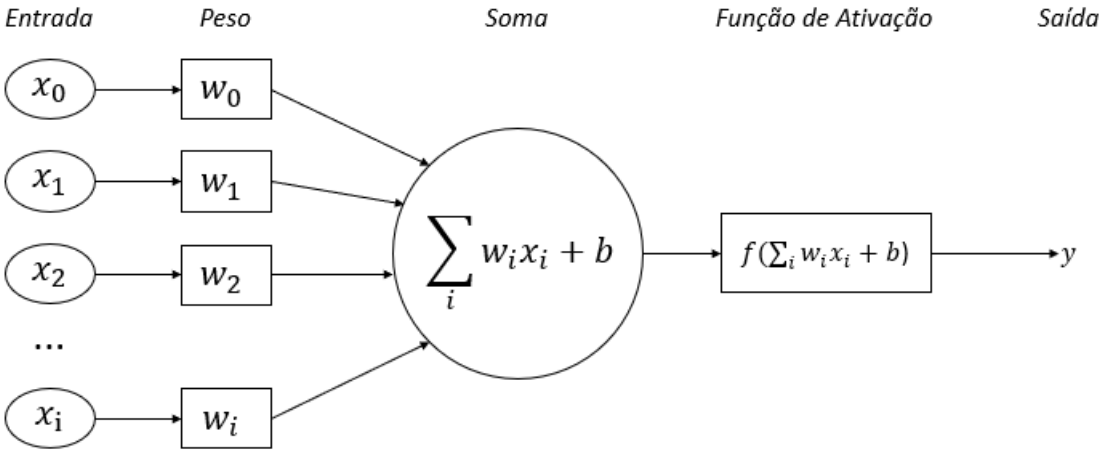


Figura 2 – Representação de um neurônio artificial. Fonte: elaborada pelo autor.

entradas multiplicadas pelos pesos através da equação:

$$A = \sum_i x_i w_i + b$$

Onde A é o resultado da soma e b é o valor do viés (*bias*). Para calcular a saída, à soma é aplicada uma função chamada de função de ativação (HIRANSHA et al., 2018).

$$y = f(A)$$

1.4.1 Função de ativação

O propósito da função de ativação é de restringir a saída de acordo com o problema proposto. Como dados financeiros tendem a ser não-lineares, esse fato sugere que funções não-lineares são mais apropriadas, tais como a função Sigmoide (Equação 1.1) e Tangente Hiperbólica (Equação 1.2). Elas são comumente utilizadas em dados de séries temporais, porque são continuamente diferenciáveis, sendo esta uma propriedade desejável no treinamento da rede.

- Função Sigmoide

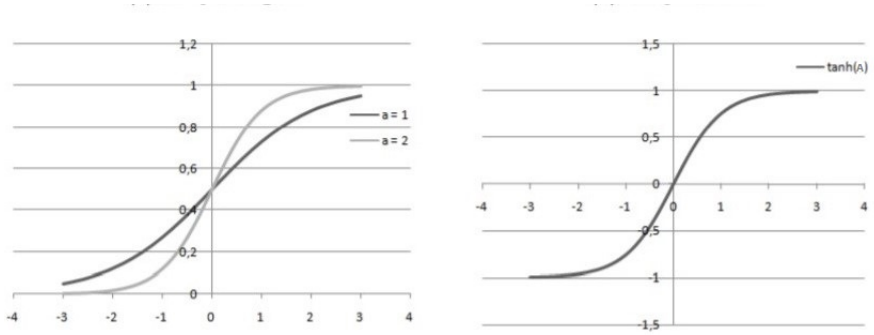
$$f(A) = \frac{1}{1 + e^{-\alpha A}} \tag{1.1}$$

Na função sigmoide, o α é o parâmetro de inclinação.

- Função Tangente Hiperbólica

$$f(A) = tanh(A) \tag{1.2}$$

A Figura 3 ilustra graficamente as duas funções de ativação mencionadas:



(a) Função Sigmóide (b) Função Tangente Hiperbólica

Figura 3 – Funções de ativação. Fonte: elaborada pelo autor.

1.4.2 Processos de Aprendizado de uma Rede Neural Artificial

De acordo com GIRIOLI e RIBEIRO (2009):

> A rede neural se baseia nos dados para extrair um modelo geral. Portanto, a fase de aprendizado deve ser rigorosa. De 50% a 90% do total de dados devem ser separados para o treinamento da rede neural, dados estes escolhidos aleatoriamente, a fim de que a rede "aprenda" as regras e não "decore" exemplos. O restante dos dados só é apresentado à rede neural na fase de testes a fim de que ela possa "deduzir" corretamente o inter-relacionamento entre os dados.
>
> O próximo passo é a definição da configuração da rede, que normalmente é feita de forma empírica. Em seguida, há o treinamento da rede. Nesta fase serão ajustados os pesos das conexões. Uma boa escolha dos valores iniciais dos pesos da rede pode diminuir o tempo necessário para o treinamento. O treinamento deve ser interrompido quando a rede apresentar uma boa capacidade de generalização e quando a taxa de erro for suficientemente pequena, ou seja, menor que um erro admissível.
>
> O último passo é o teste da rede. Durante esta fase o conjunto de teste é utilizado para determinar a performance da rede com dados que não foram previamente utilizados. O desempenho da rede, medido nesta fase, é uma boa indicação de sua performance real.

1.4.3 Arquitetura

A forma como os neurônios são agrupados e conectados é conhecida como arquitetura da rede neural. Combinações de diferentes características podem ser feitas a fim de se obter redes com arquiteturas e aplicações bastante distintas.

Antes, é preciso definir dois aspectos importantes: o algoritmo de aprendizado que será implementado, sendo as formas mais comuns o algoritmo supervisionado e o não

supervisionado; e a maneira como os sinais se propagam dentro da sua estrutura interna, determinando se a rede possui retroalimentação ou não, podendo ser classificada como não recorrente, recorrente, entre outras(VELLASCO, 2007).

1.4.4 Aprendizado Supervisionado

No aprendizado supervisionado, a rede tem sua saída atual comparada com a saída desejada, recebendo informações sobre o quão apurada está a resposta atual, ou seja, são exibidos os valores corretos para cada par entrada-saída, de forma que a RNA possa ajustar seus pesos e diminuir o erro geral de treinamento.

Outro aspecto a ser determinado é de quando ocorrerá o processo de alteração dos pesos. Um método de treinamento que requer o processamento de todos pares-entradas (uma época) para só depois calcular a mudança nos pesos da rede é chamado de método de treinamento por lote (batch no inglês) ou por ciclo (SILVA; OLIVEIRA, 2001).

Existem métodos de treinamento que propõem que as alterações nos pesos sejam feitas após o processamento de cada par entrada-saída, sem esperar que todos os padrões sejam apresentados. Tais métodos são chamados de métodos de treinamento online ou Delta.

Os exemplos mais conhecidos de algoritmos de aprendizagem supervisionada são a regra delta e a sua generalização para redes de múltiplas camadas, o algoritmo backpropagation.

1.4.5 Aprendizado Não Supervisionado

No algoritmo não supervisionado, não existe um vetor de resposta desejada. Logo, a RNA realiza uma forma de compressão nos dados, agrupando os valores que apresentam padrões similares.

A rede neural reconhece um padrão passando por uma seção de treinamento, durante a qual se apresenta repetidamente um conjunto de padrões de entrada junto com a categoria a qual cada um pertence. Em seguida, apresenta-se à rede um padrão que nunca foi visto, mas que pertence a população de padrões utilizados para o treinamento e a rede. Por causa da informação extraída no aprendizado, a rede é capaz de identificar a categoria correta daquele padrão particular (VELLASCO, 2007).

Para esse tipo de treinamento pode-se utilizar a regra de aprendizagem competitiva.

1.4.6 Redes Não Recorrentes

Outra característica a ser determinada é a direção do sinal de propagação das conexões, podendo ser *feedforward* ou *backward*, como ilustrado na Figura 4.

Nas redes não recorrentes ou redes diretas, o fluxo de sinal é apenas em um sentido.

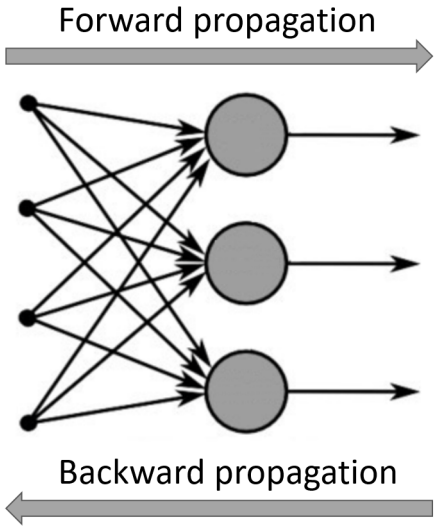


Figura 4 – Sentido de propagação. Fonte: (ACADEMY, 2019).

1.4.7 Redes Neurais Recorrentes

Nas redes recorrentes, existe pelo menos um ciclo de retroalimentação, em que o sinal retorna para uma camada anterior, tornando o treinamento mais complexo, mas especializando a rede para certos tipos de aplicação, como dados sequenciais. A Figura 5 ilustra esse tipo de conexão.

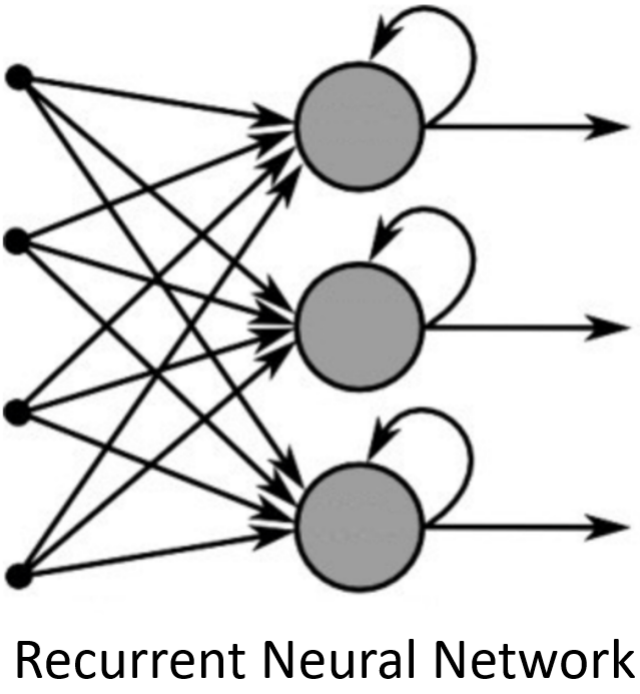


Figura 5 – Rede recorrente. Fonte: (ACADEMY, 2019).

Em mais detalhes, Redes Neurais Recorrentes são uma classe de redes neurais que recebe entrada de duas fontes: uma do presente, e outra de um ponto passado. A informação das duas fontes são utilizadas para decidir como reagir a uma nova entrada de dados, e isso é feito através de um circuito de *feedback*, onde a saída de cada instante é uma entrada para o instante seguinte. Devido a essa característica, podemos dizer que elas possuem memória, fazendo das RNRs mais semelhantes da forma como humanos processam informação, possibilitando o reconhecimento de um contexto através da memória (Zaremba; Sutskever; Vinyals, 2014).

Normalmente, RNRs são compostas por 3 tipos de camadas: camada de entrada, camada intermediária ou escondida, e camada de saída. Cada neurônio na camada de entrada é conectado com cada neurônio da camada intermediária seguinte, até a camada de saída. Em RNRs, é comum os modelos terem mais de uma camada intermediária.

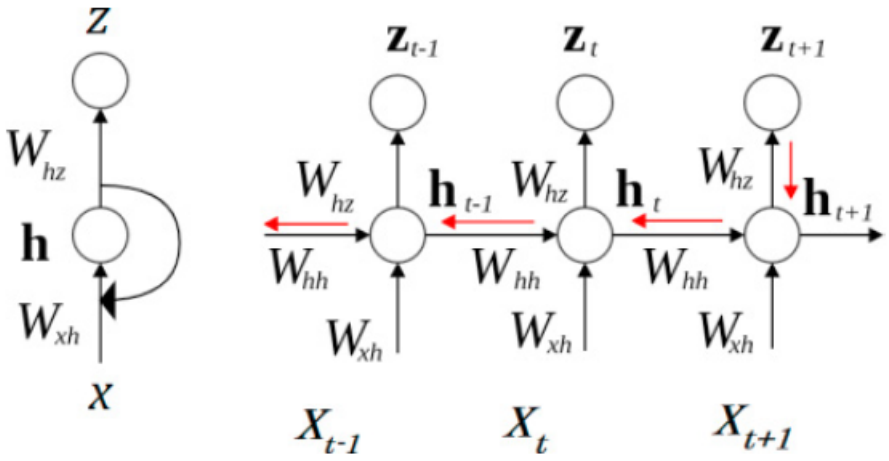


Figura 6 – Rede Neural Recorrente. Fonte: Hiransha et al. (2018).

A Figura 6 mostra uma representação de redes neurais recorrentes, onde a equação de entrada das camadas intermediárias é dada por:

$$h_t = g_n(W_{xh}X_t + W_{hh}h_{t-1} + b_h),$$

onde h_t é a camada intermediária no instante t, g_n é a função de ativação, W_{xh} é a matriz de peso de entrada, X_t é a entrada no instante t, h_{t-1} é a camada intermediária no instante $t-1$, W_{hh} é a matriz de peso do neurônio recorrente, e b_h é o valor de viés.

Já a equação da saída da camada intermediária é dada por:

$$Z_t = g_n(W_{hz}h_t + b_z),$$

onde Z_t é o vetor de saída, W_{hz} é a matriz de peso para a camada de saída, e b_z é o viés (HIRANSHA et al., 2018).

O principal problema da RNR é que o gradiente da função de custo decai exponencialmente com o tempo e o modelo para de aprender. Esse problema é chamado de problema da dissipação do gradiente (HAYKIN, 2008).

1.4.8 Rede *Long Short-Term Memory*

A rede LSTM é uma variante especial de RNR capaz de aprender dependências de longo prazo. Ela foi projetada especificamente para evitar o problema de dissipação/explosão do gradiente. Uma LSTM é adequada para classificar e prever dados de séries temporais. A Figura 7 mostra uma representação gráfica da LSTM.

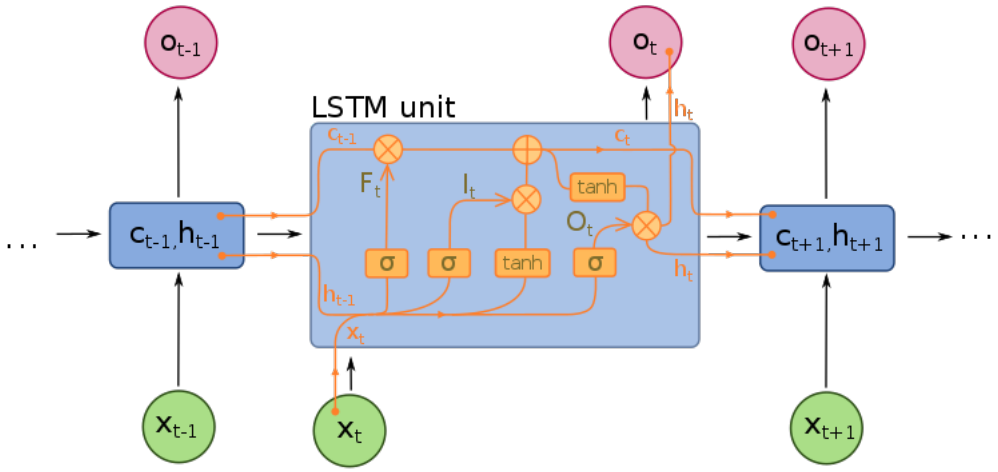


Figura 7 – Célula LSTM. Fonte: Wikipedia contributors (2020).

Essa arquitetura comum da célula de LSTM é composta por um estado da célula (c_t), uma saída chamada de *hidden state* (h_t), um *input gate* (I_t), um *output gate* (O_t) e um *forget gate* (F_t). As equações de uma célula LSTM podem ser definidas como:

$$I_t = \sigma(W_{xi}X_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$

$$F_t = \sigma(W_{xf}X_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$

$$c_t = F_t \times c_{t-1} + I_t \times tanh(W_{xc}X_t + W_{hc}h_{t-1} + b_c)$$

$$O_t = \sigma(W_{xo}X_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$

$$h_t = O_t \times tanh(c_t)$$

Como podemos ver na Figura 7, o estado da célula é uma linha horizontal que passa através da rede inteira, carregando informação da célula anterior para a atual, e assim por diante. A decisão de guardar informação no estado da célula é tomada pelo *forget gate*. A saída do *forget gate* é adicionada ao estado da célula através de uma operação de multiplicação elemento a elemento. Já o *input gate* é composto por uma camada sigmoide e uma camada de tangente hiperbólica, e junta essas duas no estado da célula. A saída é formada por uma multiplicação elemento a elemento do *output gate* e a tangente hiperbólica (HIRANSHA et al., 2018).

1.4.9 Rede *Gated Recurrent Unit*

GRU é um outro tipo de arquitetura de RNR que procura resolver o problema de dissipação do gradiente. É uma rede similar à LSTM, mas de estrutura um pouco mais simplificada. A GRU utiliza apenas dois tipos de portões: *update gate* e *reset gate* (Cho et al., 2014). Esse tipo de rede possui menos parâmetros, e por isso geralmente é treinada mais rapidamente ou com um número menor de dados, mas, com um maior volume de dados, as LSTMs podem acabar apresentando resultados melhores (WEISS; GOLDBERG; YAHAV, 2018). A Figura 8 mostra uma representação da célula de GRU.

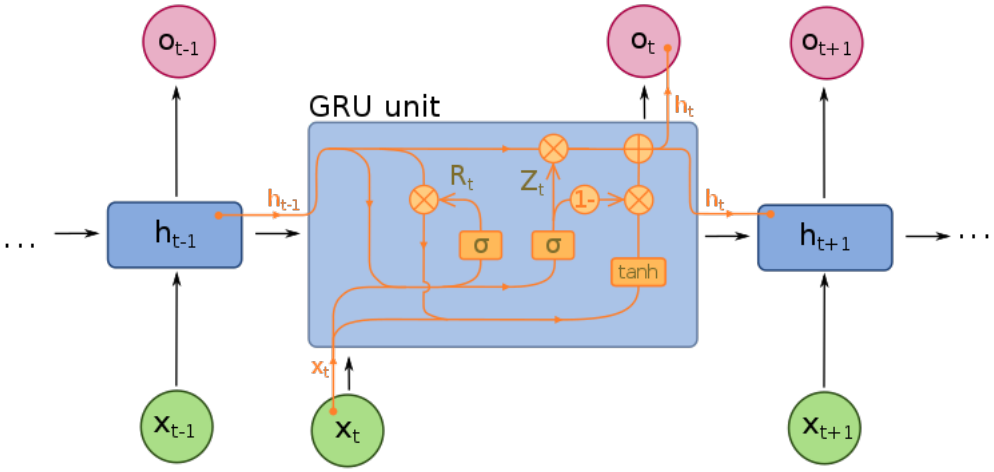


Figura 8 – Célula GRU. Fonte: Wikipedia contributors (2020).

As equações da célula GRU podem ser definidas como:

$$Z_t = \sigma(W_x X_t + U_z h_{t-1} + b_z)$$

$$R_t = \sigma(W_r X_t + U_r h_{t-1} + b_r)$$

$$\hat{h_t} = tanh(W_h X_t + U_h(R_t \times h_{t-1}) + b_h)$$

$$h_t = (1 - Z_t) \times h_{t-1} + Z_t \times \hat{h_t}$$

Nas equações acima, x_t representa o vetor de entrada, h_t o vetor de saída, $\hat{h_t}$ é o vetor candidato a ativação, z_t é o vetor do *update gate*, e rt é o vetor do *reset gate*.

1.5 Medidas de Avaliação

A forma mais comum de medir a acurácia de um modelo de rede neural que prevê os valores futuros de ações é através do cálculo de erro do modelo. Quanto menor o erro de um modelo, melhor ele se ajustou ao conjunto de dados e mais próximo sua predição chegou dos valores reais. Nessa seção serão apresentados as duas medidas utilizadas ao longo desse trabalho: o Erro Absoluto Médio (EAM) e o Erro Quadrático Médio (EQM).

1.5.1 Erro Absoluto Médio

EAM é uma fórmula matemática que pode ser utilizada para calcular o erro das previsões dos modelos. É uma parte importante para o relatório, pois é fortemente baseado em estimativas e previsões futuras de conjuntos de dados. o erro absoluto médio é calculado através da diferença do valor real e do valor predito dividido pelo número de dados:

$EAM = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n}$,

onde x_i é o i-ésimo valor real, y_i é o i-ésimo valor predito e n é o número de dados utilizados.

1.5.2 Erro Quadrático Médio

O EQM é um outro método de calcular a acurácia e o erro nos modelos preditivos utilizados nesse trabalho. A equação do erro quadrático médio pode ser definida como:

$EQM = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2$,

onde x_i é o i-ésimo valor real, y_i é o i-ésimo valor predito e n é o número de dados utilizados.

2 Trabalhos Relacionados

Nessa seção, serão apresentados trabalhos que abordam problemas semelhantes aos tratados nesse estudo, que utilizaram abordagens diversas, e que serviram de aprendizado para o aprofundamento no tema previsão de redes neurais aplicadas ao mercado financeiro.

Iwasaki e Chen (2018) mostram em seu trabalho uma abordagem interessante, onde desenvolvem uma rede neural profunda que, além de utilizar os dados de ações da bolsa de valores de Tóquio e Osaka, considera também opinião de analistas, que indicam ações de relevância através de relatórios, usando métodos de interpretação de texto. Essa abordagem foca principalmente em dois tipos de redes neurais: a LSTM e a rede neural convolucional.

Já Wang, Xu e Zheng (2018) utilizam um algoritmo que procura em artigos de notícias de várias fontes e também em redes sociais opiniões do público em geral para influenciar seu modelo de predição, e finalmente propõem um novo modelo de aprendizado de máquina chamado *Deep Random Subspace Ensembles* (DRSE), que integra algoritmos de *deep learning* e métodos de *ensemble learning* (uma técnica que combina múltiplos algoritmos de aprendizado) para uma melhor mineração de dados, e mostra resultados impressionantes.

Além desses citados anteriormente, foram encontrados dois artigos que abordaram com a comparação de diferentes tecnologias de *Deep Learning* aplicadas na predição de valores de ações e de índices de ações, que possuem uma relação mais próxima com este trabalho, e serão melhor detalhados a seguir. Os dois trabalhos possuem objetivos semelhantes, com um deles tratando métodos de *Deep Learning* de forma mais abrangente, e o outro focando nos modelos de Rede Neural Recorrente.

O primeiro artigo consiste em um estudo da predição dos valores de ações de setores variados registradas na National Stock Exchange (NSE) da Índia e na New York Stock Exchange (NYSE), realizado pelos autores Hiransha et al. (2018), publicado na revista "Procedia Computer Science". Seu artigo é intitulado "NSE Stock Market Prediction Using Deep-Learning Models" (HIRANSHA et al., 2018).

O segundo trabalho é um outro artigo também publicado na revista "Procedia Computer Science" e foi desenvolvido pelos autores Saud e Shakya (2020). Com o título de "Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE", ele foca em predizer os as ações do Nepal Investment Bank (NIB) e Nabil Bank Limited (NABIL) e comparar os resultados das redes neurais recorrentes (SAUD; SHAKYA, 2020).

Em ambos os trabalhos, os autores têm como foco a comparação de predições utilizando diferentes métodos, que serão descritos nas próximas seções.

2.1 Estudo de comparação de predições utilizando diferentes métodos de *Deep Learning*

O trabalho apresentado por Hiransha et al. (2018) aplica diversas técnicas de *Deep Learning*, incluindo redes neurais recorrentes, Perceptron Multicamadas (MLP, *Multilayer Perceptron*) e Redes Neurais Convolucionais (CNN, *Convolutional Neural Network*), para analisar dois mercados internacionais de ações diferentes, treinando os modelos com apenas uma ação do setor automotivo e os validando com ações dos setores bancário, automotivo, energético e de tecnológico.

Os autores utilizam conjuntos de dados contendo o símbolo da ação, preços de abertura, máximo, mínimo, último, fechamento, e médio, quantidade total negociada, volume de negócios e número de negociações. Desses dados, eles extraem apenas o preço de fechamento diário de cada ação, já que os investidores tomam a decisão de comprar ou vender cada ação baseada no preço de fechamento do mercado (HIRANSHA et al., 2018).

Através de várias predições utilizando diferentes parâmetros de tamanho da janela, que é o número de dados anteriores que são utilizados para realizar a predição, a fim de descobrir o tamanho de janela ótimo, com o menor valor de erro absoluto médio percentual, para ser utilizado nas comparações.

Definido o tamanho da janela que será utilizado, os autores treinam modelos RNR, LSTM, CNN e MLP utilizando dados da empresa automotiva TATA MOTORS de janeiro de 1996 até junho de 2015, e testando os modelos com dados das empresas Maruti, Axis Bank e HCL Technologies, com conjuntos de dados de 2007 até 2017. Depois desse primeiro teste, também é utilizado como teste conjuntos de dados do Bank of America e Chesapeak Energy, a fim de analisar se o modelo era capaz de identificar se o modelo era capaz de identificar padrões até de mercados distintos de ações.

A semelhança do trabalho de Hiransha et al. (2018) com o trabalho aqui proposto é a comparação do desempenho de métodos diferentes de aprendizado de máquina a fim de comparar a precisão de cada método. Aqui também serão feitos testes com tamanhos distintos de janela e comparados os valores de erro após a validação dos modelos.

A principal diferença é que o trabalho desenvolvido por Hiransha et al. (2018) treina apenas um modelo para cada tipo de rede neural utilizando o conjunto de dados histórico da TATA MOTORS e validando com dados de outras empresas, enquanto esse trabalho faz o treinamento e validação das redes neurais utilizando os dados da mesma empresa.

2.2 Estudo de predição do valor de ações utilizando variantes de RNR

Saud e Shakya (2020) apresentam em seu trabalho um estudo de predição de ações no setor bancário da bolsa de valores do Nepal utilizando variações de redes neurais recorrentes. As variações estudadas nesse trabalho são: VRNN, LSTM e GRU.

Utilizando dados históricos de 2007 até 2018, foram utilizados modelos com 6 camadas (uma camada de entrada, quatro intermediárias e uma de saída), com a camada de entrada composta por 14 neurônios, as intermediárias compostas por 50 neurônios cada, e a de saída composta por apenas um neurônio. Os conjuntos de dados são pré-processados antes do treinamento dos modelos, e o atributo a ser predito é o valor do fechamento do dia seguinte. Os atributos utilizados como entrada são: valor de abertura, fechamento, máximo, mínimo, volume negociado, valor total negociado, máximo-mínimo, abertura-fechamento, média móvel de 3 dias, média móvel de 10 dias, média móvel de 30 dias, desvio padrão, índice de força relativa e Williams' %R (SAUD; SHAKYA, 2020).

Os autores utilizam de diversos gráficos dos valores reais e preditos das ações do Nepal Investment Bank e Nabil Bank, comparando o erro mínimo obtido por cada modelo e utilizando 7 tamanhos diferentes da janela: 2, 5, 10, 15, 20, 25 e 30 dias. São feitos 10 experimentos para cada tamanho de janela, e o menor erro absoluto médio percentual obtido pra cada janela é comparado entre VRNN, LSTM e GRU.

Os tipos de redes neurais utilizadas nesse trabalho foram inspiradas pelo artigo de Saud e Shakya (2020), mas a intenção deste é utilizar ações de empresas de áreas diferentes, não só do setor bancário, e fazer um teste mais aprofundado nos hiperparâmetros utilizados.

3 Desenvolvimento

Este capítulo tem a finalidade de descrever como os hiperparâmetros dos modelos foram definidos. Para medir a acurácia de diferentes tipos de redes neurais recorrentes utilizando diferentes ações da Bolsa de Valores brasileira, foi implementado um modelo utilizando a linguagem de programação Python, com o *framework* Keras e com base na biblioteca para aprendizado de máquina Tensorflow.

O Python é uma linguagem de programação amplamente utilizada que permite abstração de dados e possui uma biblioteca padrão extensa (LUSZCZEK; DONGARRA, 2007). A linguagem ganha destaque pela facilidade de implementação e por um grande número de funcionalidades, dispondo de bibliotecas que auxiliam o programador no desenvolvimento de projetos de AM de qualquer tipo de finalidade, como Keras, Tensorflow e Scikit-learn.

3.1 Coleta de dados

Os dados do preço histórico das ações da Petrobras (PETR4), Grupo Fleury (FLRY3) e Vale (VALE3) foram coletadas do site Yahoo! Finance e baixadas no formato *comma separated value* (.csv) (YAHOO!..., 2020). O conjunto de dados extraído continha dados de 01 de janeiro de 2010 até 31 de dezembro de 2019, possuindo informações da data, valor de abertura, valor máximo, valor mínimo, valor de fechamento, valor de fechamento ajustado e volume.

Date	Open	High	Low	Close	Adj Close	Volume
2010-01-04	36.950001	37.320000	36.820000	37.320000	31.196165	13303600.0
2010-01-05	37.380001	37.430000	36.799999	37.000000	30.928682	21396400.0
2010-01-06	36.799999	37.500000	36.799999	37.500000	31.346632	18720600.0
2010-01-07	37.270000	37.450001	37.070000	37.150002	31.054062	10964600.0
2010-01-08	37.160000	37.389999	36.860001	36.950001	30.886881	14624200.0
...	...	...	...	...	...	...
2018-12-20	22.309999	22.320000	21.350000	21.490000	20.106049	141767000.0
2018-12-21	21.400000	21.980000	21.250000	21.549999	20.162184	90384300.0
2018-12-26	20.590000	21.680000	20.420000	21.680000	20.981472	77104200.0
2018-12-27	21.520000	21.980000	21.280001	21.670000	20.971794	74083700.0
2018-12-28	22.110001	22.830000	22.080000	22.680000	21.949251	61634700.0

Tabela 1 – Exemplo do conjunto de dados (PETR4) extraído do site Yahoo!... (2020), mostrando as 5 primeiras e 5 últimas linhas

Os dados presentes na Tabela 1 são descritos como:

- ***Date* (Data):** data do dia que ocorreram as transações.
- ***Open* (Abertura):** preço da ação durante a abertura do mercado, antes de iniciarem as negociações do dia.
- ***High* (Máximo):** valor mais alto que a ação atingiu durante as negociações no dia.
- ***Low* (Mínimo):** valor mais baixo que a ação atingiu durante as negociações no dia.
- ***Close* (Fechamento):** valor da ação no momento em que o horário de negociações do dia chegou ao fim.
- ***Adjusted Close* (Fechamento Ajustado):** valor de fechamento ajustado para incluir quaisquer distribuições de proventos e ações corporativas que ocorreram em qualquer momento antes da abertura do dia seguinte.
- ***Volume* (Volume):** número de ações negociadas durante o dia.

O objetivo desse trabalho é fazer a predição dos valores de fechamento citados acima.

3.2 Tratamento de Dados

Antes de utilizar o conjunto de dados para treinamento dos modelos de redes neurais, é necessário fazer um pré-processamento com o objetivo de retirar do conjunto qualquer tipo de informação que possa atrapalhar o aprendizado das redes neurais, e adicionar informações relevantes que melhorem o desempenho dos modelos.

Primeiramente, são retiradas as linhas que contenham valores nulos (*null*) ou zeros para limpar informações irrelevantes do *dataset* (por exemplo, dias em que o volume de ações negociadas era 0, indicando possivelmente um feriado, o que não traria nenhuma informação relevante para o treinamento dos modelos). Em seguida, a coluna de data é excluída, e todo o conjunto de dados é normalizado para valores entre 0 e 1 através da função MinMaxScaler(). Essa normalização é importante pois aumenta a acurácia de vários modelos de aprendizado de máquina (JAMES et al., 2018), e é feita através da equação:

$$x_{norm} = \frac{(x - xmin)}{(x_{max} - x_{min})}$$

A Tabela 2 mostra a base de dados após essa normalização.

De acordo com o trabalho de Rui e Lee (2000), o volume negociado de uma certa ação não influencia diretamente em seu preço no futuro. Ao testar modelos treinados por uma base de dados com e sem a coluna de volume, foi possível observar ainda que o modelo

Open	High	Low	Close	Adj Close	Volume
0.985555	0.994583	0.992413	0.994595	0.994519	0.019004
0.998495	0.997893	0.991806	0.984985	0.984774	0.030583
0.981041	1.000000	0.991806	1.000000	1.000000	0.026755
0.995185	0.998495	1.000000	0.989490	0.989342	0.015658
0.991875	0.996690	0.993627	0.983484	0.983252	0.020894
...	...	...	...	...	...
0.544989	0.543184	0.522914	0.519219	0.590511	0.202804
0.517605	0.532952	0.519879	0.521021	0.592556	0.129288
0.493229	0.523924	0.494689	0.524925	0.622402	0.110287
0.521216	0.532952	0.520789	0.524625	0.622050	0.105966
0.538971	0.558531	0.545068	0.554955	0.657658	0.088154

Tabela 2 – Base de dados após normalização

que não utilizava a coluna de volume para fazer as predições obteve melhor resultado, e por isso essa informação não foi utilizada para os outros testes. Na seção 3.4 é mostrado o resultado que levou a essa escolha.

Para melhorar e facilitar o treinamento das redes neurais, foram adicionadas 3 medidas de Médias Móveis (MM): as MMs dos últimos 3, 10 e 30 dias (SAUD; SHAKYA, 2020). A MM é uma medida que pode ser utilizada para fazer predições simples dos valores de ações Lauren e Harlili (2014), consegue diminuir o erro resultante das predições com redes neurais artificiais.

3.3 Construção dos Modelos

Com os dados necessários para o aprendizado dos modelos já tratados, foram construídos o modelos de rede neural LSTM, VRNN e GRU em linguagem Python utilizando a interface de implementação para algoritmos de aprendizado de máquina TensorFlow (ABADI et al., 2015). A *Application Programming Interface* (API) de *deep learning* que é executada através do TensorFlow é chamada de Keras (CHOLLET et al., 2015).

Os modelos foram construídos utilizando os mesmos hiperparâmetros e número de camadas, e os parâmetros ótimos a serem utilizados foram definidos com o modelo LSTM servindo de teste. O conjunto de dados foi separado, sendo 90% utilizado para treinamento, e 10% utilizado para teste. A definição dos parâmetros ótimos é descrita detalhadamente na próxima seção.

3.4 Definição dos Hiperparâmetros

Hiperparâmetros são parâmetros cujos valores são utilizados para controlar o processo de aprendizado de redes neurais. Diferentemente dos parâmetros de peso dos neurônios, que é calculado através do treinamento dos modelos, os hiperparâmetros devem ser definidos pelo programador. Neste trabalho, foram feitos testes para valores diferentes dos parâmetros de número de camadas, número de neurônios de cada camada, porcentagem de *dropout*, otimizador, tamanho do lote e função de ativação.

Para o número de camadas das redes, foram feitos diferentes testes separadamente, e para os outros parâmetros, foi utilizado inicialmente o Tensorboard Abadi et al. (2015) para treinar 72 modelos com parâmetros diferentes a fim de encontrar um modelo que servisse de base, com o conjunto de dados da ação FLRY3 e número de épocas e tamanho da janela fixos.

A partir do modelo com o menor valor de erro dentre os treinados e analisados através do Tensorboard, foram feitos novos testes, dessa vez com o número de épocas maior, variando os hiperparâmetros um a um, e analisando os resultados de erro novamente. O modelo com menor erro absoluto médio observado no Tensorboard possuiu os seguintes valores: 256 neurônios para cada camada, 20% de *dropout*, otimizador Adam, lote de 64 amostras e função de ativação *tanh*.

3.4.1 Camadas das Redes Neurais

Ao construir os modelos, é necessário definir o número de camadas que os modelos irão utilizar. Não há uma definição exata para qual o número de camadas ou neurônios que um modelo deve ter para obter uma melhor performance, portanto esse tipo de parâmetro deve ser testado dependendo da finalidade do modelo e das características do conjunto de dados (STATHAKIS, 2009). Para esse trabalho, um modelo utilizando 4 camadas (1 de entrada, 2 intermediárias e 1 de saída) se mostrou mais preciso.

3.4.2 Número de Neurônios de Cada Camada

Cada camada de uma rede neural recorrente tem um número de neurônios. A camada de saída foi fixada em apenas 1 neurônio, e as outras 3 camadas tiveram seu número de neurônios variado entre 32, 64, 128 e 256 para cada uma das camadas.

Na Figura 9, podemos ver que os menores valores de EQM e EAM foram obtidos para camadas de 128 neurônios cada, portanto esse valor será utilizado como novo padrão para os outros testes.

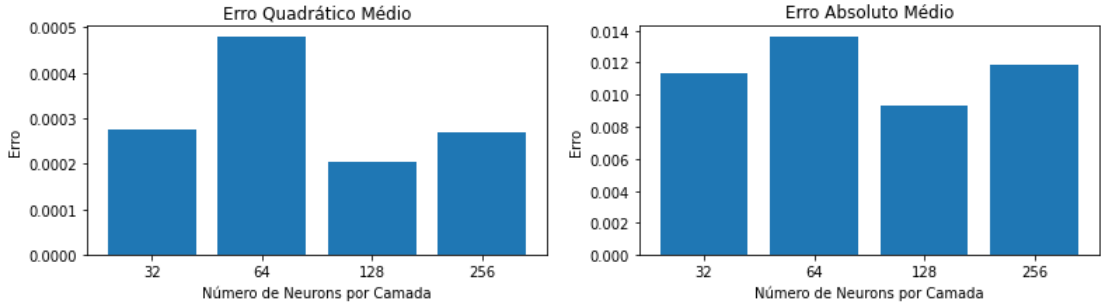


Figura 9 – Valores mínimos de erro obtidos para diferentes números de neurônios.

3.4.3 *Dropout*

O *dropout* é um parâmetro importante pois reduz o chamado sobreajuste (do inglês *overfitting)*, um tipo de erro de modelação que acontece quando o modelo se ajusta excessivamente ao conjunto de dados utilizado para o treinamento, de forma que o modelo perde a capacidade de generalização. Um modelo sobreajustado tem bons resultados para prever os valores do conjunto de testes, mas tem baixo desempenho para o conjunto de validação do modelo.

O fator de *dropout* escolhe aleatoriamente uma porcentagem das células dentro de cada camada do modelo e substitui o valor do *output* por 0 (SRIVASTAVA et al., 2014).

A Figura 10 ilustra os valores mínimos de *dropout* obtidos pelo modelo ao longo dos treinamentos. Pode-se perceber que o valor ótimo de *dropout* é 30% por ter menor EQM e EAM, portanto será esse o valor usado para o treinamento dos próximos modelos.

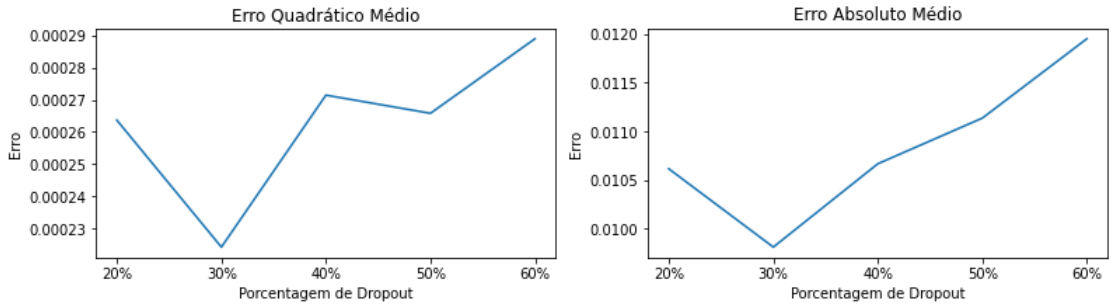


Figura 10 – Valores mínimos de erro obtidos para diferentes porcentagens de *dropout*.

3.4.4 Otimizador

Otimizadores são algoritmos utilizados em redes neurais com o intuito de minimizar o erro, identificando os valores de peso ótimos a serem empregados na rede neural baseado nos dados de treinamento através do valor mínimo local da função. Os otimizadores mais comuns utilizados para RNRs são o Adam, o Stochastic Gradient Descent (SGD) e o RMSProp.

A Figura 11 mostra os valores mínimos de erro alcançados pelos otimizadores citados anteriormente. Apesar dos valores de erro mínimo apresentados pelos otimizadores Adam e RMSProp serem bem próximos, o Adam ainda obteve resultados melhores de EQM e EAM, tanto para os erros mínimos quanto levando em conta a média dos resultados de todos os testes realizados, e por isso continuará sendo usado para os testes.

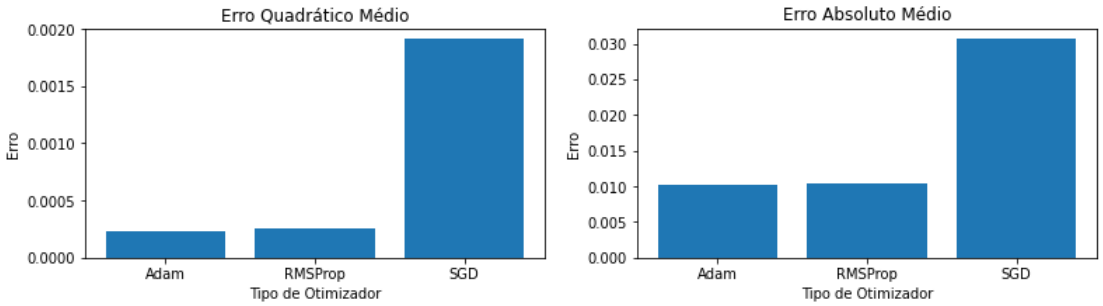


Figura 11 – Valores mínimos de erro obtidos para diferentes otimizadores.

3.4.5 Tamanho do Lote

O tamanho do lote é um termo utilizado em aprendizado de máquina para representar quantos dados de treinamento serão utilizados em uma iteração. Esse parâmetro define o número de amostras a serem apresentadas para o modelo antes do peso ser atualizado, ou seja, para um lote de tamanho 32, as primeiras 32 amostras (0-31) do conjunto de dados de treinamento serão tomadas e treinadas no modelo, em seguida as próximas 32 amostras (32-63) serão usadas para o treinamento, e assim por diante até chegar ao final das amostras.

Pode-se perceber pela Figura 12 que o modelo utilizando treinado com 16 amostras de dados de cada vez teve um melhor desempenho quando comparado aos outros, e por isso esse será o parâmetro utilizado para o treinamento dos próximos modelos.

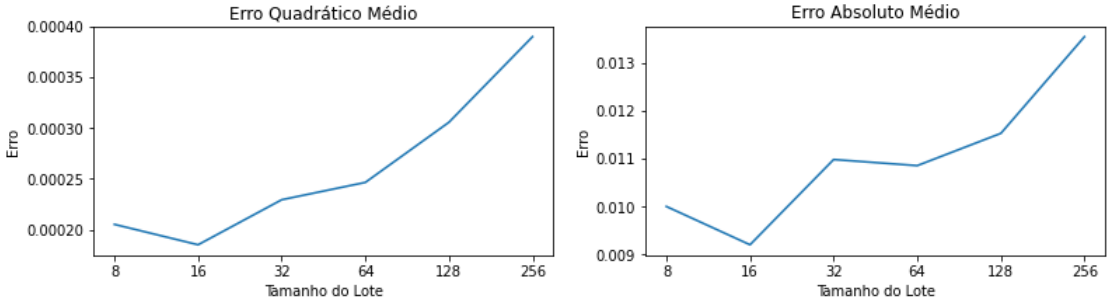


Figura 12 – Valores mínimos de erro obtidos para diferentes tamanhos de lote.

3.4.6 Função de Ativação

Para redes neurais artificiais, a função de ativação é uma função não-linear que é aplicada à saída de um neurônio e ajuda a regular o vetor de peso que será passado aos próximos neurônios na rede. Para redes neurais recorrentes, as funções de ativação mais utilizadas são a sigmoide, a tangente hiperbólica (*tanh*) e a unidade linear retificada (ReLU, do inglês *Rectified Linear Unit*, e serão essas três as funções utilizadas para os testes.

Devido ao erro menor nos dois casos, mostrado na Figura 13, a função de ativação utilizada para o modelo será a tangente hiperbólica.

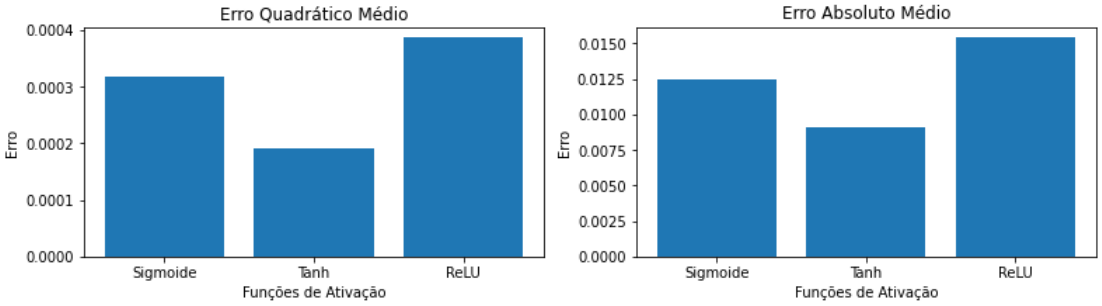


Figura 13 – Valores mínimos de erro obtidos para diferentes funções de ativação.

3.4.7 Número de Épocas

Uma época é uma iteração de quando conjunto de dados de treinamento inteiro é passado pela rede neural uma vez. Portanto, o número é épocas é considerado o número de passagens do conjunto de dados através da rede neural. É um parâmetro importante para se definir, pois quanto maior o número de épocas, mais o modelo se adapta ao conjunto de dados, e por isso deve ser escolhido com cautela. Se o número de épocas for muito pequeno, o modelo pode não se ajustar corretamente ao treinamento, causando valores altos de erros; se o número for muito grande, o modelo pode se ajustar excessivamente ao conjunto, causando sobreajuste.

Observando a Figura 14, temos o menor valor de erro para o modelo treinado com 500 épocas, e para maior número de épocas, esse valor para de diminuir. Apesar de parecer ser o número ótimo de épocas a ser tomado para o modelo, vejamos o erro obtido pelos modelos para a previsão, utilizando o conjunto de dados de teste na Figura 15:

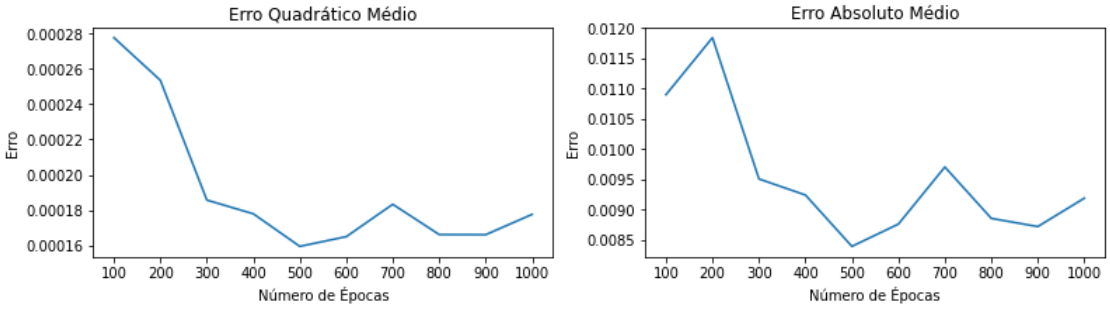


Figura 14 – Valores mínimos de erro obtidos variando o número de épocas.

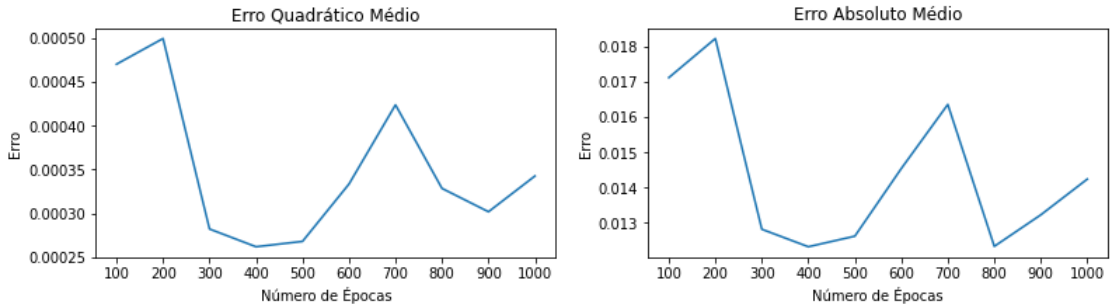


Figura 15 – Valores mínimos de erro obtidos variando o número de épocas para os dados de teste.

Pelo segundo gráfico, vemos que o modelo que têm a melhor predição é na verdade o de 400 épocas, portanto, será aplicado nas etapas seguintes do trabalho.

3.4.8 Tamanho da Janela

O tamanho da janela (em inglês, *window size*, também chamado de *look back period*) é o número de dados que o modelo utilizará do conjunto para fazer a próxima previsão. Por exemplo, no contexto desse trabalho, para uma janela de tamanho 30, o modelo considerará os 30 últimos dias do conjunto para prever o valor da ação no dia seguinte.

Novamente, olhando para a Figura 16, o melhor parâmetro a ser escolhido parece ser a janela de tamanho 20, apesar de o tamanho 2 também possuir erro bastante baixo. Então, vamos novamente observar o erro quando o modelo é aplicado ao conjunto de teste na Figura 17:

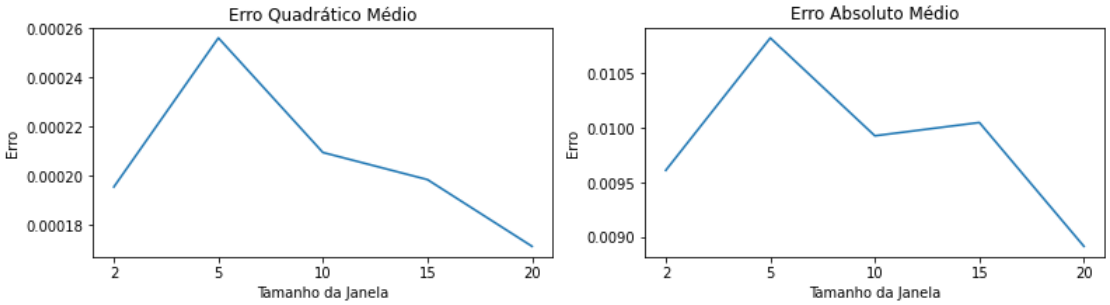


Figura 16 – Valores mínimos de erro obtidos variando o tamanho da janela de aprendizado.

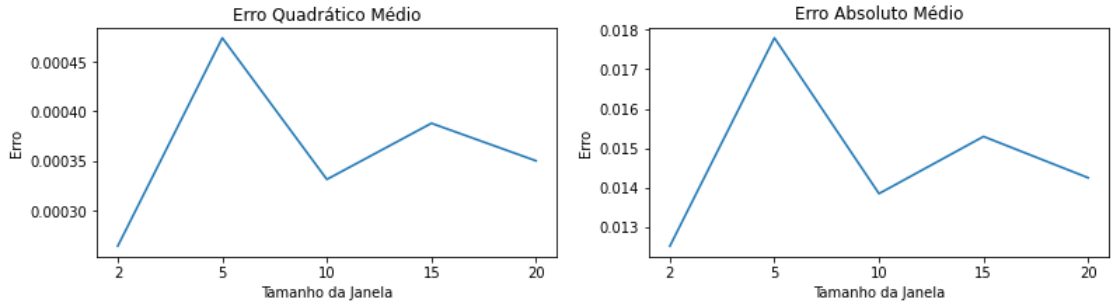


Figura 17 – Valores mínimos de erro obtidos variando o tamanho da janela de aprendizado para os dados de teste.

Portanto, a partir da Figura 17, pode-se perceber que o valor menor de erro para os dados de teste é obtido quando a janela de aprendizado é de tamanho 2, e será o valor utilizado como parâmetro para os modelos de teste.

4 Resultados Obtidos

Como mencionado anteriormente, para obtenção dos resultados serão feitos testes de 4 ações da bolsa de valores: Grupo Fleury (FLRY3), Petrobras (PETR4) e Vale (VALE3); usando 3 modelos de RNR diferentes (VRNN, LSTM e GRU). Os gráficos resultantes dos testes serão apresentados a seguir de acordo com a ação. Os parâmetros utilizados para os modelos foram: 128 neurônios de cada camada intermediária e de entrada, *dropout* de 30%, otimizador Adam, lote de tamanho 16, função de ativação *tanh*, 400 épocas e janela de tamanho 2.

	VRNN	LSTM	GRU
FLRY3	0,000998	0,000261	0,000393
PETR4	0,000948	0,000242	0,000239
VALE3	0,000890	0,000652	0,000599

Tabela 3 – Valores mínimos de EQM obtidos durante os testes para ação e tipo de rede neural

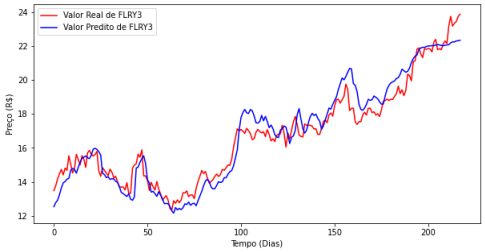
Como pode-se ver na Tabela 3, quando calculado o erro quadrático médio dos modelos, o VRNN tem o erro relativamente maior do que dos outros dois para todas as ações, enquanto o erro do GRU é ainda menor do que o do modelo LSTM, mas a diferença é pequena.

	VRNN	LSTM	GRU
FLRY3	0,02539	0,01253	0,01612
PETR4	0,02626	0,01187	0,01179
VALE3	0,01962	0,01603	0,01473

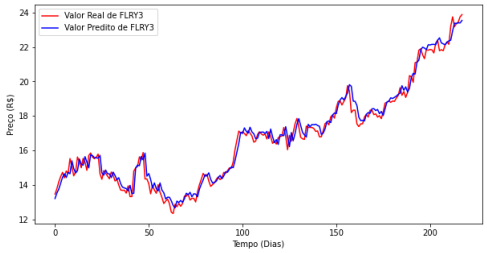
Tabela 4 – Valores mínimos de EAM obtidos durante os testes para ação e tipo de rede neural

Comprovando o que foi visto na Tabela 3, a Tabela 4 mostra também como o modelo de rede neural recorrente mais simples é menos preciso na predição dos que os LSTM e GRU. Esses dois últimos modelos tiveram resultados bem parecidos para as ações PETR4 e VALE3, com o GRU obtendo valores menores de erro nos dois casos, enquanto para FLRY3, o modelo LSTM teve uma melhor predição com uma margem maior do que para outras ações. Isso pode ter acontecido devido ao fato de que os parâmetros foram otimizados usando essa última ação e o modelo LSTM.

No caso da ação FLRY3, a rede neural recorrente mais simples (Figura 18a) conseguiu identificar o padrão relativamente bem no começo, mas a partir da alta que essa ação teve no dia 100, não foi tão bem-sucedida quanto antes. A LSTM (Figura 18b)



(a) Valores reais e preditos da ação FLRY3 utilizando VRNN



(b) Valores reais e preditos da ação FLRY3 utilizando LSTM

Figura 18

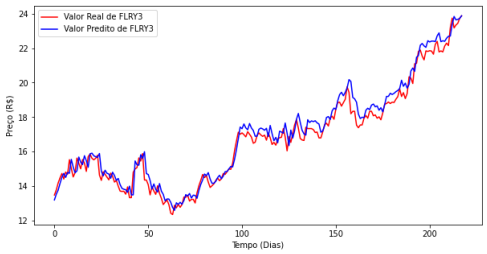
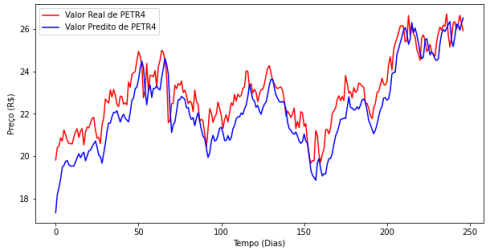
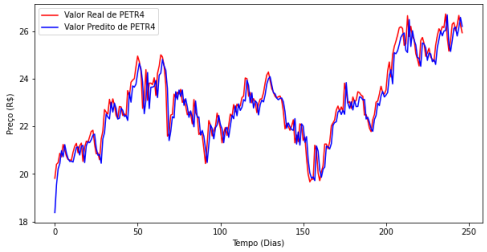


Figura 19 – Valores reais e preditos da ação FLRY3 utilizando GRU

conseguiu identificar bem o padrão e teve bons resultados durante todo o treinamento, e a GRU (Figura 19) no começo teve um desempenho semelhante à LSTM mas, assim como a VRNN, não conseguiu identificar tão bem o padrão depois do dia 100.



(a) Valores reais e preditos da ação PETR4 utilizando VRNN



(b) Valores reais e preditos da ação PETR4 utilizando LSTM

Figura 20

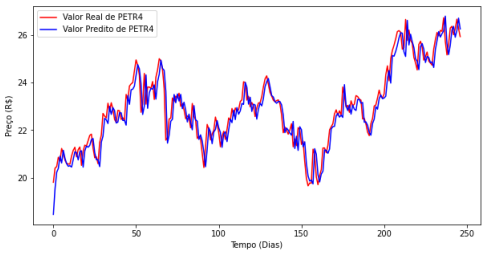
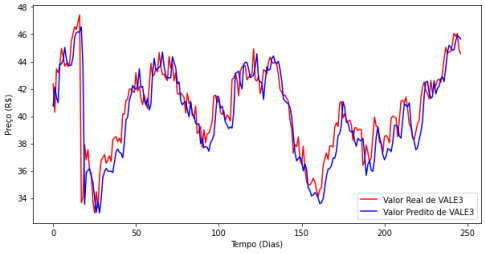


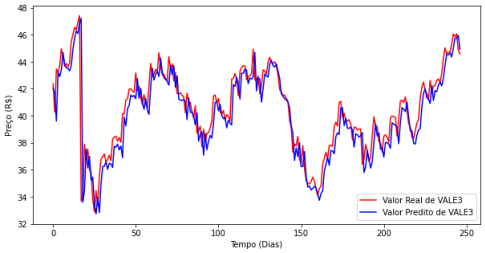
Figura 21 – Valores reais e preditos da ação PETR4 utilizando GRU

Já para a ação PETR4, a VRNN (Figura 20a) consegue identificar bem o padrão, mas tem dificuldade para se ajustar mais precisamente aos valores reais entre os dias 0 e

50 e entre 100 e 200. Os modelos LSTM e GRU (Figura 20b e Figura 21, respectivamente) conseguiram se ajustar muito bem ao modelo, predizendo valores muito próximos aos reais. A GRU levou uma pequena vantagem com erros um pouco menores, o que dá pra ser visto principalmente perto do dia 210.



(a) Valores reais e preditos da ação VALE3 utilizando VRNN



(b) Valores reais e preditos da ação VALE3 utilizando LSTM

Figura 22

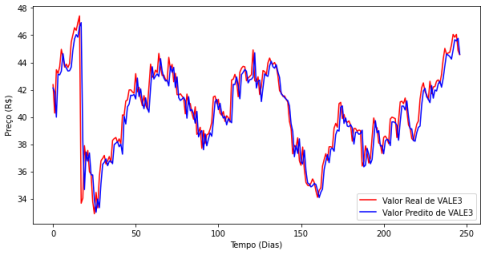


Figura 23 – Valores reais e preditos da ação VALE3 utilizando GRU

Finalmente, para a ação VALE3, o modelo VRNN conseguiu se encaixar melhor ao conjunto de teste, obtendo erros menores do que quando comparado às outras duas ações, mas ainda obtendo desempenho inferior ao do LSTM e GRU (respectivamente, Figura 22b e Figura 23). Como para a ação PETR4, os dois últimos modelos tiveram resultados semelhantes, com pequena vantagem do modelo GRU, que pode ser observada principalmente a partir do dia 150.

5 Trabalhos Futuros

Esse trabalho abordou a predição do valor de fechamento de ações da bolsa de valores com 3 tipos de redes neurais artificiais (LSTM, VRNN e GRU), onde os parâmetros dos modelos finais foram definidos através da otimização da LSTM. Em geral, a LSTM e a GRU tiveram melhor resultado, enquanto a VRNN não conseguiu prever os valores com tanta precisão.

Da forma como os modelos foram otimizados, foram usados os mesmos parâmetros para as três redes neurais recorrentes, o que pode ter beneficiado o resultado da LSTM, que foi utilizada para otimização dos parâmetros. Uma outra abordagem poderia ser otimizar todos os parâmetros para cada rede neural separadamente, o que provavelmente resultaria em parâmetros variados para cada modelo, e comparar novamente os resultados obtidos de cada uma das redes neurais. Nessa abordagem, provavelmente a VRNN e a GRU obteriam resultados mais precisos.

Além disso, outra forma de medir a precisão dos modelos poderia ser utilizada. Uma abordagem recorrente nos artigos encontrados nos últimos anos, principalmente no tema de *algotrading*, utiliza a predição da direção da variação do valor da ação para calcular a precisão. Nesse tipo de abordagem, é calculado a porcentagem do modelo que previu a movimentação correta da ação, ou seja, se o modelo previu que o valor de uma ação ia subir quando ele realmente subiu (sinal de compra da ação), ou abaixar quando ele realmente abaixou (sinal de venda da ação). Alguns trabalhos incluem mais um parâmetro a se analisar, que seria um sinal de *hold*, que é quando o valor da ação nem aumentou nem diminuiu, mas se manteve em uma margem definida, e por isso não deve ser nem comprada nem vendida de acordo com o modelo.

Mais uma forma de medir a precisão, também voltada para artigos focados em *algotrading*, é utilizar o modelo de predição juntamente com alguma estratégia de investimento e calcular o lucro que o modelo traria se a estratégia tivesse sido aplicada no período de teste.

Por fim, esse trabalho teve como objetivo o foco em redes neurais recorrentes, mas essa comparação pode ser feita também para modelos além desses, como para redes neurais convolucionais, perceptrons de múltiplas camadas, máquinas de vetores de suporte, entre várias outras. Como citado anteriormente nesse trabalho, também é válida a junção de múltiplos métodos em um modelo de previsão mais robusto, e também estão sendo desenvolvidas novas técnicas que podem aumentar a precisão dos modelos, como a avaliação de textos retirados da internet de analistas do mercado financeiro, que podem influenciar no preço de ações.

Conclusão

Este trabalho de conclusão de curso abordou o desafio de utilizar técnicas de redes neurais recorrentes para prever o valor de fechamento futuro de ações do mercado financeiro.

Para predizer o valor de três ações diferentes da bolsa de valores do Brasil (FLRY3, PETR4 e VALE3), foram utilizadas abordagens com três tipos redes neurais recorrentes para cada uma: a rede neural recorrente simples, também conhecida como *Vanilla Recurrent Neural Network*, a rede *Long Short-Term Memory* (LSTM), e a rede *Gated Recurrent Unit* (GRU). A acurácia de cada modelo foi medida através do cálculo do Erro Quadrático Médio (EQM) e do Erro Absoluto Médio (EAM).

A primeira abordagem, com o modelo VRNN, conseguiu captar os padrões em grande parte dos dados de teste, mas para nenhuma das ações teve um resultado tão bom quando comparado aos outros modelos. O valor de EQM obtido nos teste ficou em torno de 0,0009, e os valores de EAM ficaram acima de 0,019.

A segunda abordagem proposta, com o modelo LSTM, obteve resultados muito bons, com valores de EQM e EAM perto de 0,00025 e 0,013 respectivamente, sendo que a acurácia foi um pouco menor para a ação VALE3. Foi o melhor modelo para a predição da ação FLRY3, mas teve o segundo melhor desempenho para as outras duas, com uma diferença bem pequena do melhor. Apesar disso, o desempenho melhor para a FLRY3 pode ser resultado da otimização dos parâmetros, pois o modelo e a ação em questão foram utilizados para definir os parâmetros para todos os modelos.

A terceira e última abordagem, com o modelo GRU, obteve o melhor resultado para duas ações (PETR4 e VALE3), e o segundo melhor para a terceira. o EQM desse modelo teve o menor valor de todos os testes (0,000239 com os dados de teste da PETR4), e também o menor valor de EAM (0,0117 para essa mesma ação).

Como o modelo GRU teve o melhor resultado em dois dos três casos, pode-se dizer que foi o modelo que obteve melhor desempenho, mas o LSTM teve um desempenho quase tão bom quanto. Já o modelo VRNN, apesar de conseguir identificar padrões, teve o pior resultado para esse tipo de predição, com diferenças grandes de erro para os outros dois.

Referências

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Citado 2 vezes nas páginas 33 e 34.

ACADEMY, D. S. *Deep Learning Book*. 2019. Disponível em: <http://www.deeplearningbook.com.br/>. Acesso em: 07 de dezembro de 2020. Citado 2 vezes nas páginas 8 e 23.

CHEN, Y.-S.; CHENG, C.-H.; TSAI, W.-L. Modeling fitting-function-based fuzzy time series patterns for evolving stock index forecasting. *Applied Intelligence*, v. 41, p. 327–347, 2014. Citado na página 13.

Cho, K. et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv e-prints*, p. arXiv:1406.1078, jun. 2014. Citado na página 26.

CHOLLET, F. et al. *Keras*. 2015. <https://keras.io>. Citado na página 33.

Exastax. *The Difference Between AI and Machine Learning*. 2017. [Online; accessed 30-December-2020]. Disponível em: <https://medium.com/@exastax/the-difference-between-ai-and-machine-learning-32cfef316372>. Citado 2 vezes nas páginas 8 e 19.

GIRIOLI, L. S.; RIBEIRO, E. M. S. Utilização de redes neurais artificiais para análise técnica no mercado de ações: Estudo dos índices ibovespa e dow jones industrial average (djia). *EPeQ/Fafibe*, v. 1, p. 01–05, 2009. Citado na página 21.

HAYKIN, S. *Neural Networks and Learning Machines*. Third edition. New York: Pearson Education, 2008. Citado na página 24.

HIRANSHA, M. et al. Nse stock market prediction using deep-learning models. *Procedia Computer Science*, v. 132, p. 1351 – 1362, 2018. ISSN 1877-0509. International Conference on Computational Intelligence and Data Science. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050918307828>. Citado 6 vezes nas páginas 8, 20, 24, 25, 28 e 29.

IWASAKI, H.; CHEN, Y. Topic sentiment asset pricing with dnn supervised learning. *Econometric Modeling: Capital Markets - Asset Pricing eJournal*, 2018. Citado 2 vezes nas páginas 14 e 28.

JAMES, G. et al. *An Introduction to Statistical Learning: with Applications in R*. 2018. Disponível em: <https://qubeshub.org/publications/847/1>. Citado na página 32.

KIM, K. Financial time series forecasting using support vector machines. *Neurocomputing*, v. 55, p. 307–319, 2003. Citado na página 13.

KIM, K.-j.; HAN, I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, v. 19, p. 125–132, 2000. Citado na página 13.

Lauren, S.; Harlili, S. D. Stock trend prediction using simple moving average supported by news classification. In: *2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA)*. [S.l.: s.n.], 2014. p. 135–139. Citado na página 33.

LUGER, G. F.; STUBBLEFIELD, W. A. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. [S.l.]: Benjamin/Cummings, 1993. Citado na página 17.

LUSZCZEK, P.; DONGARRA, J. High performance development for high end computing with python language wrapper (plw). *The International Journal of High Performance Computing Applications*, v. 21, n. 3, p. 360–369, 2007. Disponível em: <https://doi.org/10.1177/1094342007078444>. Citado na página 31.

MENDENHALL, W.; REINMUTH, J. E.; BEAVER, R. J. *Statistics for Management and Economics*. [S.l.]: Duxbury Press, 1993. Citado na página 16.

MITCHELL, T. *Machine Learning*. [S.l.]: McGraw Hill, 1997. ISBN 0-07-042807-7. Citado na página 18.

MOGHADDAM, A. H.; MOGHADDAM, M. H.; ESFANDYARI, M. Stock market index prediction using artificial neural network. *Journal of Economics, Finance and Administrative Science*, v. 21, p. 89–93, 2016. Citado na página 19.

PAI, P.-F.; LIN, C.-S. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, v. 33, p. 497–505, 2005. Citado na página 13.

REFENES, A. N.; ZAPRANIS, A.; FRANCIS, G. Stock performance modeling using neural networks: a comparative study with regression models. *Neural Networks*, v. 7, p. 375–388, 1994. Citado na página 13.

RUI, O.; LEE, C. Does trading volume contain information to predict stock returns? evidence from china's stock markets. *Review of Quantitative Finance and Accounting*, v. 14, p. 341–60, 02 2000. Citado na página 32.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Prentice Hall, 2010. ISBN 978-0-13-604259-4. Citado na página 17.

Ryll, L.; Seidens, S. Evaluating the Performance of Machine Learning Algorithms in Financial Market Forecasting: A Comprehensive Survey. *arXiv e-prints*, p. arXiv:1906.07786, jun. 2019. Citado na página 13.

SAUD, A. S.; SHAKYA, S. Analysis of look back period for stock price prediction with rnn variants: A case study on banking sector of nepse. *Procedia Computer Science*, v. 167, p. 788–798, 2020. Disponível em: <https://doi.org/10.1016/j.procs.2020.03.419>. Citado 3 vezes nas páginas 28, 30 e 33.

SHUMWAY, R. H.; STOFFER, D. S. *Time Series Analysis and Its Applications*. Fourth edition. [S.l.]: Springer, 2017. Citado na página 17.

SILVA, E.; OLIVEIRA, A. C. de. *Dicas para a Configuração de Redes Neurais*. 2001. Rio de Janeiro. Citado na página 22.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>. Citado na página 35.

STATHAKIS, D. How many hidden layers and nodes? *International Journal of Remote Sensing*, Taylor Francis, v. 30, n. 8, p. 2133–2147, 2009. Disponível em: <https://doi.org/10.1080/01431160802549278>. Citado na página 34.

TURING, A. Computing machinery and intelligence. *Mind*, v. 59, p. 433–460, 1950. Citado na página 17.

VELLASCO, M. M. B. R. *Redes Neurais Artificiais*. 2007. [Rio de Janeiro: PUC, Apostila]. Citado na página 22.

WANG, Q.; XU, W.; ZHENG, H. Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles. *Neurocomputing*, v. 299, p. 51–61, 2018. Disponível em: <https://doi.org/10.1016/j.neucom.2018.02.095>. Citado 2 vezes nas páginas 14 e 28.

WEISS, G.; GOLDBERG, Y.; YAHAV, E. On the practical computational power of finite precision rnns for language recognition. *56th Annual Meeting of the Association for Computational Linguistics*, p. 740–745, 2018. Citado na página 26.

Wikipedia contributors. *Recurrent neural network — Wikipedia, The Free Encyclopedia*. 2020. [Online; accessed 7-December-2020]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Recurrent_neural_network&oldid=991832590>. Citado 3 vezes nas páginas 8, 25 e 26.

YAHOO! Finance. 2020. Disponível em: <https://https://finance.yahoo.com/>. Acesso em: 07 de dezembro de 2020. Citado 2 vezes nas páginas 9 e 31.

Zaremba, W.; Sutskever, I.; Vinyals, O. Recurrent Neural Network Regularization. *arXiv e-prints*, p. arXiv:1409.2329, set. 2014. Citado na página 24.

ZHANG, G.; PATUWO, B. E.; HU, M. Y. Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, v. 14, p. 35–62, 1998. Citado na página 19.

Índice

Desenvolvimento, 31
- Coleta de Dados, 31
- Contrução dos Modelos, 33
- Definição dos Hiperparâmetros, 34
  - *Dropout*, 35
  - Camadas das Redes Neurais, 34
  - Função de Ativação, 37
  - Número de Neurônios de Cada Camada, 34
  - Número de Épocas, 37
  - Otimizador, 35
  - Tamanho da Janela, 38
  - Tamanho do Lote, 36
- Tratamento de Dados, 32

Fundamentação teórica, 16
- Aprendizado de Máquina, 18
- Erro Absoluto Médio, 27
- Erro Quadrático Médio, 27
- Inteligência Artificial, 17
- Medidas de Avaliação, 26
- Rede *Gated Recurrent Unit*, 26
- Rede *Long Short-Term Memory*, 25
- Redes Neurais Artificiais, 19
- Redes Neurais Recorrentes, 23
- Séries Temporais, 16

Resultados Obtidos, 40

Trabalhos Futuros, 43

Trabalhos Relacionados, 28
- Estudo de comparação de predições utilizando diferentes métodos de *Deep Learning*, 29
- Estudo de predição do valor de ações utilizando variantes de RNR, 30