

**FACULDADE DE TECNOLOGIA DO IPIRANGA**

**CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

MARIO THOMAZ APRA

APLICAÇÃO DE REDES NEURAIS PARA PREVISÃO DE PREÇO DE FECHAMENTO DAS AÇÕES NO MERCADO FINANCEIRO

SÃO PAULO

2017



**FACULDADE DE TECNOLOGIA DO IPIRANGA**

**CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

MARIO THOMAZ APRA

APLICAÇÃO DE REDES NEURAIS PARA PREVISÃO DE PREÇO DE FECHAMENTO DAS AÇÕES NO MERCADO FINANCEIRO

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia do Ipiranga, como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientadora: Prof.ª Dr. ª Andréia Cristina G. Machion.

SÃO PAULO

2017

AGRADECIMENTOS

A todos que colaboraram...

*“É muito melhor ousar feitos poderosos, ousar ganhar gloriosos triunfos, mesmo que se marque pelo fracasso, do que ser classificado com os pobres de espírito que não aproveitam nem experimentam muito, porque vivem na penumbra cinzenta que desconhece a vitória e a derrota“ (Tradução do autor).*

- *Theodore Roosevelt*

RESUMO

Uma rede neural que é capaz de prever os preços das ações pode ser muito útil para qualquer pessoa que queira investir em ações. Hoje em dia é necessário muito trabalho para extrair os dados da BM&FBOVESPA, normalizá-los, criar uma rede, configurá-la, treiná-la e verificar se ela é eficiente. Assim, este trabalho tem como objetivo criar e validar redes neurais aplicadas à predição de preços no mercado financeiro. Além disso, construir um software que facilite todas essas etapas para que até uma pessoa sem conhecimento em programação possa criar uma rede neural eficiente. Caracterizando-se como uma pesquisa qualitativa, documental com o método indutivo. O resultado encontrado foi ...

Palavras-chave: Inteligência Artificial, Rede Neural Artificial, predição, mercado financeiro.

ABSTRACT

A neural network that can predict stock prices can be very useful for anyone who wants to invest in stock market. Nowadays is necessary a lot of work to extract the data from BM&FBOVESPA, normalize them, build a network, configure it, train it and check if it is efficient. That way, this work has the propose build and validate neural network to predict stock market prices. Besides that, build a software that facilitates all these steps so that even a person without programming knowledge can create an efficient neural network. Characterizing as qualitative research, documentary research with the inductive method. The founded result was....

Keywords: Artificial Intelligence, Artificial Neural Network, prediction, stock market.

LISTA DE FIGURAS

[Figura 1. Modelo de um Neurônio Artificial 19](#_Toc475376262)

[Figura 2. Funções de ativação 20](#_Toc475376263)

[Figura 3. ActivationLOG 21](#_Toc475376264)

[Figura 4. ActivationSigmoid 22](#_Toc475376265)

[Figura 5. ActivationTANH. 23](#_Toc475376266)

[Figura 6. Comparação entre Elliott (contínua) e Sigmoid (pontilhada) 23](#_Toc475376267)

[Figura 7. Comparação Symmetric Elliott (contínua) e TANH (pontilhada). 24](#_Toc475376268)

[Figura 8. Multilayer perceptron (MLP) 25](#_Toc475376269)

[Figura 9. TLFN focada. 26](#_Toc475376270)

[Figura 10. Recurrent Network 27](#_Toc475376271)

[Figura 11. Custo de entendimento 27](#_Toc475376272)

[Figura 12. Hierarquia do aprendizado 28](#_Toc475376273)

[Figura 13. Redes com e sem dropout 29](#_Toc475376274)

[Figura 14. Overfitting com e sem dropout 29](#_Toc475376275)

[Figura 15. Maiores altas em 5 dias 33](#_Toc475376276)

[Figura 16. Maiores quedas em 5 dias 33](#_Toc475376277)

[Figura 17. Product Backlog 40](#_Toc475376278)

[Figura 18. Sprint Backlog 1 41](#_Toc475376279)

[Figura 19. Sprint Backlog 2 41](#_Toc475376280)

[Figura 20. Sprint Backlog 3 41](#_Toc475376281)

[Figura 21. Atores 42](#_Toc475376282)

[Figura 22. I001 – Gerenciar Rede Neural 44](#_Toc475376283)

[Figura 23. I002 - Treinar Rede Neural 46](#_Toc475376284)

[Figura 24. I003 - Gerar Relatório 47](#_Toc475376285)

LISTA DE TABELAS

[Tabela 1. Comparativo Elliott 24](#_Toc475376286)

[Tabela 2. Preços PRIO3 “puros” 32](#_Toc475376287)

[Tabela 3. PRIO3 normalizado 32](#_Toc475376288)

[Tabela 4. Comparativo softwares 36](#_Toc475376289)

[Tabela 5. Notas softwares 36](#_Toc475376290)

LISTA DE EQUAÇÕES

[Equação 1. Neurônio Artificial 20](#_Toc475376291)

[Equação 2. ActivationLOG 21](#_Toc475376292)

[Equação 3. ActivationSigmoid 22](#_Toc475376293)

[Equação 4. ActivationTANH 22](#_Toc475376294)

[Equação 5. Elliott 23](#_Toc475376295)

[Equação 6. Symmetric Elliott 23](#_Toc475376296)

[Equação 7. Normalização 31](#_Toc475376297)

[Equação 8. Normalização com margem 31](#_Toc475376298)

[Equação 9. Normalização exemplo 32](#_Toc475376299)

LISTA DE ABREVIATUAS E SIGLAS

RNA - Rede Neural Artificial

LOG - Logaritmo

TANH - Tangente Hiperbólica

MLP - *Multlayer Perceptron*

TLFN - *Time Lagged Feed-forward Network*

IDE – *Integrated Development Environment*

API - *Application Programming Interface*

AI – *Artificial Intelligence*

IBM - *International Business Machines*

SUMÁRIO

[1 INTRODUÇÃO 16](#_Toc475376300)

[1.1 Objetivos 17](#_Toc475376301)

[1.2 Justificativa 17](#_Toc475376302)

[1.3 Métodos e Tecnologias 17](#_Toc475376303)

[1.4 Organização do Trabalho 18](#_Toc475376304)

[2 REDES NEURAIS ARTIFICIAIS 19](#_Toc475376305)

[2.1 Motivação para utilizar Redes Neurais na Predição. 20](#_Toc475376306)

[2.2 Funções de Ativação 20](#_Toc475376307)

[2.2.1. ActivationBiPolar 21](#_Toc475376308)

[2.2.2. Activation Competitive 21](#_Toc475376309)

[2.2.3. ActivationLinear 21](#_Toc475376310)

[2.2.4. ActivationLOG 21](#_Toc475376311)

[2.2.5. ActivationSigmoid 22](#_Toc475376312)

[2.2.6. ActivationSoftMax 22](#_Toc475376313)

[2.2.7. ActivationTANH 22](#_Toc475376314)

[2.2.8. Elliott 23](#_Toc475376315)

[2.2.9. Symmetric Elliott 23](#_Toc475376316)

[2.3 Topologias da Rede Neural 24](#_Toc475376317)

[2.3.1 Feed-Forward 24](#_Toc475376318)

[2.3.2 Recurrent Network 26](#_Toc475376319)

[2.4 Aprendizagem 27](#_Toc475376320)

[2.4.1. Backpropagation 30](#_Toc475376321)

[2.4.2. ResilientPropagation 30](#_Toc475376322)

[2.4.3. ManhattanPropagation 30](#_Toc475376323)

[2.4.4. QuickPropagation 30](#_Toc475376324)

[2.4.5. ScaledConjugateGradient 31](#_Toc475376325)

[2.5 Normalização 31](#_Toc475376326)

[3 Benchmark 34](#_Toc475376327)

[3.1 Excel + VBA 34](#_Toc475376328)

[3.2 Weka 34](#_Toc475376329)

[3.3 Neuroph Studio 35](#_Toc475376330)

[3.4 Matlab 35](#_Toc475376331)

[3.5 NeuroFURG 35](#_Toc475376332)

[3.6 Comparativo 36](#_Toc475376333)

[4 REQUISITOS DO SISTEMA DE SOFTWARE 37](#_Toc475376334)

[4.1 Identificação dos requisitos 37](#_Toc475376335)

[4.1.1 Prioridades dos requisitos 37](#_Toc475376336)

[4.1.2 Requisitos Funcionais 38](#_Toc475376337)

[4.1.3 Requisitos Não-Funcionais 39](#_Toc475376338)

[4.1.4 Regras de Negócio 40](#_Toc475376339)

[4.1.5 Product e Sprint Backlog 40](#_Toc475376340)

[4.2 Modelagem dos requisitos funcionais 42](#_Toc475376341)

[4.2.1 Atores 42](#_Toc475376342)

[4.2.2 Diagrama de Caso de uso 42](#_Toc475376343)

[4.2.3 Especificação do Caso de Uso 43](#_Toc475376344)

[5 Bibliografia 49](#_Toc475376345)

# INTRODUÇÃO

As ações são títulos representativos do capital social de uma companhia aberta, que é negociada em bolsa de valores. Seu preço é fruto das condições de oferta e demanda que refletem a economia do País, específicas da empresa e de seu setor econômico. (FORTUNA, 2008).

Tentar prever os preços das ações na bolsa de valores é algo que todo mundo que conhece um pouco deste mundo já sonhou algum dia. O segundo estágio para escolher um investimento começa com crenças relevantes sobre o desempenho futuro, tais como possíveis descontos ou retornos sobre os investimentos. Um exemplo é vender uma ação a um preço maior a qual foi pago por ela e, neste caso, nota-se a importância de prever os preços futuros (MARKOWITZ, 1952).

O estudo de Inteligência Artificial (IA) tem em uma de suas razões o entendimento do ser humano, mas não como filosofia ou psicologia, que também se concentra na inteligência, mas em como construir entidades inteligentes. Outra razão é a importância desta construção por si só. É notório que ninguém consegue prever o futuro em detalhes, mas um computador com a capacidade da inteligência humana pode ter um grande impacto no dia-a-dia e no futuro da civilização. (RUSSELL e NORVIG, 2005)

A empresa que trabalha com IA voltada para o mercado que mais chamou atenção foi a Numerai, que é uma empresa criada no final de 2015 como sendo um torneio para predição do mercado financeiro para cientistas de dados e desde lá, já recebeu 14 bilhões de preços previstos. O mais interessante é que eles pegam os modelos que possuem maior acurácia e usam em seu fundo de investimento real e em troca, remunera os melhores contribuintes com Bitcoin e, além disso, é gratuito e livre para qualquer pessoa submeter sua rede para eles. (NUMERAI, 2016)

Voltando um pouco para a origem da AI, um questionamento muito importante no começo da sua evolução é: um computador pode pensar? Essa foi uma pergunta com a qual Alan Turing (1950), “avô” da IA, começou sua publicação. Ele faz uma comparação com a objeção de Lady Lovelace, que diz que uma máquina só pode fazer o que dizem para ela fazer. Esta pergunta continua sendo atual, é muito difícil responder se um computador pode pensar ou não.

Para tentar responder essa pergunta, uma motivação de se trabalhar com redes neurais é a diferença que um cérebro humano processa uma informação em relação a um computador convencional. O cérebro é altamente complexo, não linear e processa as informações paralelamente e muito rapidamente. Há alguns casos em que o cérebro humano é muito superior, em capacidade de processamento e entendimento, do que um computador, como é o caso da visão, ou o reconhecimento de um de um rosto familiar dentre vários rostos desconhecidos em aproximadamente 150 ms (HAYKIN, 1998).

Por outro lado, uma das conquistas mais simbólicas e famosas do uso da aplicação da AI ocorreu em maio de 1997, na qual *Deep Blue*, computador da IBM, conquistou um feito inédito que foi ganhar de um dos melhores jogadores de xadrez da história, Garry Kasparov, com um placar de duas vitórias, uma derrota e três empates para a máquina. Este feito foi importante não só para vencer uma partida de xadrez, mas a arquitetura usada na construção desta brilhante máquina foi aplicada na análise de risco para o mercado financeiro, na dinâmica molecular para descobrir e desenvolver novos remédios, no *data mining* e, principalmente, para desenvolver e entender os limites da programação e arquitetura paralela de uma forma massiva (IBM, 2011).

Não é de hoje que alguns fundos de investimentos usam de alguma forma o poder da computação para auxiliar a atuar no mercado financeiro. Esse poder vai desde o uso *algorithmic trading* para aumentar a liquidez de um papel, como no trabalho do Hendershott, Jones e Menkveld (2011) ou para operar em mercados, como no do Chabou, Chiquioine, *et al* (2014) até o uso da AI para tomar as decisões, inclusive passando por momentos difíceis nessa área, como o recente *Brexit* (Saída do Reino Unido da União Europeia),e conseguindo ótimos retornos, como no caso da Kathleen e Komaki (2016).

Vários trabalhos acadêmicos foram realizados demonstrando a eficácia do uso de AI e redes neurais para predição dos preços das ações, um dos primeiros feito por Kimoto, Asakawa, *et al.* (1990), ou realizado por Thomaz e Vellasco (2005), ou por Refenes, Zapranis e Francis (1994), entre outros. Os trabalhos elencados não abordavam a criação de uma rede em si, somente sua aplicação. Desta forma, essa pesquisa tem como diferencial o desenvolvimento de um software.

## 1.1 Objetivos

O objetivo deste trabalho é criar e validar redes neurais aplicadas à predição de preços no mercado financeiro.

## 1.2 Justificativa

Ao tentar reproduzir trabalhos já feitos na área de redes neurais aplicadas ao mercado financeiro, perde-se muito tempo com toda a parte *background* e necessita-se de um bom conhecimento em programação para criar uma rede.

Em todos os trabalhos pesquisados, as etapas de extração dos dados, normalização, configuração, treinamento e validação a rede e, por fim, gerar um relatório com os resultados, não era o objetivo e era uma etapa que todos os pesquisadores perdiam certo tempo. Nos trabalhos, várias ferramentas foram utilizadas, como o Excel e o Neuroph Studio no trabalho do Krieger (2012), Matlab e Eviews no trabalho do Oliveira (2007), entre outros.

## 1.3 Métodos e Tecnologias

O trabalho é apresentado por meio de levantamentos bibliográficos, tipo de pesquisa documental, qualitativa e com o método indutivo. O software está escrito na linguagem Java 8, com a biblioteca YahooFinance API para extração dos dados e Encog Framework para a criação da rede neural. O versionamento do sistema é feito pelo GitHub com o ambiente de desenvolvimento (IDE) Eclipse.

## 1.4 Organização do Trabalho

Este trabalho é composto por 7 capítulos, incluindo esta introdução. O Capítulo 2 introduz os conceitos gerais de redes neurais. O Capítulo 3 apresenta algumas ferramentas para a criação de redes neurais, analisando suas contribuições e o que deixa a desejar para esta tarefa específica. O Capítulo 4 apresenta os requisitos do software desenvolvido. O Capítulo 5 apresenta o desenvolvimento do projeto. O Capítulo 6 apresenta os resultados obtidos. O Capítulo 7 apresenta as considerações finais.

# REDES NEURAIS ARTIFICIAIS

Rede Neural pode ser definida como um processador massivamente distribuído e paralelo, feito de unidades de processamentos simples que tem uma propensão natural em armazenar conhecimentos empíricos e tornar isso disponível para o uso. Lembra um cérebro em dois aspectos (HAYKIN, 2009):

1. O conhecimento é adquirido pela rede a partir do ambiente em que se encontra durante um processo de aprendizagem
2. Os pesos sinápticos, ou forças de interconexão, são usados para armazenar os conhecimentos obtidos.

Assim, podemos dizer que uma Rede Neural Artificial é uma combinação de neurônios artificiais. Para entender uma RNA, é imprescindível saber como funciona um neurônio. Segundo (TAFNER, 1998, p. 1):

“O neurônio artificial é uma estrutura lógico-matemática que procura simular a forma, o comportamento e as funções de um neurônio biológico. Assim sendo, os dendritos foram substituídos por *entradas*, cujas ligações com o corpo celular artificial são realizadas através de elementos chamados de *peso* (simulando as sinapses). Os estímulos captados pelas entradas são processados pela *função de soma*, e o limiar de disparo do neurônio biológico foi substituído pela *função de transferência*.”

A Figura 1 ilustra uma representação de um neurônio artificial:



Figura . Modelo de um Neurônio Artificial

Fonte: (VON ZUBEN e ATTUX, 2010).

A Equação 1 demonstra a fórmula de um neurônio artificial com o bias (VON ZUBEN e ATTUX, 2010):



Equação 1. Neurônio Artificial

Onde *Yk*é a saída do neurônio, *f* representa a função de ativação, dentro dos parênteses um somatório dos sinais de entrada (xj) com os pesos sinápticos (wkj) com o bias (bk).

## 2.1 Motivação para utilizar Redes Neurais na Predição.

O principal benefício do uso de uma RNA (Rede Neural Artificial) é a capacidade de generalização, que é a capacidade de uma rede aprender, isto é, produzir uma saída adequada mesmo para entradas não encontradas durante o treinamento (HAYKIN, 2009). Um exemplo prático: uma alteração grande no volume financeiro negociado geralmente indica um aumento na volatilidade das ações.

Outro ponto importante é a adaptabilidade, que é a capacidade de uma RNA alterar seus pesos sinápticos, devido a modificações no meio ambiente, em tempo real (HAYKIN, 2009). No contexto complexo do mercado financeiro, onde diversos fatores podem afetar o preço de uma ação, como fatores psicológicos, notícias, divulgação de resultados, regulamentação, entre outros, esta capacidade é essencial.

## 2.2 Funções de Ativação

A função de ativação restringe o valor recebido pela Função Soma, limitando a amplitude da saída de um neurônio em um intervalo de [0,1] ou [-1,1] (HAYKIN, 2009). Ela tem a função de manter o neurônio ativa quando as entradas fornecidas forem corretas e mantê-lo inativo quando forem incorretas (RUSSELL e NORVIG, 2005).

Existem vários tipos de funções de ativação, dentre elas a função linear (a), função rampa (b), função *step* (c) e função sigmoide (d). Esta última é a mais utilizada e é uma mistura entre as funções lineares e não lineares (HAYKIN, 2009), que serão exemplificadas na Figura 2.



Figura . Funções de ativação

Fonte: (DO CARMO ROQUE e DE MELLO, 2009)

A seguir é apresentado algumas das funções de ativação que estão implementadas no software. As funções que possuírem derivada pode ser utilizado com a aprendizagem do tipo *propagation*. (HEATON, 2011).

### 2.2.1. ActivationBiPolar

Utilizada em redes que requerem valores binários (*true* ou *false*). A ideia básica é que se o valor for positivo o resultado é 1, senão é -1 (HEATON, 2011). Não possui derivada. Está ilustrada na Figura 2 (c).

Este tipo de função de ativação foi utilizado primeiramente por McCulloch e Pits (1943), onde a saída do neurônio assumia o valor de 1 ou 0 (no lugar do -1). Em seu trabalho foi referenciado com o termo de *all-or-none property* (propriedade de tudo ou nada).

### 2.2.2. Activation Competitive

É utilizada para forçar um seleto grupo de neurônios para ganhar. O vencedor é o grupo com a maior saída. Para isso, percorre todos os parâmetros e acha os vencedores, e atribui aos perdedores o valor zero. Não possui derivada.

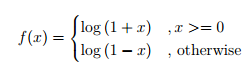
Este tipo de função de ativação pode ser usado tanto para redes competitivas quanto para mapas auto-organizáveis (HEATON, 2011).

### 2.2.3. ActivationLinear

É utilizada em tipos específicos de redes, que não possuem função de ativação, como por exemplo, mapas auto-organizáveis. Este tipo está representado na Figura 2 (a), na qual o valor que é passado para x é igual a f(x). Por essa razão, na primeira (entrada) e na última (saída), será sempre utilizado esse tipo de função.

Pode resultar em números negativos. Como sua derivada é constante, também pode ser utilizada no treinamento do tipo *propagation* (HEATON, 2011).

### 2.2.4. ActivationLOG



Equação . ActivationLOG



Figura . ActivationLOG

Fonte: (HEATON, 2011)

Possui uma curva similar com a tangente hiperbólica, que será apresentada posteriormente. Este tipo de função de ativação é útil para prevenir saturação, portanto é uma escolha possível quando o treinamento não obtém sucesso com a tangente hiperbólica.

Ela pode fazer com que a etapa de treinamento demore significativamente. Pode resultar em números negativos e possui derivada. (HEATON, 2011)

### 2.2.5. ActivationSigmoid



Equação . ActivationSigmoid

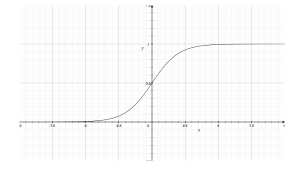


Figura . ActivationSigmoid

Fonte: (HEATON, 2011)

Só produz números positivos. É uma escolha bem comum para redes *feed-forward* e algumas redes recorrentes simples. Se requerido números negativos, a tangente hiperbólica pode ser uma solução melhor. Possui derivada (HEATON, 2011).

### 2.2.6. ActivationSoftMax

Só vai produzir saídas iguais a uma. Geralmente é utilizada nas camadas de saída para problemas que envolvem classificação, mas também pode ser usada nas camadas escondidas. (HEATON, 2011)

### 2.2.7. ActivationTANH



Equação . ActivationTANH

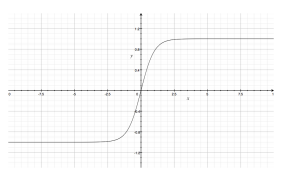


Figura . ActivationTANH.

Fonte: (HEATON, 2011)

Utiliza funções de tangente hiperbólica. É, provavelmente, a função mais comum, por trabalhar com valores positivos e negativos. Possui derivada (HEATON, 2011).

### 2.2.8. Elliott



Equação . Elliott

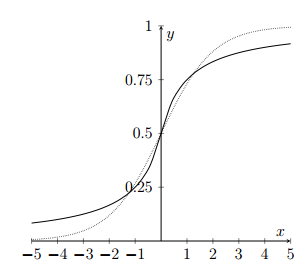


Figura . Comparação entre Elliott (contínua) e Sigmoid (pontilhada)

Fonte: (WINRICH, 2015)

Foi originalmente proposta por David L. Elliott em 1993 como uma alternativa computacionalmente mais eficiente comparada a sigmoide e a tangente hiperbólica. Também produz valores entre 0 e 1. Possui derivada (WINRICH, 2015).

### 2.2.9. Symmetric Elliott



Equação . Symmetric Elliott

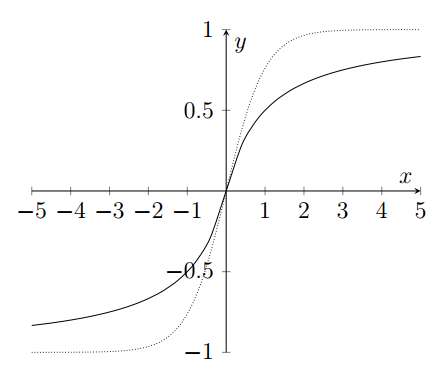


Figura . Comparação Symmetric Elliott (contínua) e TANH (pontilhada).

Fonte: (WINRICH, 2015)

A ideia é a mesma da Elliott, mas para substituir a TANH, também produz valor entre -1 e 1 e possui derivada.

Conforme a Tabela 1 apesar de requerer mais iterações, o tempo total de treinamento é menos do que a metade na média dos treinos se comparada a TANH.

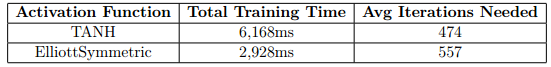


Tabela . Comparativo Elliott

Fonte: (WINRICH, 2015)

## 2.3 Topologias da Rede Neural

Existem dois tipos de topologias para uma RNA: *feed-forward* (alinhadas à frente) e *recurrent network* (redes recorrentes) (RUSSELL e NORVIG, 2005).

### 2.3.1 Feed-Forward

As redes feed-forward podem conter uma (*single-layer*) ou mais camadas (*multilayer*). Estão presentes uma camada de entrada e uma de saída (nesta ordem), e as demais camadas que estão entre elas, também chamadas de ocultas. Adicionando essas camadas ocultas, a rede tem a possibilidade de extrair estatísticas mais robustas da camada de entrada. (HAYKIN, 2009)



Figura . Multilayer perceptron (MLP)

Fonte: (HAYKIN, 2009)

De acordo com a Figura 8, percebe-se que os dados têm um fluxo que vai da camada de entrada, passando pelas camadas escondidas (se tiver) e por fim vai para a camada de saída. Além disso, após passar por uma camada, o dado nunca volta para uma camada anterior nem para a mesma.

#### Incorporação do Tempo em Feed-Forward

Dentre algumas aplicações que necessitam a incorporação do tempo, destaca-se a "Previsão e modelagem de séries temporais. Para que uma rede tenha este dinamismo, é necessário que possua memória, podendo ser de curto ou longo prazo. Para fazer isso, uma das formas possíveis é adicionar atrasos de tempo, que pode ser adicionada nos pesos sinápticos ou na entrada da rede (HAYKIN, 2009).

Um exemplo para este caso, que é abordado no trabalho, é a TLFN focada (*focused time lagged feed-forward network*), onde os atrasos de tempo são incorporados somente na camada de entrada.



Figura . TLFN focada.

Fonte: (HAYKIN, 1998)

A Figura 9 ilustra uma TLFN focada, onde as entradas são apresentadas à camada de entrada neurônio por neurônio, ou seja, com um *delay.* No instante *n* é apresentado ao primeiro neurônio, no n*-1* no segundo neurônio e assim por diante, até atingir o número *n-p* de atraso.

### 2.3.2 Recurrent Network

Outra forma de dar dinamismo e incorporar o fator tempo a uma rede é com o uso de uma rede recorrente, que é o caso deste trabalho. A diferença entre a rede recorrente e a *feed-forward* é que ao menos um neurônio é alimentado por outro que se encontra na mesma camada ou em uma camada posterior. Este mecanismo gera um ciclo de *feedback*, que tem um profundo impacto na capacidade de aprendizagem da rede e em sua performance (HAYKIN, 2009).

Um exemplo de uso para redes recorrentes é com a previsão de séries temporais, nas quais o resultado de um período é fonte de dado para outro período (HAYKIN, 2009).

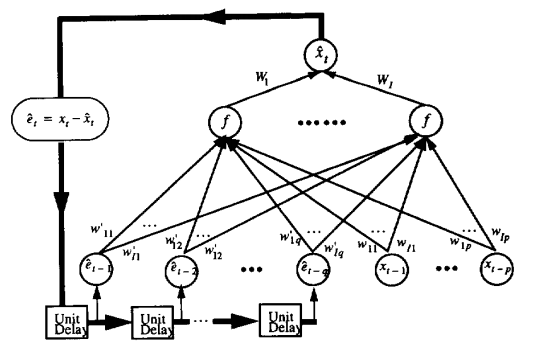


Figura . Recurrent Network

Fonte: (CONNOR, MARTIN e ATLAS, 1994).

A Figura 10 ilustra uma rede recorrente, e nota-se a diferença dita anteriormente: no fluxo de dados tem uma parte cíclica.

Não existem regras para a escolha da topologia da rede, somente indicações. Um bom método para a escolha é por meio da "tentativa e erro", baseando-se em trabalhos anteriores (MANTOVANI, 2011).

## 2.4 Aprendizagem

A aprendizagem consiste no ajuste dos pesos das sinapses para obter as saídas desejadas (HEATON, 2011). Ela é um dos fatores mais importantes na boa formação da RNA. O gargalo para a ampla utilização dos sistemas inteligentes é a aquisição de conhecimento, que ainda é artesanal e subjetiva. (MONARD e BARANAUSKAS, 2003). A Figura 11 mostra o custo de aprendizado para vários tipos de dados diferentes:

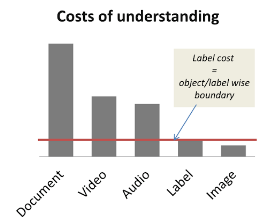


Figura . Custo de entendimento

Fonte: (CHERMAN, TSOUMAKAS e MONARD, 2016).



Figura . Hierarquia do aprendizado

Fonte: (MONARD e BARANAUSKAS, 2003).

Monard e Baranauskas (2003) explicam muito bem a Figura 12, suas tipologias e vários paradigmas de aprendizado. Neste trabalho será explicado somente o aprendizado supervisionado, que será utilizado.

“Um ponto importante a ser considerado é a escolha de atributos com boa capacidade preditiva. Não importa qual método seja empregado, os conceitos que podem ser aprendidos estão à mercê dos dados e da qualidade dos atributos. Por exemplo, para a tarefa de determinar se uma pessoa está ou não com gripe, pode-se escolher atributos com baixo poder preditivo, tais como (cor-do-cabelo, cor-do-olho, modelo-do-carro, número-de-filhos) ou atributos com alto poder preditivo, tais como (temperatura, resistência-da-pele, exame do-pulmão). Para esta tarefa específica, no segundo caso, melhores previsões em exemplos não-rotulados provavelmente ocorrerão do que com o primeiro conjunto de atributos.” (MONARD e BARANAUSKAS, 2003, p.43).

Usando os pontos levantados pela citação anterior e conversas com especialistas no mercado financeiro, foi decidido deixar disponível os seguintes dados para formar a entrada: preço de abertura, preço máximo do dia, preço mínimo do dia, preço de fechamento (cotação) e volume.



Figura . Redes com e sem *dropout*

Fonte: (SRIVASTAVA, HINTON, *et al.*, 2014)

Além disso, existe mais uma entrada muito importante: o *dropout rate*, que é utilizado para evitar que a rede decore ao invés de aprender, conhecido como *overfitting*. Com o uso do *dropout*, alguns neurônios são desativados, como se fossem removidos temporariamente (daí o nome *drop*), como se pode ver na Figura 13.

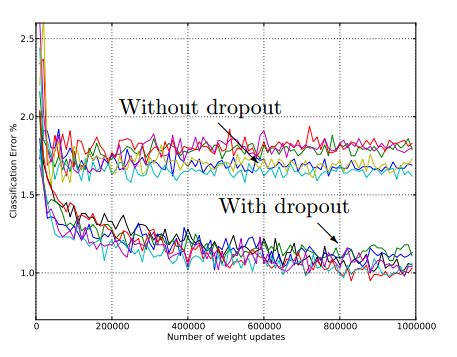


Figura . *Overfitting* com e sem *dropout*

Fonte: (SRIVASTAVA, HINTON, *et al.*, 2014)

A mostra um gráfico comparando a diferença entre uma rede que faz o uso do *dropout* e outra sem, em diferentes arquiteturas computacionais. Dá para notar que a rede fica com *overfitting* antes das primeiras 100.000 nas primeiras atualizações dos números dos neurônios, já na rede que usa o *dropout* ela perde a eficiência na casa dos 800.000.

A seguir é apresentado os principais algoritmos de treinamento abordados pelo software desenvolvido:

### 2.4.1. Backpropagation

O livro de Chauvin e Rumelhart (1995) aborda muito bem o funcionamento e algumas aplicações do algoritmo. Uma breve explicação pode ser feita da seguinte forma:

1. Após os dados chegarem à saída são analisados os erros da camada de saída. É determinado o erro de acordo com a contribuição de cada neurônio da camada de saída, usando como base seu conhecimento (peso).
2. Esses pesos são ajustados para minimizar o erro da saída.
3. Este processo continua de forma regressiva pelas camadas da rede (por isso o nome *Backpropagation*).

### 2.4.2. ResilientPropagation

Este algoritmo foi criado com a finalidade de corrigir o problema da magnitude da derivada parcial de cada peso (usado para descobrir a função erro do passo 1 descrito anteriormente) no *Backpropagation*. Além disso, tem o problema que a taxa de aprendizagem é o mesmo valor para a rede toda. (RIEDMILLER e BRAUN, 1993)

O *ResilientPropagation* utiliza um valor de atualização especial (similar à taxa de aprendizagem) em cada conexão que é automaticamente determinada (ao contrário da forma anterior). (HEATON RESERACH, 2014)

### 2.4.3. ManhattanPropagation

Para corrigir os problemas já descritos do *Backpropagation*, este algoritmo usa a derivada parcial para somente indicar o sinal para a atualização da matriz dos pesos sinápticos, já o valor (módulo) é obtido com uma constante qualquer. Em geral, deve-se começar com uma constante alta e ir diminuindo conforme a necessidade (HEATON RESERACH, 2014).

### 2.4.4. QuickPropagation

Algoritmo desenvolvido por meio de um estudo empírico com o objetivo de aumentar a rapidez do treinamento, em comparação ao *Backpropagation*. (FAHLMAN, 1988)

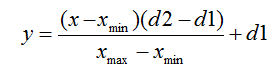
### 2.4.5. ScaledConjugateGradient

Assim como o algoritmo anterior, este foi criado para aumentar a rapidez do treinamento, em comparação ao *Backpropagation*. (MØLLER, 1993)

## 2.5 Normalização

Apesar de uma RNA ser bem “inteligente”, ela não consegue processar o dado de qualquer forma. Ela aceita receber números flutuantes (*float*), de acordo com a função de ativação (entre -1 e 1, ou 0 e 1) (HEATON, 2011).

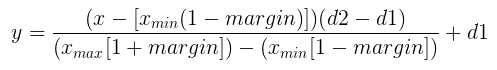
É possível normalizar de várias formas, desde regra de três até com o uso de equações. A ideia básica é verificar qual é o valor máximo possível, o mínimo, e ver a razão do numero entre eles, como mostra na Equação 7.



Equação . Normalização

Neste caso, o *x* é o valor a ser normalizado, *xmax* e *xmin* são os valores extremos da entrada, e *d*1e *d*2 são os valores extremos da saída. Suponhamos que temos n dados de entrada do limite [0,10], então xmin = 0 e xmax = 10. Reduziremos os dados ao limite [0,1], então d1 = 0 e d2 = 1.

Para a implementação do software, além desses limites, foi acrescentado um valor de “gordura”, a fim de evitar que um possível resultado não conhecido venha a interferir na rede como um todo.



Equação . Normalização com margem

Como mostra a Equação 8, a margem é adicionada somente nos valores extremos. É possível observar que para fazer este procedimento, precisa-se percorrer toda a entrada dos dados e descobrir quais são os valores extremos.

A Tabela 2 exemplifica os dados utilizados pelo software no período de 13 dias úteis para a ação de PRIO3. A tabela é separada entre os valores dos preços e em quantidades de ações negociadas (volume). Essa separação é importante para que os significados de cada valor sejam preservados e que o limite tenha um sentido.

As marcas em verde representam os valores máximos da amostra, e os valores em vermelho representam os valores mínimos da amostra. Esses valores serão úteis para a normalização dos dados, conforme já mostrado nas equações anteriores.



Tabela . Preços PRIO3 “puros”

Fonte: Autor (2017)

Para uma margem de 100%, ou seja, os valores extremos dos preços serão 0 e 36 e dos volumes, 0 e 57400 e para uma entrada entre 0 e 1 a equação dos preços fica da seguinte maneira:



Equação . Normalização exemplo

Com isso, a Tabela 3 mostra os todos os dados normalizados conforme as condições anteriores. Para o volume, usa-se a mesma ideia, porém os valores para *xmax* e *xmin* serão 57400 e 0, respectivamente.



Tabela . PRIO3 normalizado

Fonte: Autor (2017)

O valor padrão (*default*) no software é 100%, ou seja, uma ação pode alterar em 100% seus preços limites (dobrar seu preço máximo ou “zerar”) sem que isso interfira na rede. Este é um ambiente totalmente possível, como mostra a Figura 13 e a Figura 14.



Figura . Maiores altas em 5 dias

Fonte: (BARCHART, 2016)



Figura . Maiores quedas em 5 dias

Fonte: (BARCHART, 2016)

# Benchmark

Neste capítulo apresentam-se alguns softwares disponíveis que contribuem na criação de uma RNA que faz predição dos preços de ações, bem como a comparação deles. A comparação é feita com base na quantidade de etapas que o software consegue realizar em relação ao total de etapas necessárias e desejáveis.

## 3.1 Excel + VBA

O Excel é um dos mais famosos editores de planilhas no mercado. Com ele é possível descobrir as ideias ocultas nos dados com a estruturação, formatação, organização e análise, mostrar os dados da melhor forma com o uso de gráficos, tendências e padrões em evidência e compartilhar e colaborar o arquivo com outras pessoas (MICROSOFT, 2016)

O VBA é uma linguagem de programação que auxilia e customiza os aplicativos do Office. Com ele é possível automatizar tarefas repetitivas e integrar entre aplicações do Office. (MICROSOFT, 2009)

Com o Excel é possível criar campos para escolher o período de data em que quer extrair os dados, escolher os papeis e os campos. Após essa operação, com o uso do VBA faz-se a extração dos dados de alguma fonte, como por exemplo, do site do Yahoo. Com os dados devidamente extraídos, descobre os valores limites da amostra e faz a normalização dos valores com possíveis margens no Excel. A parte de treinar a rede é possível com o uso de alguns add-ins como o NeuroXL, ou o NeuroNetwork Add-in. Essas opções não são tão completas, mas é possível criar uma rede, mesmo que bem simples. Aparte de gerar o relatório e executada facilmente com a criação de gráficos no Excel e com cálculo de desvio padrão ou variância usando fórmulas já prontas.

## 3.2 Weka

Weka é uma poderosa coleção de algoritmos de *machine learning* no contexto de *data mining*. O nome faz a referência de um pássaro que não voa, encontrado somente na Nova Zelândia, já que foi desenvolvido na Universidade de Waikato localizado neste país. (UNIVERSITY OF WAIKATO)

Com o Weka é possível normalizar os dados, escolhendo somente os limites (BROWNLEE, 2016). Não da para escolher uma margem de forma fácil, só se aplicar a Equação 8 e descobrir quais serão os valores limites, então essa capacidade será desconsiderada.

Para a criação da rede, podem-se escolher quantas camadas a rede terá, bem como quantos neurônios por camada. A escolha da função de ativação, bem como o algoritmo de aprendizagem não é suportada. Também é possível escolher o período de treinamento, o de validação e configurar alguns avisos caso a rede dê algum tipo de erro (WEKA).

## 3.3 Neuroph Studio

O Neuroph Studio é um framework leve para Java que auxilia o desenvolvimento de arquiteturas de redes neurais. Este software tem o intuito de simplificar o desenvolvimento de redes neurais (NEUROPH).

Com o uso desse software é possível criar uma RNA bem completa, escolhendo tanto quantos neurônios por camada e quantas camas, como também a função de ativação (chamada por eles de função de transferência) por neurônio e, por fim, o algoritmo de aprendizado (chamada de regra de aprendizado). Além da rede, é possível, como no caso do Weka, fazer a normalização dos valores somente escolhendo os limites, sem a margem. (NEUROPH, 2014)

Ele também tem o recurso de gerar um gráfico com o total de erros por iteração, mas nem o desvio padrão nem a variância são nativas. (NEUROPH, 2014)

## 3.4 Matlab

O Matlab é uma ferramenta voltada para otimização ou resolução de problemas científicos ou da engenharia. Ele possui uma linguagem de programação própria e está presente em sistemas de segurança automotiva, veículos espaciais interplanetários, dispositivos de monitoramento de saúde, redes de internet de celulares, entre outros. É usado para *machine learning*, processamento de signos, processamento de imagem, visão computacional, comunicação, computação financeira, robóticas e muito mais (MATHWORKS)

Com ele é possível normalizar os dados de entrada (assim como os dois anteriores, sem a possibilidade de escolher a margem), definir quantas camadas e quantos neurônios por camada, escolher a função de ativação por camada e o algoritmo de treinamento. Além disso, ele também gera um gráfico com o erro por época (DEMUTH e BEALE, 2003).

## 3.5 NeuroFURG

O NeuroFURG é uma ferramenta de construção e simulação de RNA com o foco dos modelos de neurônios Perceptron e Adaline (não abordado neste trabalho) para estudantes de computação. (NEUROFURG, 2011)

Esta ferramenta é muito simples e não tem a funcionalidade de adicionar camadas ocultas, ou escolher quantos neurônios por camada, ou até mesmo a função de ativação, nem o algoritmo de aprendizagem. Com ele só dá para fazer uma rede com dois neurônios, e sem muita complexidade. É possível gerar um gráfico com o erro. (NEUROFURG, 2011)

## 3.6 Comparativo



Tabela . Comparativo softwares

Fonte: Autor (2017)

Podemos notar que nenhum software é capaz de realizar todas as operações. Enquanto em uns são voltados somente na rede em si, em outros é possível fazer a normalização dos dados e a extração, conforme a Tabela 4.

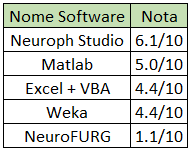


Tabela . Notas softwares

Fonte: Autor (2017)

As notas da Tabela 5 foram calculadas com a quantidade de etapas que o determinado software faz e pela quantidade de etapas totais. Dividindo o primeiro com o segundo valor, tem-se a porcentagem de etapas que o software consegue atingir, e multiplicando esse valor por 10, chega-se nas notas.

Com base na Tabela 4 e na Tabela 5, o conjunto mais eficiente para realizar todo o procedimento de criar e validar uma RNA para predição de ações no mercado financeiro é com o uso do Excel + VBA com o Neuroph Studio.

# REQUISITOS DO SISTEMA DE SOFTWARE

Este capítulo tem como objetivo especificar os requisitos funcionais, não funcionais e a regras de negócio, bem como apresentar o protótipo de telas e o cronograma de atividades do desenvolvimento do software. O texto a seguir tem como objetivo relembrar conceitos e padrões de especificação dos requisitos de software.

## 4.1 Identificação dos requisitos

Por convenção, a referência a requisitos é feita através do nome da subseção onde eles estão descritos, seguidos do identificador do requisito, de acordo com a especificação a seguir:

* **Requisitos funcionais** devem ser identificados por: [RFxxx] – nome e descrição e serem descritos na seção requisitos funcionais.
* **Regras de negócio** devem ser identificadas por: [RNxxx] – nome e descrição e serem descritas na seção requisitos funcionais, subitem regras de negócio.
* **Requisitos não funcionais** devem ser identificados por: [RNFxxx] – nome e descrição e serem descritos na seção requisitos não funcionais.

Os requisitos devem ser identificados com um identificador único. A numeração inicia com o identificador [RF001], [RN001] ou [RNF001] e prossegue sendo incrementada à medida que forem surgindo novos requisitos ou regras.

Por convenção, a referência aos casos de uso é feita através do nome da subseção onde eles estão descritos, seguidos do identificador do caso de uso, de acordo com a especificação a seguir:

Casos de Uso devem ser identificados por: CSUXXX – nome a ser descrito na função modelagem funcional.

Os casos de uso devem ser identificados com um identificador único. A numeração inicia com o identificador CSU001 e prossegue sendo incrementada à medida que forem surgindo novos casos de uso.

### 4.1.1 Prioridades dos requisitos

Para estabelecer a prioridade dos requisitos, foram adotadas as denominações “essencial”, “importante” e “desejável”.

* **Essencial** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
* **Importante** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
* **Desejável** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

### 4.1.2 Requisitos Funcionais

Neste item devem ser descritos os requisitos funcionais que especificam ações que um sistema deve ser capaz de executar, ou seja, os objetivos do sistema, incluindo prioridade e regras de negócio. A seguir são apresentados exemplos.

**[RF001] – Importar Dados da BMF**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | X | Essencial | - | Importante | - | Desejável |

**Descrição**: Este requisito permite que o sistema importe os preços de fechamento e os volumes ajustados por proventos da Bolsa de Valores do Brasil.

**[RF002] – Gerenciar rede neural**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | X | Essencial | - | Importante | - | Desejável |

**Descrição**: Este requisito permite que o usuário gerencie uma RNA.

**[RF003] – Treinar rede neural**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | X | Essencial | - | Importante | - | Desejável |

**Descrição**: Este requisito permite que o usuário configure o ambiente de treinamento de uma RNA.

**[RF004] – Gerar relatório**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | - | Essencial | X | Importante | - | Desejável |

**Descrição**: Este requisito permite que o sistema gere um gráfico comparando os preços esperados e o calculado, bem como o coeficiente de variação e o desvio padrão.

### 4.1.3 Requisitos Não-Funcionais

Neste item devem ser apresentados os requisitos não funcionais, que especificam restrições sobre os serviços ou funções providas pelo sistema. A seguir são apresentados alguns exemplos de requisitos não funcionais.

**Requisitos de Produto:** Conjunto de requisitos para prover conteúdo aos usuários cadastrados.

**[RNF001] – Usabilidade**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | - | Essencial | - | Importante | X | Desejável |

**Descrição**: O sistema deve prover interface simples e intuitiva, de fácil navegação para facilitar o uso do mesmo por parte dos usuários.

**[RNF002] – Apresentação da Interface Gráfica**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | X | Essencial | - | Importante | - | Desejável |

**Descrição**: O sistema deve fazer uso, exclusivamente, da língua inglesa para todo e qualquer texto apresentado.

**[RNF003] – Linguagem de programação adotada**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | X | Essencial | - | Importante | - | Desejável |

**Descrição**: O sistema deve utilizar a linguagem Java versão 1.8.

**[RNF004] – Sistema Operacional alvo**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | X | Essencial | - | Importante | - | Desejável |

**Descrição**: O sistema deve ser executado em qualquer SO que suporte a máquina virtual do Java (JVM).

### 4.1.4 Regras de Negócio

**[RN001] – Alterar topoligia reseta valores**

**Descrição:** Toda vez que alterar a topoligia de uma rede, suas pesos serão populadas com valores aleatórios.

### 4.1.5 *Product* e *Sprint Backlog*

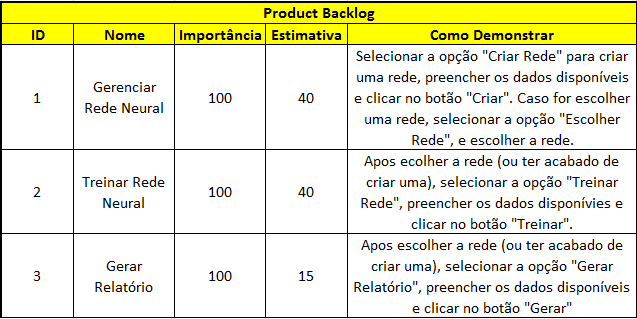


Figura . *Product Backlog*

Fonte: Autor (2017)

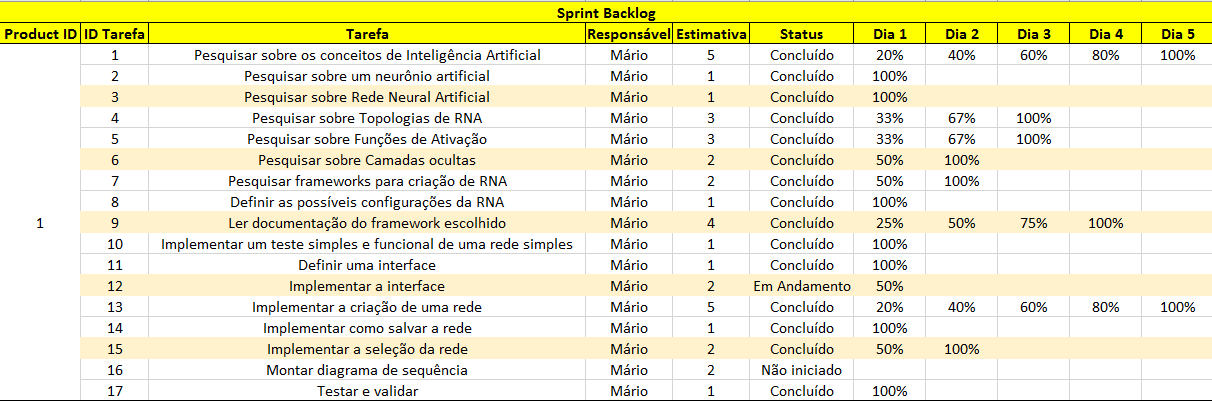


Figura . *Sprint Backlog* 1

Fonte: Autor (2017)

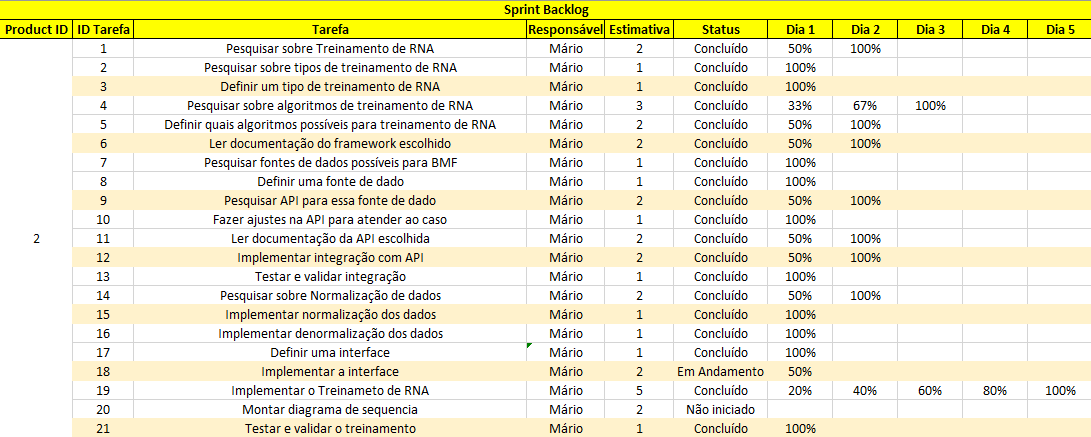


Figura . *Sprint Backlog* 2

Fonte: Autor (2017)

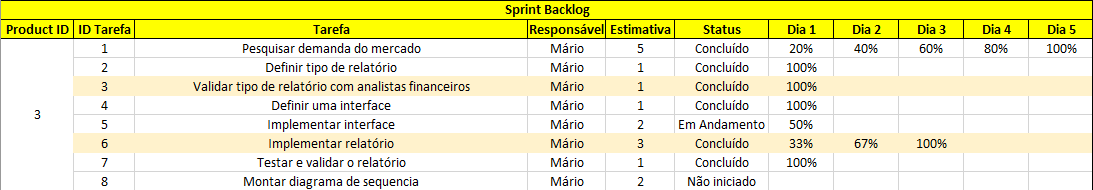


Figura . *Sprint Backlog* 3

Fonte: Autor (2017)

## 4.2 Modelagem dos requisitos funcionais

Neste item devem ser descritos os requisitos a serem atendidos funcionalmente pelo sistema de uma forma simples, possibilitando a compreensão do comportamento do sistema pela perspectiva do usuário. Devem ser descritos os atores e o diagrama de caso de uso. A seguir um exemplo de especificação de atores, do diagrama de caso de uso e da especificação de caso de uso.

### 4.2.1 Atores

A seguir são apresentados exemplos da especificação de atores.

**USUÁRIO:** Representa alguém que utiliza o sistema..

**SISTEMA BMF:** Representa a integração com o WebService da BMF.

### 4.2.2 Diagrama de Caso de uso

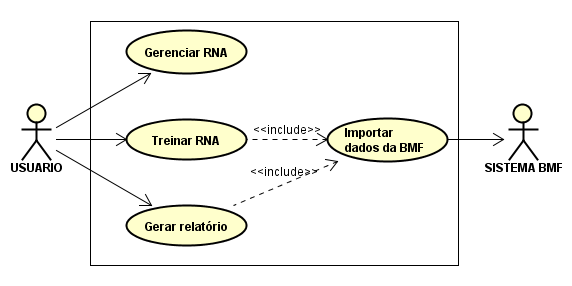


Figura . Atores

Fonte: Autor (2017)

### 4.2.3 Especificação do Caso de Uso

|  |  |
| --- | --- |
| **CSU001 – Importar Dados da BMF** | |
| Sumário: | Importar Dados da BMF |
| Ator Primário: | Sistema BMF |
| Casos de Uso Associados: CSU002 | |
| **Pré-condição:** | |
| **Fluxo Principal**  1**-** O caso de uso inicia quando o Sistema necessita das informações da BMF.  2- O sistema extrai as informações da BMF.  3**-** O caso de uso termina. | |
| **Fluxo de Excessão:** | |
| **Pós-condições:**  **a.** O Sistema armazena os dados. | |
| **Requisitos:** RF001 | |
| **Regras de Negócio:** | |
| **Interface:** | |

|  |  |
| --- | --- |
| **CSU002 – Criar Rede Neural** | |
| Sumário: | Criar Rede Neural |
| Ator Primário: | Usuário |
| Casos de Uso Associados: | |
| **Pré-condição:** | |
| **Fluxo Principal**   1. O caso de uso inicia quando o usuário deseja criar uma RNA. 2. O Sistema solicita a quantidade de camadas, de neurônios por camada, função de ativação por camada, se tem ou não bias na camada, o *dropout* da camada, ação a ser comparadas, a quantidade de data necessária para gerar o próximo dado. 3. O usuário informa os campos necessários. 4. O caso de uso termina. | |
| **Pós-condições:**  **a.** Rede Neural criada. | |
| **Requisitos:** RF002 | |
| **Regras de Negócio:** | |
| **Interface:** I001 | |

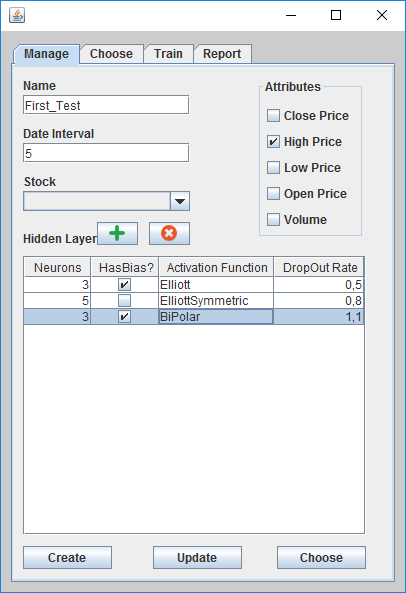


Figura . I001 – Gerenciar Rede Neural

Fonte: Autor (2017)

|  |  |
| --- | --- |
| **CSU003 – Alterar Rede Neural** | |
| Sumário: | Alterar RNA |
| Ator Primário: | Usuário |
| Casos de Uso Associados: CSU002 | |
| **Pré-condição:**  **a.** O Usuário deve ter selecionado uma RNA.(CSU002) | |
| **Fluxo Principal**   1. O caso de uso inicia quando o usuário deseja alterar a RNA selecionada 2. O Sistema informa as configurações atuais da RNA. 3. O Usuário informa as novas configurações. 4. O Sistema altera os pesos dos neurônios para um número randômico. (RN001) 5. O caso de uso termina | |
| **Pós-condições:**  **a.** A RNA atual é atualizada**.** | |
| **Requisitos:** RF002. | |
| **Regras de Negócio:** RN001 | |
| **Interface:** I001 | |

|  |  |
| --- | --- |
| **CSU004 – Treinar Rede Neural** | |
| Sumário: | Treinar Rede |
| Ator Primário: | Usuário |
| Ator Secundário: | Sistema BMF |
| **Caso de Uso associado:** CSU002. | |
| **Pré-condição:**   1. O Usuário deve ter selecionado uma RNA. (CSU002) 2. Deve haver conexão com a internet | |
| **Fluxo Principal**   1. O Sistema solicita as datas ‘de’ e ‘para’, as regras de normalização (limite inferior e superior, e a margem), o algoritmo de aprendizagem, o número máximo de iterações e o erro máximo. 2. O Usuário informa os dados. 3. O Sistema envia puxa as informações do Sistema da BMF 4. O caso de uso termina. | |
| **Fluxo de Excessão (3) – Sem Conexão com a internet**   1. O Sistema informa que não está conectado à internet 2. O caso de uso termina. | |
| **Fluxo de Excessão (3) – Sem dados da BMF**   1. O Sistema informa que não foi possível obter os dados da BMF 2. O caso de uso termina. | |
| **Pós-condições:**  **a.** O Sistema treina a RNA. | |
| **Requisitos:** RF003. | |
| **Interface**: I002 | |

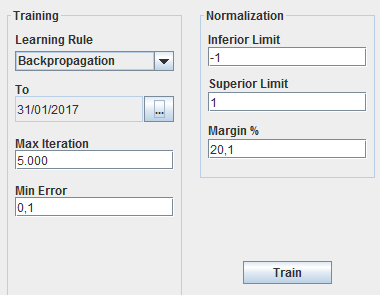


Figura . I002 - Treinar Rede Neural

Fonte: Autor (2017)

|  |  |
| --- | --- |
| **CSU005 – Gerar relatório** | |
| Sumário: | Gerar relatório |
| Ator Primário: | Usuário |
| Ator Secundário: | Sistema BMF |
| **Caso de Uso associado:** CSU002, CSU003. | |
| **Pré-condição:**   1. O Usuário deve ter configurado as variáveis. (CSU002) 2. Deve haver conexão com a internet. | |
| **Fluxo Principal**   1. O Sistema solicita as datas ‘de’ e ‘para’. 2. O Usuário informa os dados. 3. O Sistema envia puxa as informações do Sistema da BMF. 4. O caso de uso termina. | |
| **Fluxo de Excessão (3) – Sem Conexão com a internet**   1. O Sistema informa que não está conectado à internet 2. O caso de uso termina. | |
| **Fluxo de Excessão (3) – Sem dados da BMF**   1. O Sistema informa que não foi possível obter os dados da BMF 2. O caso de uso termina. | |
| **Pós-condições:**  **a.** O Sistema gera o relatório. | |
| **Requisitos:** RF004. | |
| **Interface**: I003 | |

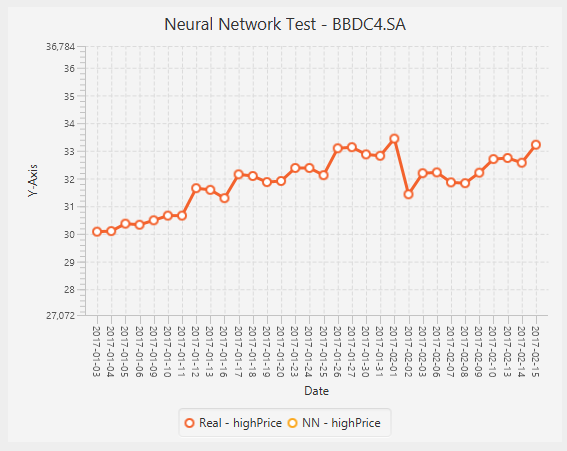


Figura . I003 - Gerar Relatório

Fonte: Autor (2017

# Bibliografia

BARCHART. Stocks - Performance. **Barchart**, 2016. Disponivel em: <http://www.barchart.com/stocks/performance/5day.php>. Acesso em: 10 Outubro 2016.

BROWNLEE, J. How to Normalize and Standardize Your Machine Learning Data in Weka. **Machine Learning Mastery**, 2016. Disponivel em: <http://machinelearningmastery.com/normalize-standardize-machine-learning-data-weka/>. Acesso em: 2 November 2016.

CHABOUD, A. P. et al. Rise of the machines: Algorithmic trading in the foreign exchange market. **The Journal of Finance**, v. 69, n. 5, p. 2045-2084, 2014.

CHAUVIN, Y.; RUMELHART, D. E. **Backpropagation:** theory, architectures, and applications. New Jersey: Psychology Press, 1995.

CHERMAN, E. A.; TSOUMAKAS, G.; MONARD, M. C. Active Learning Algorithms for Multi-label Data. **Artificial Intelligence Applications and Innovations: 12th IFIP WG 12.5 International Conference and Workshops**, Greece, 16-18 September 2016. 267-279.

CONNOR, J. T.; MARTIN, D.; ATLAS, L. E. Recurrent Neural Networks and Robust Time Series Prediction. **IEEE TRANSACTIONS ON NEURAL NETWORKS,** v. 5, n. 2, p. 240-254, Março 1994.

DEMUTH, H.; BEALE, M. **Neural Network Toolbox:** for use with Matlab. 4. ed. Natick: MathWorks, 2003. Disponivel em: <http://www.dsc.ufcg.edu.br/~hmg/disciplinas/graduacao/rn-2016.1/RN-5b%20-%20Redes%20Neurais%20no%20Matlab.pdf>.

DO CARMO ROQUE, R.; DE MELLO, F. L. Estudo sobre a empregabilidade da previsão do índice BOVESPA usando Redes Neurais Artificiais, 2009.

FAHLMAN, S. E. An empirical study of learning speed in back-propagation networks. **Carnegie Mellon University: Research Showcase**, 1988.

FORTUNA, E. **Mercado Financeiro:** produtos e serviços. 11ª. ed. Rio de Janeiro: Qualitymark Editora Ltda, 2008.

HAYKIN, S. **FEEDFORWARD NEURAL NETWORKS:** AN INTRODUCTION. [S.l.]: [s.n.], 1998.

HAYKIN, S. **Neural networks and learning machines**. Third Edition. ed. Upper Saddle River, NJ, USA: Pearson, 2009.

HEATON RESERACH. Javadoc Encog 3.3, 2014. Disponivel em: <http://heatonresearch-site.s3-website-us-east-1.amazonaws.com/javadoc/encog-3.3/org/encog/package-summary.html>. Acesso em: 1 Outubro 2016.

HEATON, J. **Programming Neural Networks with Encog3 in Java**. St. Louis: Heaton Research, 2011.

HENDERSHOTT, T.; JONES, C. M.; MENKVELD, A. J. Does algorithmic trading improve liquidity? **The Journal of Finance**, v. 66, n. 1, p. 1-33, 2011.

IBM. IBM 100 - Deep Blue. **IBM - Icons of Progress**, 2011. Disponivel em: <http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/>. Acesso em: 22 Outubro 2016.

KATHLEEN, C.; KOMAKI, I. How This Hedge Fund Robot Outsmarted Its Human Master. **Bloomberg News**, 2016. Disponivel em: <http://www.bloomberg.com/news/articles/2016-08-21/hedge-fund-robot-outsmarts-human-master-as-ai-passes-brexit-test>. Acesso em: 22 Outubro 2016.

KIMOTO, T. et al. Stock Market Prediction System with Modular Neural Networks. **1990 IJCNN International Joint Conference on**, Junho 1990. 1-6.

KRIEGER, P. E. Uso de redes neurais artificiais para predição da bolsa de valores, Itajaí, 2012. 91 folhas. Trabalho de conclusão de curso (Graduação em Ciência da Computação) - Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí, 2012.

MANTOVANI, W. A. **Utilização de Redes Neurais Recorrentes na Caracterização de Cargas Não Lineares em Sistemas Elétricos**. Ilha Solteira. 2011. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Estadual Paulista, Ilha Solteira.

MARKOWITZ, H. Portfolio Selection. **The Journal of Finance**, 7, n. 1, Março 1952. 77-91.

MATHWORKS. Matlab. **MathWorks Products**. Disponivel em: <https://www.mathworks.com/products/matlab>. Acesso em: 2 Novembro 2016.

MCCULLOCH, W. S.; PITTS, W. H. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, n. 4, p. 115-133, 1943.

MICROSOFT. Getting Started with VBA in Office 2010. **Documentation Office 2010**, 2009. Disponivel em: <https://msdn.microsoft.com/en-us/library/office/ee814735(v=office.14).aspx>. Acesso em: 31 Outubro 2016.

MICROSOFT. Produtos Office: Excel. **Produtos Office**, 2016. Disponivel em: <https://products.office.com/pt-BR/excel#>. Acesso em: 31 Outubro 2016.

MØLLER, M. F. A scaled conjugate gradient algorithm for fast supervised learning. **Neural networks**, v. 6, n. 4, p. 525-533, 1993.

MONARD, M. C.; BARANAUSKAS, J. A. **Sistemas Inteligentes:** Fundamentos e Aplicações. [S.l.]: Manole Ltda, 2003.

NEUROFURG. Uma ferramenta de apoio ao ensino de Redes Neurais Artificiais. **NeuroFURG**, 2011. Disponivel em: <http://neurofurg.sourceforge.net/>. Acesso em: 2 Novembro 2016.

NEUROPH. Neuroph Studio JavaDoc. **Neuroph Studio**, 2014. Disponivel em: <http://neuroph.sourceforge.net/javadoc/index.html>. Acesso em: 2 Novembro 2016.

NEUROPH. Neuroph v2.92. **Neuroph**. Disponivel em: <http://neuroph.sourceforge.net/>. Acesso em: 2 Novembro 2016.

NUMERAI. About: Numerai. **Numerai web site**, 2016. Disponivel em: <https://numer.ai/about>. Acesso em: 16 November 2016.

OLIVEIRA, M. A. **Aplicação de redes neurais artificiais na análise de séries temporais econômico-financeiras**. São Paulo, p. 316 folhas. 2007. Tese (Doutorado em Administração) - Faculdade de Economia, Administração e Contabilidade da Universidade de São Paulo.

REFENES, A. N.; ZAPRANIS, A.; FRANCIS, G. Stock performance modeling using neural networks: A comparative study with regression models. **Neural Networks**, Oxford, 7, n. 2, 1994. 375-388.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. **Neural Networks, 1993., IEEE International Conference On.**, 1993. 586-591.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence:** a modern approach. 3ª. ed. New Jersey: Prentice Hall, 2005.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, n. 1, p. 1929-1958, 2014.

TAFNER, M. A. Redes Neurais Artificiais: Aprendizado e Plasticidade, 1998. Disponivel em: <http://www.cerebromente.org.br/n05/tecnologia/rna.htm#neuronio>. Acesso em: 25 Setembro 2016.

THOMAZ, C. E.; VELLASCO, M. M. B. R. Análise de Tendências de Mercado por Redes Neurais Artificiais. **7º CBRN – Congresso Brasileiro de Redes Neurais**, Natal, 2005.

TURING, A. M. Computing machinery and intelligence. **Mind**, v. 59, n. 236, p. 433-460, 1950.

UNIVERSITY OF WAIKATO. Weka 3: Data Mining Software in Java. **Weka**. Disponivel em: <http://www.cs.waikato.ac.nz/ml/weka/documentation.html>. Acesso em: 2 Novembro 2016.

VON ZUBEN, F. J.; ATTUX, R. R. F. Redes Neurais Artificiais – Parte I, 2010. Disponivel em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia004\_1s10/notas\_de\_aula/topico1\_IA004\_1s10\_Parte1.pdf>. Acesso em: 25 Setembro 2016.

WEKA. Class MultilayerPerceptron. **Weka Documentation**. Disponivel em: <http://weka.sourceforge.net/doc.stable/weka/classifiers/functions/MultilayerPerceptron.html>. Acesso em: 2 Novembro 2016.

WINRICH, B. **Training and Source Code Generation for Artificial Neural Networks**. Rhode Island. 2015. Tese de Doutorado (Master of Science in Computer Science). University of Rhode Island.