UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE - UERN
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO - UFERSA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALDEMÁRIO ALVES DA SILVA

# DESENVOLVIMENTO E APLICAÇÃO DE HEURÍSTICA PARA CALCULAR PESOS E BIAS INICIAIS PARA O "BACK-PROPAGATION" TREINAR REDE NEURAL PERCEPTRON MULTICAMADAS

MOSSORÓ-RN
2017

ALDEMÁRIO ALVES DA SILVA

# DESENVOLVIMENTO E APLICAÇÃO DE HEURÍSTICA PARA CALCULAR PESOS E BIAS INICIAIS PARA O "BACK-PROPAGATION" TREINAR REDE NEURAL PERCEPTRON MULTICAMADAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - associação ampla entre a Universidade Estadual do Rio Grande do Norte (UERN) e a Universidade Federal Rural do Semi-Árido (UFERSA) - para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Dario José Aloise
Coorientador: Prof. Dr. Araken de Medeiros Santos

MOSSORÓ-RN
2017

## FICHA CATALOGRÁFICA

S586d Silva, Aldemário Alves da.

Desenvolvimento e aplicação de Heurística para calcular pesos e bias iniciais para o "Back-Propagation" treinar Rede Neural Perceptron Multicamadas. / Aldemário Alves da Silva. -- Mossoró, 2017.

112f.: il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal Rural do Semi-Árido / Universidade Estadual do Rio Grande do Norte.

Orientador: Prof. Dr. Dario José Aloise.

Coorientador: Prof. Dr. Araken de Medeiros Santos.

1.Aprendizado de máquina. 2.Rede Neural Perceptron Multicamadas. 3.Heurística. 4.Reconhecimento de padrão. 5.Classificação de dados. I. Título

CDU: 006.31

Bibliotecária Keina Cristina Santos Sousa
CRB15/120

ALDEMÁRIO ALVES DA SILVA

# DESENVOLVIMENTO E APLICAÇÃO DE HEURÍSTICA PARA CALCULAR PESOS E BIAS INICIAIS PARA O "BACK-PROPAGATION" TREINAR REDE NEURAL PERCEPTRON MULTICAMADAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 18 / 08 / 2017

Prof. Dr. Dario José Aloise
Orientador e Presidente - UERN

Prof. Dr. Araken de Medeiros Santos
Coorientador - UFERSA

Prof. Dr. Francisco das Chagas de Lima Júnior
Examinador Interno - UERN

Prof. Dr. Marcelo Augusto Costa Fernandes
Examinadora Externa - UFRN

# Agradecimentos

Agradeço principalmente a Deus, o criador do Universo, pela força, saúde e inteligência para seguir em frente e conseguir terminar esta pesquisa.

Agradeço:

- o apoio e incentivo e do meu pai (Altino Pereira) e de minha mãe (Maria Marinete).

- a compreensão e companheirismo da minha esposa (Ana Maria).

- Aos meus filhos (André e Andrei) valeu pela compreensão.

Agradeço aos professores do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal Rural do Semi-Árido (UFERSA) e Universidade Estadual do Rio Grande do Norte (UERN), obrigado pelo ensino.

Agradeço a orientação dos professores Dr. Dario José Aloise (UERN) e Dr. Araken de Medeiros Santos (UFERSA).

Agradeço aos servidores da UFERSA, especialmente aos colegas e amigo(as) da Biblioteca "Orlando Teixeira", ou que em outro momento já fez parte da biblioteca.

Por fim, agradeço aos irmãos e amigos da Igreja Evangélica "Assembleia de Deus" pela solidariedade.

# Resumo

O treinamento de Rede Neural Perceptron Multicamadas (RNPM) feito por algoritmo exato para encontrar a máxima acurácia é NP-Difícil. Sendo assim, usa-se o algoritmo "Back-Propagation" que necessita de um ponto de partida (pesos e bias iniciais) para computar o treinamento da RNPM. Esta pesquisa desenvolveu e aplicou um algoritmo heurístico HeCI - Heurística para Calcular Pesos e Bias Iniciais - para computar os dados de treinamento da RNPM e retornar o ponto de partida para o "Back-Propagation". A HeCI usa Análise de Componentes Principais, Método dos Mínimos Quadrados, Função de Densidade de Probabilidade da Normal Distribuição Gaussiana, duas configurações estratégicas e controla parcialmente o número de épocas de treinamento da RNPM. Experimentalmente, a RNPM foi treinada usando "Back-Propagation" com HeCI, para reconhecer padrões e resolver problemas de classificação de dados. Seis estudos de caso com "datasets" entre as áreas de Saúde, Negócio e Botânica foram usados nos experimentos. A metodologia desta pesquisa usa análise Dedutiva pelo método Experimental com abordagem Quantitativa e testes de hipóteses: Teste de Fridman com Pós-Teste de Tukey HSD Post-hoc e Teste de Wilcoxon-M-W. Os resultados de acurácia incrementaram melhoria significativa atestada pela avaliação dos testes de hipóteses, inferindo estatisticamente robustez de resultado motivado pela HeCI.

**Palavras-Chave:** Aprendizado de Máquina. Rede Neural Perceptron Multicamadas. Heurística. Reconhecimento de Padrões. Classificação de Dados.

# Abstract

The training of Multilayer Perceptron Neural Network (MLPNN) done by exact algorithm to find the maximum accuracy is NP-hard. Thus, we use the algorithm Back-Propagation who needs a starting point (weights and bias initials) to compute the training of the MLPNN. This research has developed and implemented a heuristic algorithm HeCI - Heuristic to Calculate Weights and Bias Initials - to compute the data to train the MLPNN and return the starting point for the Back-Propagation. HeCI uses Principal Component Analysis, Least Square Method, Probability Density Function of the Normal Gaussian Distribution, two strategic configurations, and partially controls the number of MLPNN training epochs. Experimentally, HeCI was used with Back-Propagation in MLPNN training to recognize patterns and solve data classification problems. Six case studies with datasets between Health, Business and Botany were used in the experiments. The methodology of this research uses Deductive analysis by the Experimental method with Quantitative approach and hypothesis tests: Test of Fridman with post Teste of Tukey HSD Post-hoc and Wilcoxon Test-M-W. The results of accuracy have increased significantly improving attested by evaluation of tests of hypotheses, inferringstatistical robustness of the result motivated by HeCI.

**Keywords:** Machine Learning. Neural Network Multilayer Perceptron. Heuristic. Pattern Recognition. Data Classification.

# Lista de Algoritmos

# Lista de Abreviaturas e Siglas

| | |
|---|---|
| AM | Aprendizagem de Máquina |
| AS | Aprendizagem Supervisionada |
| CAPES | Comissão de Aperfeiçoamento de Pessoal do Nível Superior |
| DI | Documento de Implantação |
| HeCI | Heurística para Calcular Pesos e Bias Iniciais |
| InitNW | Algoritmo de Inicializar Pesos e Bias de Nguyem-Widrow |
| MidPoint | Valor do Ponto Médio do Vetor de Entrada de Dados |
| MMQ | Método dos Mínimos Quadrados |
| PAC | Aprendizagem Provavelmente Aproximadamente Correta |
| PCA | Análise de Componentes Principais |
| PO | Pesquisa Operacional |
| Rand | Valores Aleatórios |
| RandS | Valores Aleatórios Simétricos |
| RandSmall | Valores Aleatórios Pequenos |
| RNA | Rede Neural Artificial |
| RNPM | Rede Neural Perceptron Multicamadas |
| TrainBR | "Back-Propagation Bayesian Regularization" |
| TrainGDA | "Back-Propagation Gradient Descent with Adaptive Learning Rate" |
| TrainGDX | "Back-Propagation Gradient Descent with Momentum Adaptive Learning Rate |
| TrainRP | "Back-Propagation Resiliente Propagation" |

# Lista de Ilustrações

# Lista de Símbolos

| | |
|---|---|
| $e$ | Constante de Leonhard Euler |
| $\mathbb{R}$ | Conjunto de Números Reais |
| $\mathbb{Z}^+$ | Conjunto de Números Inteiros Positivos com o Zero |
| det | Determinante |
| $\mathcal{H}_0$ | Hipótese Nula |
| $\mathcal{H}_1$ | Hipótese Alternativa |
| $\mapsto$ | Implica em |
| $X^T$ | Matriz Transposta |
| $X^{-1}$ | Matriz Inversa |
| $\sum$ | Soma |
| $\forall$ | Para Todo |
| $\in$ | Pertence |

# Lista de Quadros

# Lista de Tabelas

# Sumário

# 1 Introdução

A Ciência da Computação busca algoritmos capazes de processar dados em quantidade de passos finitos que forneçam uma saída de informação com ótima acurácia para qualquer entrada de dados válida. Explica Brookshear (2013, p. 2) que a computação constrói uma base científica que prover estrutura física e lógica para aplicações diversas, apontando à investigação de algoritmos como base para o avanço computacional.

Nas palavras de Cormen et al. (2012, p. 3), algoritmo "é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída", remetendo a um conjunto finito de símbolos para computar dados pela sequência: entrada, processamento e saída.

Devido a grande quantidade e complexidade da entrada de dados gerada por problemas do mundo real, há o surgimento e a necessidade de processar modelos matemáticos para representar o problema a ser computado. Explica Hillier e Lieberman (2006, p. 12) que "após a formulação do modelo matemático [...], a próxima fase em um estudo de Pesquisa Operacional é desenvolver o procedimento de solução".

A Pesquisa Operacional segundo Pizzolato e Gandolpho (2013, p. 2) "faz uso de métodos quantitativos para gerenciamento de sistemas e a tomada de decisão". É responsável pela busca viável e finito de valor ótimo global de mínimo ou máximo de uma função matemática, que represente de forma aproximada ou exata o ótimo global para resolver o problema.

Surge dentro da Pesquisa Operacional, os métodos de otimização, entre eles a Rede Neural Artificial (RNA) definido por Haykin (2001, p. 28): "é um processador maciçamente paralelamente distribuído constituído de unidades [perceptron] de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso" em Aprendizagem de Máquina (AM).

A RNA consiste em um método baseado em otimização, descrito em Faceli et al. (2011, p. 62): "Algumas técnicas de AM realizam a busca pela hipótese que descreve os dados recorrendo à otimização de alguma função. Nesse processo, um problema de aprendizado é [...] um problema de otimização em [...] minimizar (ou maximizar) uma função objetivo".

Com o avanço da pesquisa científica é demonstrado por Russel e Norvig (2013, p. 662) apud Minsky e Papert (1969) uma prova através de teorema sobre a AM para o perceptron, que uma RNA de camada única poderia representar conceitos e resolver problemas apenas linearmente separáveis, e afirmou a falta de algoritmo de AM para Redes Neurais de Múltiplas Camadas (RNPM).

Para sanar a falta de um algoritmo de aprendizagem para a RNPM, Rumelhart et al.

(1986, p. 328) demonstraram um algoritmo de Retropropagação de Erro ou "Back-Propagation" para treinamento de RNPM.

O "Back-Propagation" a partir de um ponto de partida de valores iniciais de pesos e bias calcula a saída de RNPM. Então a diferença entre a saída e a resposta supervisionada é recalculada e retropropagada na RNPM. Todos os pesos e bias em todas as camadas são recalculados para minimizar o erro de saída. O processo é repetido até atingir "$n$" épocas de treinamento ou minimizar o erro Rumelhart et al. (1986).

É entendido em Rumelhart et al. (1986), Haykin (2001) e Luger (2013) que o "Back-Propagation" enfrenta problemas de ficar preso em algum mínimo local e a dificuldade de encontrar valores ótimos de pesos[1] e bias[1] com máxima acurácia: "Temos agora um processo de aprendizagem que pode, em princípio, desenvolver um conjunto de pesos para produzir um mapeamento arbitrário da entrada à saída".

Destaca Haykin (2001) e Samarasinghe (2006) pontos importantes a considerar para o "Back-Propagation" treinar RNPM: quais valores inicializar os pesos e bias para não saturar o perceptron, não ficar preso a um mínimo local e convergir para a máxima acurácia.

É constatado, explica Faceli et al. (2011), que variações de técnicas específicas para o "Back-Propagation" tem sido estudadas, dado um conjunto de dados a ser aprendido pela RNPM, de maneira que maximize a acurácia, para aumento da generalização do conhecimento em ambiente de produção.

Para o atual estado da arte em inteligência computacional aplicada a RNPM, segundo Luger (2013), continua o desafio de treinamento para classificação de grupos específico dentro de um conjunto de dados em pior caso, fomentando a pesquisa em busca de procedimentos que adicione melhoria ao “Back-Propagation" para encontrar o ótimo global de pesos e bias.

Além disso, afirma Livni Roi e Shamir (2014) e Zhang et al. (2015), que a AM de uma RNPM é NP-Difícil, crítica em convergência para o ótimo global de pesos e bias, quando aplicada aos mais diversos problemas do mundo real para resolve-los com a máxima acurácia.

## 1.1 Problema de Pesquisa

Dada a aplicação de RNPM em aprender a reconhecer padrões para classificar dados em grupos específicos, pergunta-se:

Quais valores inicializar pesos e bias para o "Back-Propagation" convergir para a máxima acurácia?

---

[1] Adaptado de Luger (2013) Peso ou Peso Sináptico e Bias: um valor constante, positivo ou negativo, que representa de forma numérica o conhecimento a ser aprendido pela RNPM, dentro de um espaço no $R^n$ possivelmente grande, mas finito É o número constante que multiplica a entrada de dados para indução do conhecimento.

## 1.2 Justificativa

Diante do problema de pesquisa em foco, é feito uma investigação em literatura científica e indagado sobre a justificativa da questão, delimitando o tema com autores específicos ao assunto.

Investiga-se RNPM, sobre épocas de treinamento e valores de ótimo global para pesos e bias, ou seja, a máxima acurácia, focando em melhorar a busca por valores que produzam maior generalização do conhecimento aprendido por RNPM.

A literatura científica sobre o algoritmo "Back-Propagation" e RNPM é analizada, buscando justificativa e informação sobre o problema de pesquisa. Para tanto, observa-se:

Quadro 1: Confirmação do Problema de Pesquisa: Literatura Científica - Parte 1/2

| Autor | Afirmação |
|---|---|
| Haykin (2001, p. 257) | O algoritmo Retropropagação [...] no espaço de pesos [...] tem tendência a ziguezaguear em torno da verdadeira direção que leva a um mínimo na superfície de erro. [. . . ] tende a convergir lentamente. [. . . ], a taxa de convergência [...] pode tornar o algoritmo martirizante do ponto de vista computacional. |
| Kovács (2006, p. 64) | Foram indicados problemas práticos com relação a eficiência do algoritmo [Back-Propagation] em função da natureza dos elementos do conjunto de treinamento. A natureza desse conjunto [...] influi grandemente na velocidade do processo de convergência. |
| Braga et al. (2007, p. 79) | A principal dificuldade no treinamento de redes [RNPM] com o algoritmo [Back-Propagation] está relacionada à sua sensibilidade às características da superfície de erro, o que dificulta a sua convergência [encontrar os pesos ótimos globais] [...]. |
| Silva et al. (2010, p. 157) | Em virtude de a superfície de erro produzida pela PMC [perceptron multicamadas] ser não-linear, há então a possibilidade de que o processo de aprendizado direcione a matriz de pesos da rede para um ponto de mínimo local, que pode não corresponder aos valores mais apropriados aos propósitos de generalização de resultados [máxima acurácia]. |
| Faceli et al. (2011, p. 120) | [. . . ] a rede [RNPM] pode convergir para um mínimo local ou demorar muito para encontrar uma solução adequada. [. . . ] Uma crítica feita ao algoritmo [Back-Propagation] é sua lentidão na convergência para um bom conjunto de pesos e a sua queda de desempenho quando utilizado em grandes conjuntos de dados e problemas complexos. |

Fonte: Produção do Autor (2017).

É unânime a constatação dos autores do Quadro 1, em descrever o problema de encontrar pesos ótimos globais para a máxima acurácia de RNPM treinada pelo "Back-Propagation".

O problema de convergência para o ótimo global de pesos e bias, em generalizar o conhecimento com máxima acurácia por RNPM, é de difícil solução, conforme afirma:

Quadro 2: Confirmação do Problema de Pesquisa: Literatura Científica - Parte 2/2

| Autor | Afirmação |
|---|---|
| Russel e Norvig (2013, p. 639-641) | A principal complicação vem da adição de camadas ocultas da rede [RNPM]. É necessário certa quantidade de esforço [computacional] para obter a estrutura da rede correta e alcançar a convergência para algo próximo ao ótimo global no espaço de peso. |
| Luger (2013, p. 390, 632) | [...] o algoritmo Retropropagação tem as suas próprias dificuldades. Como no caso da subida de encosta, ele pode convergir para mínimos locais [...] Por fim o custo computacional da retropropagação pode ser elevado, em especial quando a rede [RNPM] converge lentamente. [...] Encontrar a solução mínima de erros é NP-Difícil. |
| Livni Roi e Shamir (2014, p. 856) | Os resultados teóricos existentes são na sua maioria negativo, mostrando que a aprendizagem com sucesso com estas redes é computacionalmente difícil no pior caso. [...] Por redução de k-coloração foi demonstrado que encontrar os pesos que melhor se adaptam ao conjunto de treinamento é NP-Difícil. |
| Zhang et al. (2015, p. 1) | As redes Neurais têm sido aplicadas com sucesso em muitas áreas da inteligência artificial [...], mas o treinamento de tal rede é NP-Difícil. |

Fonte: Produção do Autor (2017).

Com a investigação sobre o "Back-Propagation" e RNPM, conforme os Quadros 1 e 2, levanta-se a hipótese para a adição de melhoria por meio do desenvolvimento de um algoritmo heurístico, que leia os dados de treinamento da RNPM e calcule os pesos e bias iniciais, para o ponto de partida do "Back-Propagation".

Aplicando uma RNPM em problemas de Reconhecimento de Padrões para classificar dados, primeiro conclui-se a arquitetura de RNPM com "$n$" perceptron de modelo matemático:

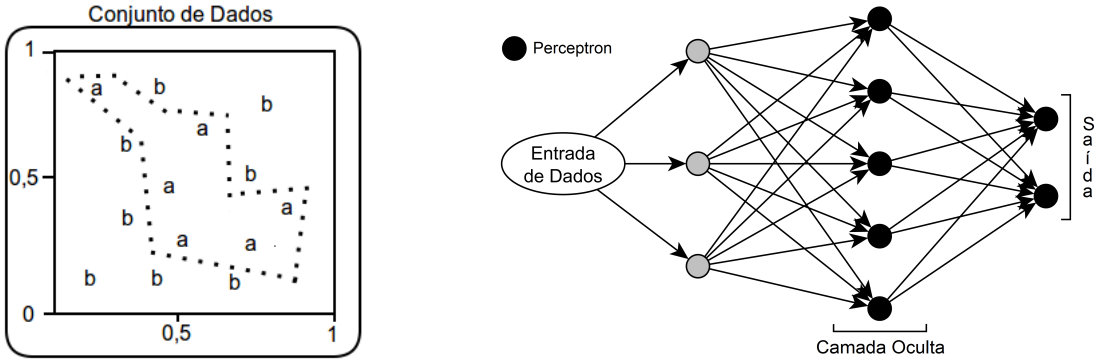
$$y = \varphi\left(\left(\sum_{i=0}^{n} x_i w_i\right) + b\right) \qquad (1.1)$$

$i \in \mathbb{Z}^+ = \{0...n\}$: Índice de vetor.
$\vec{x} \in \mathbb{R}$: Vetor de entrada de dados.
$\vec{w} \in \mathbb{R}$: Vetor de pesos.
$b \in \mathbb{R}$ : Parâmetro de flexibilização.
$\varphi(.)$: Função de ativação linear e/ou não linear.
$y \in \mathbb{R}$: Saída de informação da RNPM.

A Equação 1.1 recebe os dados de entrada na 1ª camada oculta da RNPM, e em seguida, os dados de entrada é resposta dessa equação, para a mesma equação, até a camada de saída.

Ainda mais, a Equação 1.1 conecta outras equações do mesmo tipo, com uma entrada de dados sendo um conjunto $C = \{a_1, a_2, ..., a_n, b_1, b_2..., b_n\}$ para treinamento, então temos:

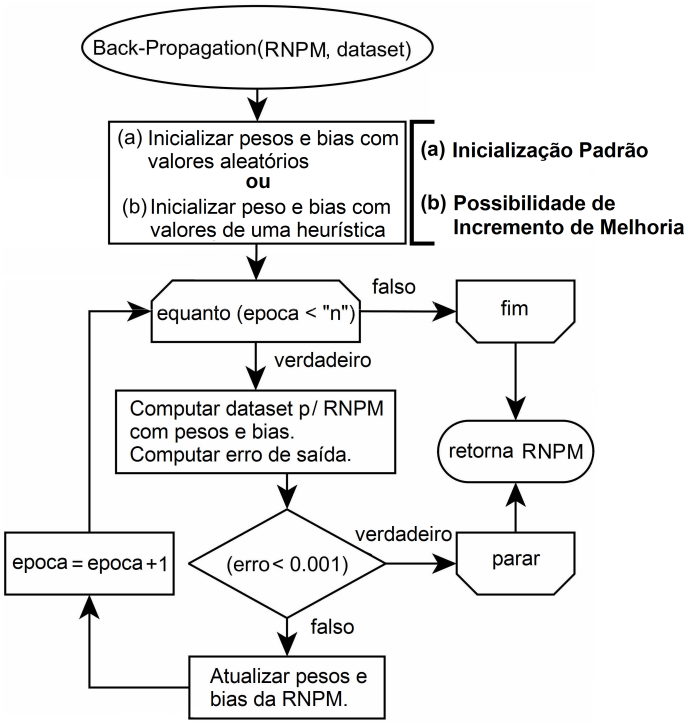
Figura 1: Descreve um Conjunto de Dados e uma Arquitetura de RNPM




Fonte: Adaptado de Faceli et al. (2011).

O conjunto de dados da Figura 1 é a entrada de dados de treinamento de RNPM pelo "Back-Propagation". O fluxograma do "Back-Propagation" é observado na Figura 2:

Figura 2: Descreve o fluxograma do "Back-Propagation"




Fonte: Adaptado de Rumelhart et al. (1986) e Faceli et al. (2011).

A Figura 2 descreve a inicialização de pesos e bias no Ponto **(a)** [aleatoriamente] e no Ponto **(b)** [heuristicamente]. O treinamento de RNPM é NP-Difícil em pior caso segundo Livni Roi e Shamir (2014) e Zhang et al. (2015). Por outro lado, o "Back-Propagation" é sensível a números aleatórios na inicialização de pesos e bias afirma Haykin (2001) e Samarasinghe (2006), o que pode levar a acurácias diferentes.

Conclui-se que uma heurística para calcular pesos e bias iniciais para o "Back-Propagation", pode otimizar a aproximação ou exatidão de máxima acurácia.

## 1.3 Hipóteses

São estabelecidas as hipóteses - $\mathcal{H}_0$ e $\mathcal{H}_1$ - para dedução lógica, por meio de comparação de resultados de experimentos computacionais e descobrimento de melhoria quantitativa de acurácia de algoritmo.

$\mathcal{H}_0$: Inicializar pesos e bias com números Aleatórios ou método Heurístico é equivalente em acurácia e tempo de treinamento.

$\mathcal{H}_1$: Inicializar pesos e bias com números Aleatórios ou método Heurístico é diferente em acurácia e tempo de treinamento.

A aceitação ou rejeição de $\mathcal{H}_0$ ou $\mathcal{H}_1$ é julgado por três testes de hipóteses conforme descritos em Hammer (2013), Nahm (2016), MathWorks (2016): (1) Teste de Friedman com (2) Pós-Teste de Tukey HSD "Post-hoc" e (3) Teste de Wicoxon-M-W. O resultado desta pesquisa é julgado por dedução lógica "se e somente se" usando a Regra Geral (Condição A e B):

$$Pesquisa(teste1,\ teste2,\ teste3) = \begin{cases} \text{A: Se } (\mathcal{H}_0 \text{ ou } \mathcal{H}_0 \text{ ou } \mathcal{H}_0) = \text{Equivalente} \\ \\ \text{B: Se } (\mathcal{H}_1 \text{ e } \mathcal{H}_1 \text{ e } \mathcal{H}_1) \quad = \text{Diferente} \end{cases} \quad \text{(Regra Geral)}$$

Testes(1,2,3): Testes de Hipóteses.

Os resultados dos experimentos computacionais serão julgados pela Regra Geral, para verificação de melhoria de acurácia de maneira universal para todos os estudos de casos.

## 1.4 Objetivo Geral

Desenvolver e aplicar um algoritmo Heurístico para calcular pesos e bias iniciais para o "Back-Propagation" treinar Rede Neural Perceptron Multicamadas (RNPM).

### 1.4.1 Objetivos Específicos

1. Desenvolver um algoritmo Heurístico para calcular pesos e bias iniciais para o ponto de partida do "Back-Propagation" treinar RNPM.
2. Aplicar, por meio de experimentos, o algoritmo Heurístico desenvolvido com o "Back-Propagation" em estudos de caso nas áreas de Saúde, Negócio e Botânica para o Reconhecimento de Padrões e Classificação de Dados.
3. Medir os resultados dos experimentos, para verificação de incremento ou decremento de acurácia, comparando os resultados de todos os algoritmos, na classificação de dados.
4. Avaliar os resultados dos experimentos com testes de hipóteses, para verificação de melhoria significativa de acurácia.
5. Comparar os resultados da heurística desenvolvida com outros resultados de algoritmos da literatura científica usados nos experimentos, e com outros trabalhos de pesquisa.

## 1.5 Motivação

Esta pesquisa é motivada pela gama de possibilidades de aplicação de RNPM em reconhecimento de padrões para classificar dados. O Quadro 3 descreve várias aplicações:

Quadro 3: Lista de Várias Aplicações de RNPM em Classificação de Dados

| Aplicação | Problemas do Mundo Real |
|---|---|
| 1 - CLASSIFICAÇÃO: classificar dados do mundo real, de acordo com o conhecimento aprendido em um conjunto de dados. | 1. Aplicações Diversas de Classificação:<br>- Imagem, Som, Sinal, Caractere, Perfil, etc;<br>2. Diagnósticos: Médico, Crédito, etc. |

Fonte: Adaptado de Braga et al. (2007), Silva et al. (2010) e Luger (2013).

O Quadro 3 apresenta várias aplicações de RNPM para auxiliar na tomada de decisão, que é inevitável em algum momento, nos levando ao posicionamento sobre alguma questão. Isso nos leva a buscar informações que auxiliem na decisão que precisa ser tomada. Em colaboração com a informação necessária, os algoritmos de aprendizagem de máquina podem auxiliar na decisão correta, com certo grau de precisão científica.

Isso motiva a pesquisa em redes neurais, em desenvolver algoritmos que possam ser programados para automatizar a captura da resolução do problemas. Nisso as redes neurais são adequadas, por exemplo, em reconhecer padrões para a classificação de dados. Daí a amplitude de aplicações a problemas do mundo real.

Em aplicações reais, consideremos que o diagnóstico cedo e preciso de doenças produz maiores chances de tratamento, minimiza custos, maximiza a possibilidade de cura e retarda a evolução da doença. Na área dos Negócios, uma decisão no tempo certo e comprecisão, pode maximizar os lucros e aumenta a credibilidade. Entretanto, a não decisão ou a falta de precisão pode levar a falência. Diagnosticar doenças, a tomada de decisão nos negócios, etc desafia o pesquisador de inteligência artificial.

A capacidade de tabular dados sobre a solução de determinado problema, observando o procedimento de um especialista ou soluções viáveis no decorrer da história de casos semelhantes, nos leva querer processar esses dados para a aprendizagem de máquina, pesquisar um modelo computacional adequado e automatizar a resposta para a resolução do problema.

Por fim, somos motivados à pesquisa para desenvolver melhores algoritmos para a computação de aprendizagem de máquina, criando soluções aproximativas ou ótimas, para auxiliar os maisdiversos ramos da sociedade a tomarem as decisões acertadas.

Conclui-se pela motivação do tema de pesquisa, sendo inspirador aplicar RNPM para generalizar o conhecimento, aprendendo com a resolução de problemas similares, e em seguida proporcionar aproximação ou exatidão de resposta, para direcionar a tomada de decisão, em problemas complexos de Reconhecimento de Padrões para classificar dados.

## 1.6 Trabalhos Relacionados

Outros trabalhos de pesquisa de maneiras diferentes de inicializar pesos e bias para o "Back-Propagation" são apresentados no Quadro 4.

Quadro 4: Trabalhos Relacionados sobre Inicializar Pesos e Bias para o "Back-Propagation"

| Trabalhos Relacionados | Maneiras de Inicializar Pesos e Bias | Aplicação em "Dataset" | Melhor Acurácia |
|---|---|---|---|
| Chesnokov (2008) | Números Aleatórios | Coração (Fibrilação Atrial Paroxística)[a] | 83,78% |
| Glorot e Bengio (2010) | $Pesos_{ij} \approx U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ Todos os Bias com zero. | MNIST (dígitos manuscritos)[b] | 97,79% ⊢→ 98,36% |
| Ciresan et al. (2012) | Distribuição Aleatória Uniforme entre [−0,05; 0,05] | MNIST (dígitos manuscritos)[b] | 98,19% ⊢→ 98,29% |
| Sodhi e Chandra (2014) | Pesos entre $[\frac{(2i\ \ 1)}{(H-1)}, \frac{(2i+1)}{(H-1)}]$ Bias $[\frac{2i}{(H-1)}\ \ ]$ | Seis Funções de Aproximação | 75,73% ⊢→ 99,83% |
| Zhao e al. (2015) | Números Aleatórios | Statlog (Risco de Crédito) [c] | 87,00% |
| Sundaram e Ramesh (2015) | Método Nyguen-Widrow entre $[-0.01; +0.7]$ | Conversão de Binário em ASCII | 96,67% |
| Murru e Rossini (2016) | Filtro de Kalman e Estatística Bayesiano | MNIST (dígitos manuscritos)[b] | 76,00% ⊢→ 91,70% |
| Pandey e Govind (2016) | Números Aleatórios | (Bupa, Diabetes, Yeast, Monk, Iris, Ionosphere, and Spambase)[c] | 73,46% ⊢→ 94,82% |
| Rodan et al. (2016) | Números Aleatórios | Spambase (E-mail com "spam"ou não "spam")[c] | 90,06% |
| Qiao et al. (2016) | Algorítmo MIWI p/ os pesos. Bias: Distribuição Uniforme Aleatória | (Iris)[c] | 99,001% |
| Ahachad et al. (2017) | Números Aleatórios entre $[-0{,}2; 0,2]$ Distribuídos Uniformemente. | (Abalone, Breast, Credit, Diabetes, German, Hepatitis, Image, etc.)[c] | 74,10% ⊢→ 97,50% |
| Arabasadi et al. (2017) | Inicializar pesos usando algoritmo Genérico. | Z-Alizadeh Sani in Tan et al. (2006) | 93,85% ⊢→ 97,00% |

Nota: [a] <https://physionet.org>. [b] <http://yann.lecun.com/exdb/mnist>. [c] <https://archive.ics.uci.edu/ml>.

O Quadro 4 apresenta outras pesquisas sobre inicialização de pesos e bias para o "Back-Propagation" treinar RNPM. São listados 12 (doze) trabalhos de pesquisa no período de 2008 a 2017, citando o estado da arte das maneiras diferentes de cálculo de inicializar pesos e bias. Entretanto, ainda destaca-se o uso predominante de números aleatórios.

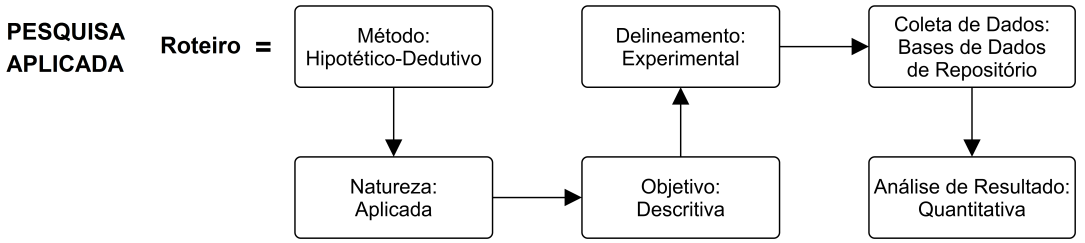
Os pesos e bias são atributos importantíssimos do perceptron, porque eles representam numericamente parte do conhecimento aprendido sobre a resolução do problema. Multiplicam os dados de entrada da RNPM, para reconhecer determinado padrão de comportamento.

Pela revisão da literatura científica e observação dos trabalhos relacionados, continua em aberto o campo de pesquisa sobre a melhor maneira de inicializar pesos e bias que resulte em máxima acurácia de RNPM treinada pelo "Back-Propagation".

## 1.7 Metodologia

Esta Pesquisa é do tipo Aplicada, faz uso de hipóteses segundo a visão de Popper (2007), para dedução lógica quantitativa comparando-as entre si, pela metodologia científico aplicada a Ciência da Computação conforme Wazlawick (2008), adotando um roteiro prático para a investigação científica explicada por Biagi (2012). A estrutura metodológica adotada segue:

Figura 3: Estrutura e Execução de Método Científico nesta Pesquisa




Fonte: Produção do Autor (2017).

A Figura 3 esclarece a estrutura metódica e científica usada para nortear esta pesquisa de forma a deixar clara a possibilidade de teste quanto a veracidade das hipóteses investigadas. Explica Popper (2007) que a aceitação de ciência depende da verificação das razões lógicas de seus resultados experimentais mediante a testabilidade de afirmação ou negação de hipóteses.

### 1.7.1 Pesquisa Quanto ao Método: Hipotético-dedutivo

Esta pesquisa deduz uma conclusão lógica e válida, de conhecimento geral para o específico através de estudos de casos, em termos numéricos a respeito de hipóteses por meio de experimento computacional, para alcançar o objetivo geral proposto, comprovando por dedução lógica a resposta para o problema estudado.

### 1.7.2 Pesquisa Quanto à Natureza: Aplicada

É feito seis estudos de caso entre as áreas de Saúde, Negócio e Botânica, com conjuntos de dados específicos. Os experimentos aplicam a heurística proposta com o "Back-Propagation" para treinar RNPM em classificar de dados. A aplicação nessas áreas tem por objetivo a busca por melhor acurácia em conjunto de dados específico para cada estudo de caso.

### 1.7.3 Pesquisa Quanto ao Objetivo: Descritiva com Proposta de Melhoria

É descrito a revisão bibliográfica conforme investigação em literatura científica sobre conceitos delimitadores e fundamentais para o desenvolvimento e aplicação da heurística proposta. Adescrição dos resultados experimentais usa a estatística Descritiva (média, desvio padrão, etc) explicada em Larson e Farber (2010) para composição de tabelas e gráficos.

A melhoria significativa de acurácia para aceitação ou rejeição de hipótese é julgada por três testes de hipóteses, conforme descrição de Nahm (2016), MathWorks (2016), e Hammer (2013): Teste de Friedman com Pós-Teste de Tukey e Teste de Wicoxon-M-W.

### 1.7.4 Pesquisa Quanto ao Delineamento: Experimental

O delineamento dos experimentos é feito observando Filho (2009, p. 1197-1211) sobre o Planejamento, Operação, Análise e Interpretação, Apresentação e Empacotamento de dados, e verificando Mcgeoch (2012, p.10) que orienta: planeje o experimento (formule uma pergunta, construa um ambiente de teste, especifique os exemplos para experimentação) e execute o experimento (realize testes, faça análise sobre os resultados e publique).

Para tanto, executa-se os experimentos com e sem o uso da heurística proposta com o "Back-Propagation" para o treinamento de RNPM. Os resultado dos experimentos são avaliados com testes de hipótese, comparando $\mathcal{H}_0$ com $\mathcal{H}_1$, para verificação de melhoria significativa.

### 1.7.5 Pesquisa Quanto à Coleta de Dados: Bases de Dados de Repositório

A coleta dados usou o repositório da Universidade do Canadá em Irvine, <http://archive.ics.uci.edu/ml>, nas áreas de Botânica, Negócio e Saúde referente aos "datasets":

- Área BOTÂNICA:
    1. Reconhecimento de Plantas "Iris".
- Área SAÚDE:
    1. Inflamação Aguda "Acute Inflammations";
    2. Câncer de Mama "Breast Cancer Wisconsin Diagnostic";
    3. Doença de Parkinson "Parkinsons".
- Área NEGÓCIO:
    1. Autenticação Bancária de Dinheiro Falso ou Verdadeiro "Banknote Autentication";
    2. Separação de Grãos de Trigo por Tipo "Seeds".

Esses "dataset's" correspondem aos estudos de caso de Reconhecimento de Padrão para Classificação de Dados e são usados nos experimentos computacionais.

### 1.7.6 Pesquisa Quanto à Análise de Dados: Quantitativa

Os dados são analisados sob a ótica quantitativa, usando os resultados numéricos para comparação de hipóteses, transformando-os em conhecimento científico.

A partir dos resultados dos experimentos é procurado diferenças de melhoria significativa de acurácia com o tempo de treinamento de RNPM, visando o objetivo geral dessa pesquisa.

Conclui-se pelo uso da análise Quantitativa, com a interpretação de dados através de comparações para aceitação ou rejeição das hipótese $H_0$ e $H_1$, sobre o uso da heurística proposta.

## 1.8 Estrutura do Trabalho

Esta pesquisa organiza-se pelas normas da Associação Brasileira de Normas Técnicas (ABNT) e está dividida em seis capítulos e referências.

O Capítulo 1 - Introdução - contextualiza e introduz o problema de pesquisa, aponta a justificativa à problemática, fixa as hipóteses, estabelece o objetivo geral e específicos, descreve a motivação, apresenta os trabalhos relacionados e especifica o uso da metodologia científica.

A revisão bibliográfica dividi-se nos Capítulos 2 e 3, observando a literatura científica referente ao tema de pesquisa, buscando teorias úteis e aplicáveis, sendo a base sólida para construção da investigação em foco.

O Capítulo 2 - Pesquisa Operacional: Otimização - revisa a grande área do tema que é a Otimização, explorando teorias sólidas e aplicáveis em passos metódicos para a resolução de problema do mundo real.

O Capítulo 3 - Rede Neural Artificia Multicamadas (RNPM) - concentra-se na área específico do tema, que é a Inteligência Computacional, investigando os conceitos fundamentais de RNPM, compreendendo sua arquitetura, construção e uso, de forma clara e precisa.

A produção científica acontece no Capítulos 4 (desenvolvimento da heurística proposta) e Capítulo 5 (aplicação da heurística proposta).

O Capítulo 4 - Proposta: Heurística para Calcular Pesos e Bias Iniciais (HeCI) - Descreve especificamente a teoria usada para desenvolvimento da heurística, fluxograma e algoritmo heurístico.

O Capítulo 5 - Experimentos Computacionais e Análise de Resultados - Especifica os materiais e métodos usados nos experimentos computacionais, apresenta todos os resultados por "dataset's" e conjunto universal de amostras de dados, lista o resultado de todos os testes de hipóteses e faz análise de resultados.

O Capítulo 6 - Considerações Finais - Faz a observação do cumprimento dos objetivos específicos e alcance do objeto geral, apresenta a contribuição para o campo de pesquisa e sugestão de trabalhos futuros.

Por último, as Referências, lista as literatura científica usada, fazendo referência a livros, artigos e sites, sendo a base teórica para a execução da pesquisa e matéria-prima para a contribuição científica.

# 2 Pesquisa Operacional: Otimização

A Pesquisa Operacional (PO) surge em ambiente militar, mas devido ao sucesso em aplicações e depois do encerramento da gerra, ingressa em contexto civil, expandido rapidamente, por causa dos motivos explicados por Hillier e Lieberman (2006, p. 2-3): aperfeiçoamento das técnicas de otimização, evolução dos computadores em processamento de milhões de cálculos matemáticos por segundo e ao impacto em melhoria da eficiência das organizações.

Relata Belfiore e Fávero (2013, p. 1-2) sobre o surgimento da PO:

> A Pesquisa Operacional surgiu na Inglaterra durante a Segunda Gerra Mundial (1939-1945) para a solução de problemas de natureza logística, tática, [...], quando um grupo de cientistas foi convocado para decidir sobre a utilização mais eficiente dos recursos militares limitados [...]. Os resultados positivos alcançados pelo grupo de cientistas ingleses fizera com que a Pesquisa Operacional fosse disseminada nos Estados Unidos.

Define Idem (2013, p. 2) em termos gerais, que a PO é a "utilização de um método científico (modelos matemáticos, estatísticos e algoritmos computacionais) para a tomada de decisões [...] atua [...] envolvendo áreas de engenharia de produção, matemática aplicada, ciência da computação e gestão de negócio".

Acrescenta Pizzolato e Gandolpho (2013, p. 2) que estudos de PO faz uso de Modelos Icônicos[1], Analógicos[2] e Simbólicos para tratar problemas complexos. Sendo preferível os Modelos Simbólicos por serem mais gerais e abstratos, porque "usam letras, números, e símbolos para representar variáveis e suas relações funcionais, tomam a forma de expressões matemáticas que refletem a estrutura do objeto representado".

Observando Hillier e Lieberman (2006), Belfiore e Fávero (2013) e Pizzolato e Gandolpho (2013) pode-se extrair seis fases para o uso da PO:

$1^o$ - Definição do problema de interesse e coleta de dados;

$2^o$ - Modelagem matemática para representação do problema;

$3^o$ - Processamento computacional para o modelo do problema;

$4^o$ - Teste e aprimoramento do modelo conforme necessário;

$5^o$ - Preparar o modelo para uso em ambiente de produção;

$6^o$ - Implementação do modelo em ambiente de produção.

Conclui-se que é necessário a observação a essas seis fases, para o sucesso da otimização em problemas do mundo real.

---

[1] Pizzolato e Gandolpho (2013, p. 2): Modelos Icônicos: referem-se a mudanças de escala. Tais modelos geralmente se parecem com o objeto que eles representam, exceto no tamanho. Exemplos: mapas, maquetes, [. . . ].

[2] Idem (2013, p. 2): Modelos Analógicos: usam um conjunto de propriedades para representar um outro conjunto. Exemplo: curvas de nível em um mapa como analógico da elevação, sistema hidráulico, [. . . ].

## 2.1 Definição do Problema de Interesse e Coleta de Dados

Na visão de Hillier e Lieberman (2006, p. 9) a "definição do problema é crucial, pois afeta enormemente quão relevantes serão as conclusões do estudo". Aborda-se o problema para extrair o objetivo a ser alcançado, deixando claro o que deve ser otimizado, identificando a função de maximização ou minimização, constantes e variáveis a serem computadas.

Dentro dessa perspectiva destaca Polya (2006) que uma abordagem inicial é o uso de perguntas, Quadro 5, para descrever matematicamente o problema para resolução com PO.

Quadro 5: Perguntas para Definição o Problema e Coleta de Dados

| Pergunta | Possível Resposta |
|---|---|
| Qual é a incógnita? | Identifica a grandeza a ser buscada para a solução do problema, de modo que respectivas incógnitas $(w_1, w_2, w_3)$ minimize ou maximize uma função matemática que representa a problemática. Por exemplo, Minimizar $f(w_1, w_2, w_3) = c_1w_1 + c_2w_2 + w_3$, onde $c_n$ é constante. |
| Quais são os dados? | Coletar os dados do problema e tabelá-los para análise eficiente inserindo-os na função objetivo para resolução. Esses dados geralmente caracterizam a entrada de dados, constantes e restrições que farão parte do modelo matemático. |
| Qual é a condicionante? | Rotular as relações entre as incógnitas que estabelece alguma condição que delimita ou direciona a solução do problema. Por exemplo, Minimizar $f(w_1, w_2, w_3) = c_1w_1 + c_2w_2 + c_3w_3$ poderia ter como condicionantes $w_1 > 1$ e $0 < (w_2 + w_3) < 10.000$, onde $c_n$ é constante. |

Fonte: Adaptado de Hillier e Lieberman (2006) e Polya (2006).

Respondendo as perguntas do Quadro 5, é alcançado o esclarecimento feito por Pizzolato e Gandolpho (2013, p. 8) que o problema deve ser conciso, mesmo que complexo, devendo ser modelado matemática de forma linear ou não linear, onde busca-se o ótimo global para o mínimo ou máximo de uma função objetivo. A formulação básica deve seguir o aspecto matemático:

$$Z = f(X_i, Y_j) \tag{2.1}$$

Sendo: $Z$ = valor de mínimo ou de máximo para a função objetiva.

$X_i$ = variáveis que podem ser controladas, ou variáveis de decisão.

$Y_j$ = parâmetros fora de controle ou variáveis independentes.

$f$ = relação funcional entre $Z, X_i$ e $Y_j$.

É indispensável a coleta de dados e a formulação correta do problema, tendo em mente o Quadro 5 e Equação 2.1, porque o sucesso de uso das fases seguintes da PO para otimização, dependem da correta execução dessa primeira fase.

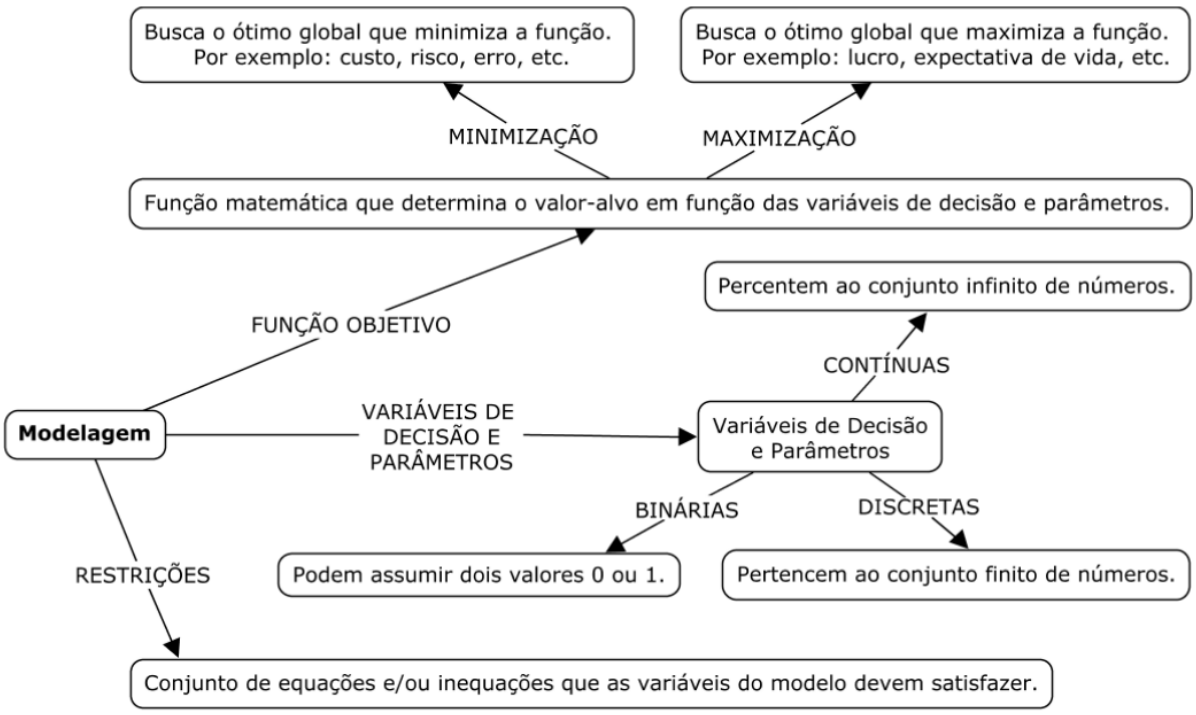
## 2.2 Modelagem Matemática para Representação do Problema

Conceitua Stewart (2013, p. 22) sobre modelo matemático que podemos atribuir a PO:

> Um modelo matemático é a descrição matemática (frequentemente por meio de uma função ou de uma equação) de um fenômeno do mundo real, como o tamanho de uma população, [...] a expectativa de vida de uma pessoa ao nascer ou o custo da redução de poluentes. O propósito desses modelos é entender o fenômeno e talvez fazer previsões sobre seu comportamento futuro.

Explica Belfiore e Fávero (2013) que a modelagem matemática de um problema real é influenciado por diversas variáveis envolvidas no processo de abstração, sendo um conjunto de três elementos principais: Função Objetivo, Variáveis de Decisão e Parâmetros, e Restrições.

Figura 4: Resumo de Estrutura Teórica para Modelagem de Função para Otimização




Minimizar $f(w_{11}, w_{12}, ..., w_{mn}) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} w_{ij}$ s.a: $\sum_{j=1}^{n} w_{ij} \leq a_i$

$f(.)$: Função Objetivo. Restrições: "s.a:". $c_{ij}, a_i$: Parâmetros. $w_{ij}$: Variáveis de decisão.

Fonte: Adaptado de Belfiore e Fávero (2013).

Entende-se que aplicando a estrutura teórica da Figura 4, o problema do mundo real torna-se o conjunto domínio; e o modelo matemático, o conjunto imagem; para o cômputo por algum algoritmo em busca de otimização, por isso explica Pizzolato e Gandolpho (2013) que é necessário fidelidade máxima do modelo ao problema do mundo real, para que o ótimo global resultante seja a melhor aproximação ou exatidão de resposta para o problema.

## 2.3 Processamento de Dados para o Modelo do Problema

Nesta fase, nas palavras de Hillier e Lieberman (2006) é executado algum algoritmo, para resolução do modelo matemático do problema, que busca uma solução ótima global, ou aproximada (sub-ótima) por causa do custo computacional (tempo e recursos de *hardware*) para o processamento de dados de soluções exatas, por atingirem um custo insustentável.

Na execução de algoritmo, Determinístico[3] ou Não Determinístico[4], verifica-s a Notação Assintótica, porque pode demandar tempo e custo com *hardware* indisponível em alguns casos, conforme entendimento feito em Nivio (2011): existem algoritmos exponencias que para um "$n$" (entrada de dados) grande consomem tempo não aceitável. A Tabela 1 descreve a Complexidade Assintótica:

Tabela 1: Classificação de Algoritmo em Complexidade Assintótica Big "O" (Grande)

| Entrada de Dados | Função de Custo de Tempo Computacional em Pior Caso | | | | |
|---|---|---|---|---|---|
| | $O(n)$ | $O(n^2)$ | $O(n^3)$ | $O(n^4)$ | $O(2^n)$ |
| 100 | 0,0000001 seg. | 0,00001 seg. | 0,001 seg. | 0,1 seg. | > 4 x $10^9$ séculos |
| 500 | 0,0000005 seg. | 0,00025 seg. | 0,125 seg. | 1:02 min:seg | > 327 x $10^{12}$ séculos |
| 900 | 0,0000009 seg. | 0,00081 seg. | 0,729 seg. | 10:56 min:seg | > 845 x $10^{12}$ séculos |

Fonte: Adaptado de Nivio (2011) e Cormen et al. (2012).

Nota: Cálculo em monoprocessador de 1 GHz ou $10^9$ instruções por segundo.

Observando a Tabela 1, esclarece Nivio (2011) que na escolha entre um algoritmo Determinístico ou não Determinístico, para resolução de um modelo matemático que demanda tempo proibitivo, deve ser escolhido um algoritmo não Determinístico de tempo polinomial, que encontre uma solução aproximada ao ótima global.

Por exemplo, áreas da Saúde, Telecomunicações, Transporte, Negócio, etc e Indústria Petrolífera, de Alimentos, Siderúrgica, etc, aplicam algoritmos de otimização em seus problemas de tomada de decisão, mas preferem um processamento de dados em tempo polinomial.

Finaliza-se com o entendimento feito em Belfiore e Fávero (2013) que a PO deve escolher algum algoritmo para orientar a tomada de decisão, mas considerando Cormen et al. (2012, p. 32): "a ordem de crescimento do tempo de execução de um algoritmo, dá uma caracterização da eficiência[...] e também nos permite comparar o desempenho relativo de algoritmos alternativos".

[3] Nivio (2011, p. 381) Algoritmo Determinístico: "tem a propriedade de que o resultado de cada operação é definido de forma única".

[4] Idem (p. 381, 2011) Algoritmo Não Determinístico: "é capaz de escolher dentre as várias alternativas possíveis a cada passo. Em outras palavras, [...] contêm operações cujo resultado não é unicamente definido ainda que limitado a um conjunto específico de possibilidades".

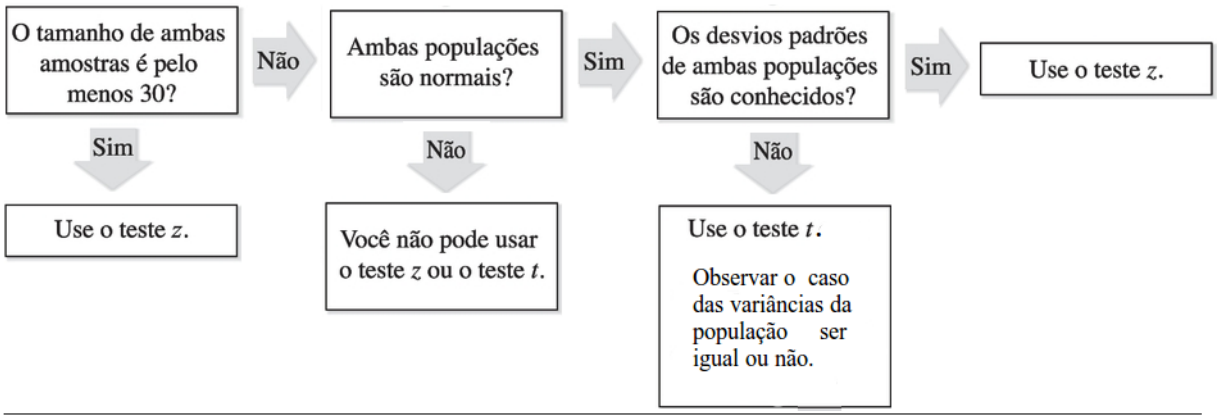
## 2.4 Aprimoramento do Modelo e Validação

Orienta Hillier e Lieberman (2006) que o aprimoramento do modelo e validação ocorre na re-examinação da definição do problema, revisão das expressões matemáticas e variação de valores dos parâmetros para averiguação da saída; contudo a validação precisa de rigor na comparação de hipóteses entre dados históricos ou modelo anterior e o novo modelo.

A verificação quanto a relevância em melhoria da nova abordagem de otimização é passível de investigação estatística para validação, conforme as palavras de Belfiore e Fávero (2013, p. 7) "um modelo pode ser considerado válido se conseguir representar ou prever, com precisão aceitável, o comportamento do sistema em estudo".

Nisso esclarece Larson e Farber (2010, p. 292) que "um teste de hipótese é um processo que usa estatísticas amostrais para testar a afirmação [...]. Então [toma-se] uma dessas duas decisões: 1. rejeitar a hipótese nula ou, 2. falha ao rejeitar a hipótese nula", onde pode ser usado, entre outros, os testes de hipótese: Testes "t" de Student ou Teste "Z", observando os conceitos de população, amostra, desvio padrão e distribuição normal[5].

Figura 5: Verificação de uso de Teste de Hipótese




Fonte: Adaptado de Larson e Farber (2010, p. 363).

Nota: [Amostra ≥ 30 use Teste "Z"] e para [Amostra < 30 use Teste "t"].

A Figura 5 estrutura as perguntas necessárias para responder a qual tipo de teste de hipótese usar. A escolha correta no teste de hipótese implica diretamente em decisões corretas para aceitação ou rejeição de melhoria de resultados experimentais dos algoritmos propostos para otimização.

Por fim fica claro nas palavras de Larson e Farber (2010, p. 292) que "um teste de hipótese é um processo que usa estatísticas amostrais para testar a afirmação". Aplicando a Figura 5 chega-se a qual teste de hipótese usar, garantindo a comprovação sob determinada condição que de fato houve relevante melhoria do novo processo sob o anterior.

[5] Larson e Farber (2010, p. 193) Distribuição Normal: é uma distribuição de probabilidade contínua para uma variável aleatória $x$ (os dados plotados apresentam-se em forma de sino, unimodais e simétricos em relação a média).

## 2.5 Preparar o Modelo para Uso em Ambiente de Produção

Dentro desta fase aponta Hillier e Lieberman (2006, p. 19): “é instalar um sistema bem-documentado para aplicação do modelo”. Esse documento deve descreve detalhadamente o modelo matemático, o algoritmo usado na resolução do problema, os resultados experimentais, o teste de hipótese e a orientação para uso do sistema em ambiente de produção.

A preparação para implantação de um novo modelo de processamento de dados para otimização, em contexto corporativo, é classificado por Sommerville (2011, p.502) de "Mudança de Processo"[6], devendo ser preparado uma análise e planejamento para implantação. Um Documento de Implantação (DI) deve ser elaborado nos seguintes moldes:

Quadro 6: Questões que Orientam a Elaboração do Documento de Implantação (DI)

| Tópicos a Preparar | Questões que o DI deve Responder |
|---|---|
| Adoção e Padronização | O DI está elaborado para ser padrão em toda a organização? Se o processo não for padronizado, então as mudanças não serão transferíveis para uso por toda a corporação. |
| Prática de Engenharia de Software | As boas práticas de engenharia de *software* (padronização de código, reuso, usabilidade etc) estão incluídas no DI? A falta dessas práticas onera a manutenção do sistema. |
| Restrições Organizacionais | Quais são as restrições organizacionais que afetam a mudança de DI ou sua execução? Restrições organizacionais podem dificultar grandemente a implantação. |
| Comunicações de Software e com Pessoas | Como as comunicações são gerenciadas no DI? O problema de comunicação é uma questão importante, e muitas vezes, os gargalos de comunicação são motivo para atrasos de implantação. |
| Capacitação de Recursos Humanos | Como as pessoas aprenderão a usar o novo sistema? A capacitação de pessoas é indispensável ao sucesso da implantação. |
| Suporte, Compatibilidade e Ferramentas | Quais aspectos do novo sistema são ou não compatíveis com os *softwares* da corporação? Na falta de compatibilidade, como criar compatibilidade ou instalar ferramentas de terceiros? |

Fonte: Adaptado de Sommerville (2011).

Por fim a abordagem metódica do Quadro 6 norteia a criação do DI para uso do sistema em ambiente de produção, acoplando otimização a corporação em melhoria ao Sistema de Apoio a Decisão (SAD), que de acordo com Laudon e Laudon (2014), depende mais de modelos matemáticos para a execução de algoritmos, partindo de condições conhecidas ou hipóteses, permitindo alcançar eficiência de decisão com os resultados de ótimo global ou aproximado.

[6] Sommerville (2011, p. 502) Mudança de Processo: "Introdução de novas práticas, métodos ou ferramentas; mudando a ordem das atividades do processo; introduzindo ou removendo os entregáveis do processo; melhorando as comunicações; ou introduzindo novos papéis e responsabilidades".

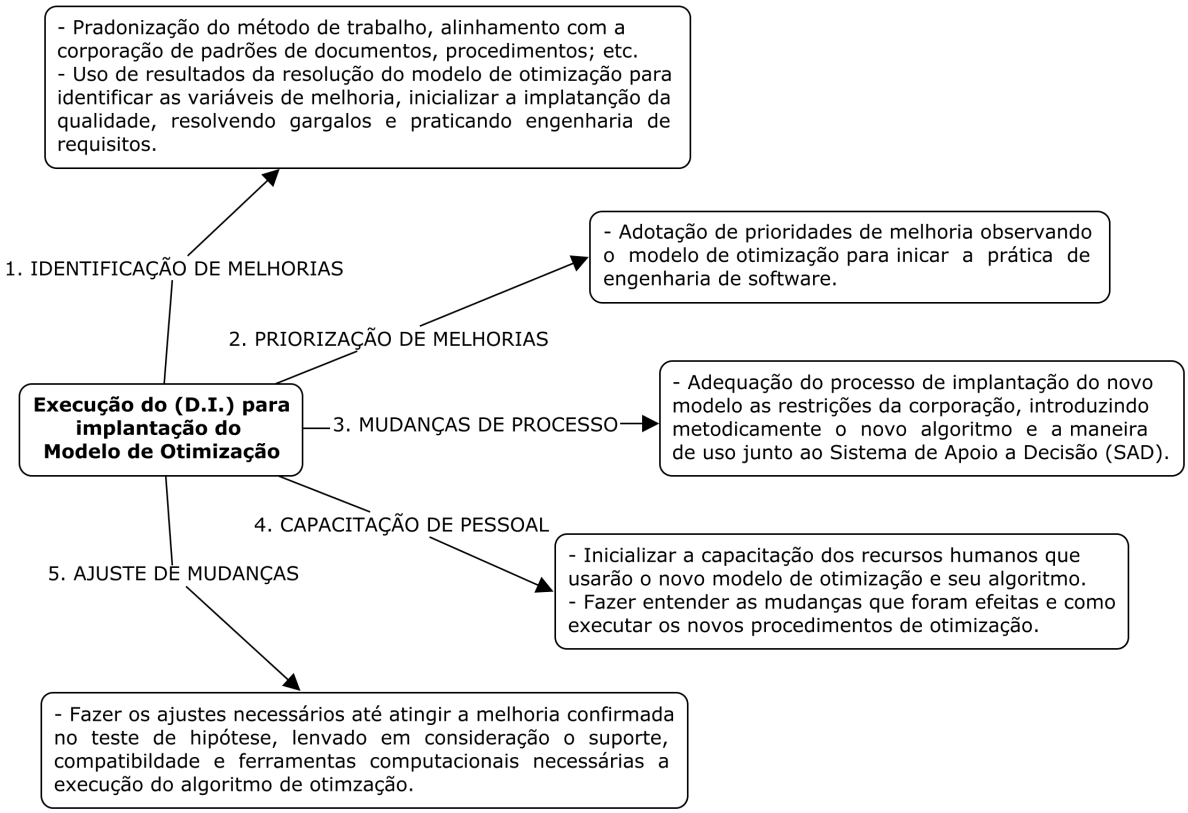
## 2.6 Implementação do Modelo em Ambiente de Produção

Decerto posiciona-se Hillier e Lieberman (2006, p. 20) sobre esta fase:

> É uma fase crítica, pois é aqui, e somente aqui, que os frutos do estudo são colhidos. Portanto, é importante para a equipe de PO [...] garantir que as soluções do modelo se traduzam precisamente em um procedimento operacional [...]. O sucesso [...] depende muito do suporte da alta gerência como o da gerência operacional. É muito mais provável que a equipe de PO ganhe esse apoio se ela tiver mantido a gerência bem informada e tiver encorajado a orientação ativa da gerência ao longo do curso do estudo.

De acordo com Sommerville (2011) "as mudanças de processo envolvem fazer modificações [...] que podem ser descritas em cinco estágios principais" para a total implementação:

Figura 6: Cinco Passos para Implementação de Novo Modelo de Otimização




Fonte: Adaptado de Sommerville (2011, p. 502).

Finalmente, executando os cinco passos da Figura 6, a PO chega ao sucesso de uso do novo modelo de otimização, alcançando o entendimento feito em Laudon e Laudon (2014), que diz ser a implementação de técnicas de otimização a maneira de produzir informação com precisão, abrangência, imparcialidade, velocidade, coerência, etc apoiando com robustez a decisão exata ou aproximada do ótima global, em problemas complexos junto ao Sistema de Apoio a Decisão (SAD), dentro do ambiente de produção.

# 3 Rede Neural Perceptron Multicamadas

A Rede Neural Artificia Multicamadas (RNPM) é definida conforme Quadro 7:

Quadro 7: Definições para RNPM

| Autor | Definição |
|---|---|
| Haykin (2001, p. 183) | Um conjunto de unidades sensoriais (nós de fonte) que constituem a camada de entrada, uma ou mais camadas ocultas de nós computacionais e uma camada de saída. |
| Braga et al. (2007, p. 67) | Estruturas com características não-lineares para a resolução de problemas de maior complexidade [...] composta por neurônios com funções de ativação sigmoidais nas camadas intermediárias. |
| Faceli et al. (2011, p. 120) | São sistemas computacionais distribuídos compostos de unidades de processamento simples, densamente interconectadas. Essas unidades, conhecidas como neurônios artificiais computam funções matemáticas. |
| Russel e Norvig (2013, p. 635) | São unidades conectadas por ligações direcionadas. Uma ligação da unidade $i$ para a unidade $j$ serve para propagar a ativação $a_i$ de $i$ para j. Cada ligação também tem um peso numérico $w_{ij}$ associado, que determina a força e o sinal de conexão. |

Fonte: Produção do Autor (2017).

A definição de RNPM conforme o Quadro 7 acontece com a evolução científica de Rede Neural para Rede Neural Perceptron Multicamdas, devido o avanço da aprendizagem de máquina no decorrer dos anos.

Entende-se por RNPM, a capacidade de processar dados de maneira não linear, para aprender números capazes de representar o conhecimento extraído sobre a solução de algum problema do mundo real.

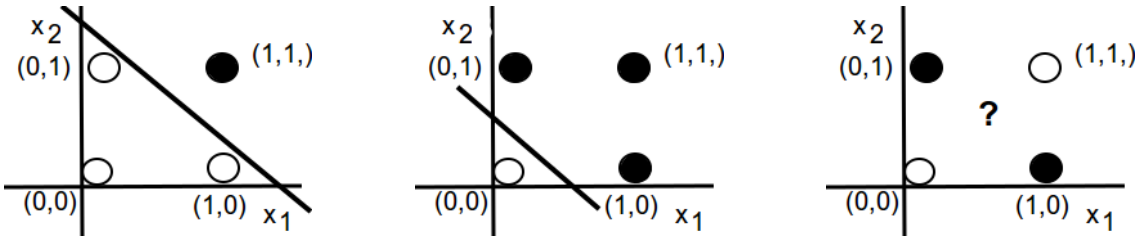
Nesse aprendizado percebe-se a capacidade de abstração numérica sobre alguma resolução de problema feita pela homem, onde agora a máquina aprende o procedimento e aplica a outros problemas similares, alcançando de forma aproximada ou exata a mesma qualidade de resposta de um profissional específico da área domínio do problema.

Com essa capacidade de abstração de aprendizagem da RNPM, percebe-se grande capacidade de aplicação aos mais diversos problemas do mundo real. Pois agora temos a capacidade computacional de programar a máquina para aprender e resolver problemas com base nesse aprendizado.

Para ilustrar uma RNPM, podemos compara-la a várias mentes que aprendem de forma em sequência de camadas, da primeira até a última, e opinam sobre determinada questão, a resposta. Por exemplo, [Dados] -> [Cérebro 1] -> [Cérebro 2] -> [Resposta].

As RNPM nascem da necessidade de resolver problemas não-lineares, onde apenas uma única camada de perceptrons não conseguem resolver, esclarece Luger (2013, p. 394) apud Minsky e Papert (1969) que "os perceptrons não podiam resolver certa classe difício de problemas, na qual os pontos de dados não são linearmente separáveis". Isso fica claro com a resolução de funções Booleanas no plano cartesiano por uma RNA de camada única:

Figura 7: Resolução de Funções Booleanas para Rede Neural Artificia de Camada Única




$x_1$ "and" $x_2$

| $x_1$ | $x_2$ | Saída |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Sistema Linear Resolvido

$x_1$ "or" $x_2$

| $x_1$ | $x_2$ | Saída |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Sistema Linear Resolvido

$x_1$ "xor" $x_2$

| $x_1$ | $x_2$ | Saída |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Sistema Não Linear Não Resolvido

Fonte: Adaptado de Russel e Norvig (2013) e Luger (2013).

A Figura 7 exibe as funções Booleanas "and, or, xor", onde "xor" não é resolvido por ser não linear, todavia segundo Russel e Norvig (2013, p. 662) apud Minsky e Papert (1969) foi demonstrando um teorema provando que uma RNA de camada única (camada de saída), somente representa conceitos linearmente separáveis, sendo incapaz de resolver problemas não lineares, e outrossim, notou a falta de algoritmo de aprendizagem para o treinamento de RNPM.

Por isso, surge a RNPM com 2 (duas) e 1 (uma) camada oculta, que nas palavras de Cybenko (1989), conforme demonstração através de teorema, mostrou que 2 (duas) camadas ocultas são suficiente para representar qualquer Função[1] matemática, e com 01 (uma) camada oculta é suficiente para representar qualquer função Contínua[2], com parâmetros de entrada de "$n$" e variáveis $\in \mathbb{R}$.

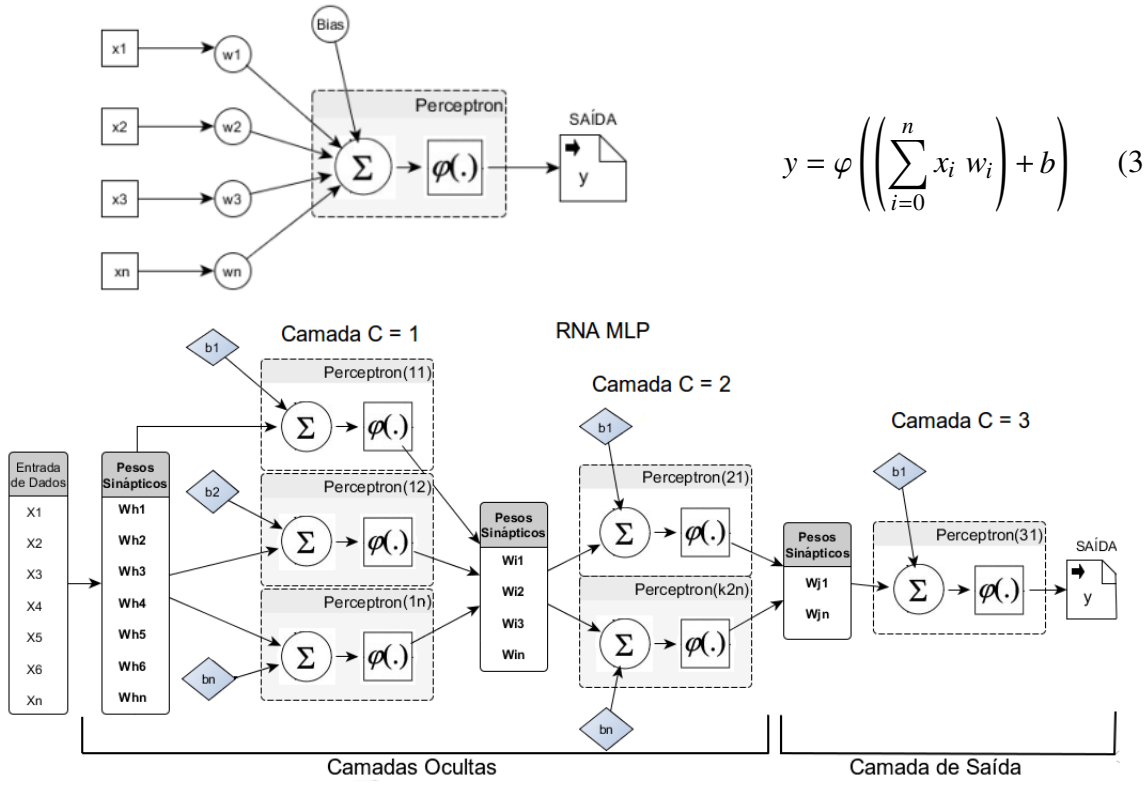
[1] Stewart (2013, p. 10) Para o Cálculo, "uma função $f$ é uma lei que associa, a cada elemento $x$ em um conjunto $\mathbb{D}$ [Domínio], exatamente um elemento, chamado $f(x)$, em um conjunto $\mathbb{E}$ [Imagem], ou seja, $\{(x, f(x)) \mid x \in \mathbb{D}\}$, onde $\mathbb{D} \xrightarrow{f} \mathbb{E}$".

[2] Idem (p. 109): "Uma função $f$ é contínua em um número $a$ se $\lim_{x \to a} f(x) = f(a)$, ou seja, [...] requer três condições verdadeiras para a continuidade de $f$ em $a$:

(1) $f(a)$ está definida
(2) $\lim_{x \to a} f(x)$ existe;
(3) $\lim_{x \to a} f(x) = f(a)$".

Explica Haykin (2001) que caso a modelagem não-linear se destaque em descrever o problema, então é preciso adicionar camadas ocultas de perceptrons, conforme Figura 8 e modelo matemático 3.2, por isso a evolução de RNA para RNPM, sendo agora capaz de resolver problemas não-lineares, por exemplo, classificação  aproximação, previsão, etc.

Figura 8: Arquitetura de um Neurônio Matemático ou Perceptron e de uma RNPM




Fonte: Adaptado de Silva et al. (2010), Faceli et al. (2011) e Russel e Norvig (2013).

Com base na Figura 8, descrevemos o modelo matemático de uma RNPM:

$$y(\vec{x}) = \left[ c3 = \varphi\left( \sum_{j=0}^{n} \vec{w}_j \left[ c2 = \varphi\left( \sum_{i=0}^{n} \vec{w}_i \left[ c1 = \varphi\left( \left( \sum_{h=0}^{n} \vec{x}_h \vec{w}_h \right) + b \right) \right]_{(c1,\, p)} + b \right) \right]_{(c2,\, p)} + b \right) \right]_{(c3,\, p)} \tag{3.2}$$

$h,\ i,\ j \in \mathbb{Z}^+ = \{0 \ldots n\}$: Índice de vetor, sendo "$h$" para $c1$, "$i$" para $c2$ e "$j$" para $c3$.

$c1,\ c2,\ c3 \in \mathbb{Z}^+ = \{0 \ldots n\}$: Número de camadas da rede neural.

$p \in \mathbb{Z}^+ = \{0 \ldots n\}$: Número de perceptron em cada camada da rede neural.

$\vec{x} \in \mathbb{R}$: Vetor de entrada de dados.

$\vec{w} \in \mathbb{R}$: Vetor de pesos para os perceptrons.

$b \in \mathbb{R}$ : Parâmetro de flexibilização para a soma do produto dos vetores $\vec{x}$ e $\vec{w}$.

$\left( \sum_{i=0}^{n} \vec{x}_i \vec{w}_i \right) + b$: Integrador de dados de entrada, pesos e bias.

$\varphi(.)$: Função de ativação linear e/ou não linear.

$y \in \mathbb{R}$: Saída de informação da RNPM.

## 3.1 Características de uma RNPM

Claramente observa-se em Haykin (2001) e Russel e Norvig (2013) que a RNPM extrai seu poder computacional através de duas abordagens principais:

1. Inicialmente pela sua estrutura maciçamente paralelamente distribuída com a conectividade entre vários neurônios, onde os dados de entrada flui até a saída, passando por todos os neurônios e todas as camadas;

2. Por último, pela sua capacidade de aprender o comportamento de um conjunto de dados e generalizar os parâmetros (arquitetura da RNPM, pesos e bias) aprendidos a outras situações distantes, mas pertencente ao mesmo domínio de dados.

Destaca Haykin (2001) que a generalização é em geral uma vantagem, pelo fato da RNPM produzir saídas adequadas para entradas de dados não presentes a época de treinamento ou aprendizagem. Essas duas abordagens principais de processamento de dados tornam possível a resolução de problemas complexos do mundo real.

Confima Luger (2013), explicando que um algoritmo de aprendizagem de máquina e uma RNPM são procedimentos de treinamento, armazenamento e generalização de conhecimento sobre alguma tarefa, em vez de programa-la diretamente. Esses procedimentos projetados adequadamente podem capturar e executar tarefas, tais como, classificação e previsão.

São características de uma RNPM:

Quadro 8: RNPM Proporciona Ótimo Tratamento sobre Dados de Entrada

| Característica | Descrição |
|---|---|
| 1. Não-Linearidade. | Um neurônio artificia pode conter uma função linear ou não-linear, sendo a não-linearidade distribuída por toda a rede. Característica muito importante, particularmente se a entrada de dados for do tipo não linear, por exemplo, designer de um objeto, sinal de voz, movimento complexo de imagens, etc. |
| 2. Mapeamento de entrada e saída. | Corresponde a apresentar a RNPM um exemplo ao acaso de um conjunto de dados e a resposta correta, então é modificado os pesos e bias até minimizar a diferença entre entrada e saída, mapeando numericamente a aprendizagem desejada. |
| 3. Adaptabilidade. | A RNPM após treinada continua capaz de adaptar seus pesos e bias a modificações de contexto do problema tratado. Podem ser treinanadas novamente para atualização, inclusive em tempo real, garantindo uma abordagem robusta e dinâmica a um contexto de problema não-estacionário. |

Fonte: Adaptado de Haykin (2001), Braga et al. (2007) e Silva et al. (2010).

Quadro 9: RNPM Fornece Credibilidade Informacional e Contextual de Resposta

| Característica | Descrição |
|---|---|
| 1. Resposta a Evidências. | Em problemas de classificação de padrões, uma RNPM pode ser implementada para fornecer resposta sobre o problema e também sobre a confiança na decisão tomada. Essa última informação pode ser utilizada para identifica padrões ambíguos, que podem existir nos dados de treinamento. Por exemplo, a classificação de 0 para o Grupo $A$, 1 para o Grupo $B$, sendo a proximidade da resposta entre 0 $e$ 1, o grau de certeza da resposta. |
| 2. Informação Contextual. | O conhecimento aprendido é representado pelos pesos e bias, função de ativação, número de camadas da rede e número de neurônios em cada camada. Cada neurônio matemático é afetado pela entrada de dados, pesos e bias, que consequentemente, representam a informação contextual do problema e resolução. |

Fonte: Adaptado de Haykin (2001), Braga et al. (2007) e Silva et al. (2010).

Por fim, em especificidade de características, as RNPM possui uniformidade de análise, projeto e desenvolvimento para fazer engenharia de software.

Quadro 10: RNPM é Uniforme em Análise, Projeto e Desenvolvimento

| Característica | Descrição |
|---|---|
| 1. Uniformidade de Análise, Projeto e Desenvolvimento. | A RNPM contêm universalidade de processadores, ou seja, o neurônio matemático. Esse neurônio é usado em todas as aplicação de rede Neurais, padronizando a análise, projeto e o desenvolvimento de software, sendo observado pelo menos três benefícios: (1) Os neurônios são componentes que podem ser padronizados e feito reuso; (2) A uniformidade em geral da arquitetura torna possível criar biblioteca de software, compartilhar teorias e algoritmos de aprendizagem em diferentes aplicações; (3) Pode ser criada com a integração de procedimentos ou funções de biblioteca de software, ampliando a capacidade de reuso de código. |

Fonte: Adaptado de Haykin (2001), Braga et al. (2007) e Silva et al. (2010).

Em virtude das características mencionadas nos Quadros 8, 9 e 10, enfatiza Faceli et al. (2011) que RNPM são aplicáveis a solução de muitos tipos de problemas do mundo real, por exemplo, tarefas de classificação previsão, aproximação, etc. Essas características fazem da RNPM um ótimo método com bom desempenho, quando usadas em aplicações reais para a solução de problemas lineares e não lineares.

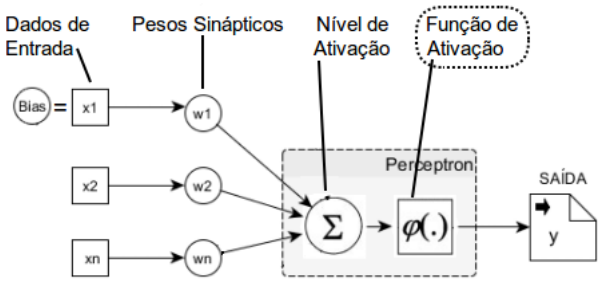
## 3.2 Funções de Ativação da RNPM

Conceitua Haykin (2001, 38): "a função de ativação, representada por $\varphi(v)$, define a saída de um neurônio [ou perceptron] em termos do campo local induzido $v$, ou seja":

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq valor \\ 0, & \text{se } v < valor \end{cases} \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ Resposta do Perceptron.} \tag{3.3}$$

O parâmetro "$v$" da função $\varphi(v)$, na Equação 3.3, é calculado somando o produto da multiplicação dos dados de entrada com os pesos, e adicionando um bias ou igualando o 1º peso "w1" a 1. Tudo isso correspondendo a "$x$" dados de entrada, conforme Equação 3.4:

Figura 9: Função de Ativação dentro do Neurônio Matemático ou Perceptron




$$y = \varphi\left(\sum_{i=0}^{n} x_i w_i\right) \tag{3.4}$$

Fonte: Adaptado de Luger (2013).

Define Luger (2013, p. 377) que a função de ativação ou função de limiar[3], na Equação 3.4, é uma "função que calcula o estado final ou de saída, do neurônio, determinando o quanto o nível de ativação do neurônio está baixo ou acima de um valor de limiar [...] produzindo o estado ligado/desligado dos neurônios reais".

A função de ativação é uma parte fixa do perceptron, Figura 9, e segundo Luger (2013) deve ser uma função contínua e diferenciável, de preferência exponencial, por causa da resolução de problemas do mundo real possuírem dados com atributos não linearmente separáveis, diversificado entre si e complexos.

Ensina Stewart (2013) que uma Função Contínua $f$ é um número $a$ se $\lim_{x \to a} f(x) = f(a)$, ou seja, $f(x)$ tende a $f(a)$ quando $x$ tende [ao ponto] $a$. De fato, a alteração em $f(x)$ pode ser mantida tão pequena quanto desejarmos, mantendo-se uma variação em "x" suficientemente pequena. Para uma função ser continua é necessários satisfazer a três condições:

Quadro 11: Condições para Existência da Função Contínua

| | | |
|---|---|---|
| 1. $f(a)$ está definida | 2. $\lim_{x \to a} f(x)$ existe. | 3. $\lim_{x \to a} f(x) = f(a)$. |

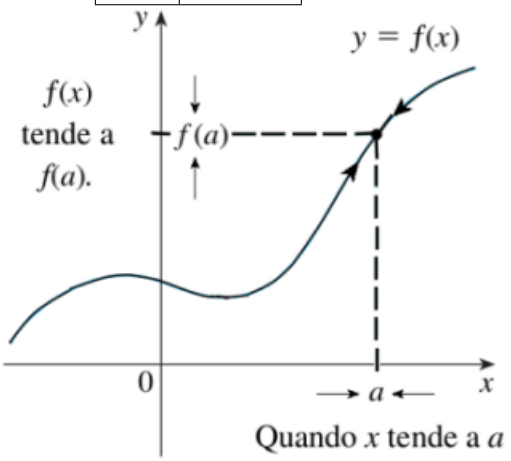
Fonte: Stewart (2013, p. 109).

Demonstrando graficamente a função Contínua, segundo Stewart (2013), temos:

[3] "Limiar: Ponto que constitui um limite, geralmente inicial.", in Dicionário Priberam da Língua Portuguesa, 2008-2016, <http://www.priberam.pt/dlpo/limiar> [consultado em 06-03-2016].

Figura 10: Gráfico de Verificação da Existência da Função Contínua

| Coordenadas | |
|---|---|
| x | y = f(x) |
| 0,25 | 3,5 |
| 0,50 | 4,0 |
| 0,75 | 4,5 |
| 0,99 | 4,98 |




$$Ponto\ a = 1,\ f(x) = \begin{cases} 2x+3, & \text{se } x \leq 1 \\ 5-x, & \text{se } x > 1 \end{cases} \tag{3.5}$$

Testabilidade à Condição do Quadro 11-1:
f($a$) = 2x + 3 $\Rightarrow$ f($a$) = 5 $\therefore$ Satisfaz.

Testabilidade à Condição do Quadro 11-2:
$\lim_{x \to a^-}$ f(x) = 2x + 3$\Rightarrow$f(x) = 5
$\lim_{x \to a^+}$ f(x) = 6 - x$\Rightarrow$f(x) = 5
$\lim_{x \to 1}$ f(x) = 5 $\therefore$ Satisfaz.

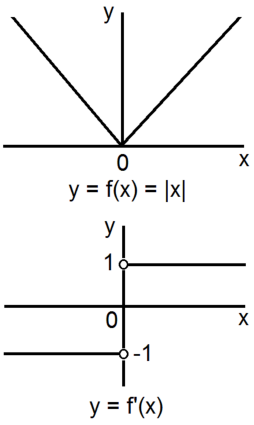
Testabilidade à Condição do Quadro 11-3:
f(x) = f($a$) $\Rightarrow$ 5 = 5 $\therefore$ Satisfaz.

Fonte: Adaptado de Stewart (2013, p. 109).

O gráfico e as coordenadas da Figura 10 representam a aplicação do conceito de continuidade, ou seja, quando $x$ se aproxima de 1 então $y = f(x)$ se aproxima de 5, portanto $f(x)$ tende a $f(a)$. Considerando a lei de formação de $f(x)$, Função 3.5, e o valor de $a = 1$, executa-se a testabilidade das condições previstas no Quadro 11, para verifica se $f(x)$ é contínua. Sendo satisfeita as três condições, aceita-se como Contínua a Função 3.5.

Outrossim, Stewart (2013, p. 143) explica que Função Diferenciável " é uma função $f$ derivável ou diferenciável em $a$, se $f'(a)$ existir", com $a \neq 0$.

Figura 11: Gráfico de uma Função Diferenciável ou Derivável




$$f'(x) = \begin{cases} 1, & \text{se } x > 0, \text{ então } |x| = x \\ -1, & \text{se } x < 0, \text{ então } |x| = -x \end{cases} \tag{3.6}$$

Se $x > 0$, então $|x|$ = x e podemos escolher $h$ suficientemente pequeno para que $x + h > 0$ e portanto $|x + h|$ = x + h.
Consequentemente, para $x > 0$ temos: $\lim_{h \to 0^+} 1 = 1$

Analogicamente, para $x < 0$, temos $|x|$ = -x e podemos escolher $h$ suficientemente pequeno para $x + h < 0$ e portanto $|x + h|$ = -(x + h).
Consequentemente, para $x < 0$ temos: $\lim_{h \to 0^-} (-1) = (-1)$

Fonte: Stewart (2013, p. 144).

O gráfico da Figura 11 exibe a diferenciabilidade ou derivação de uma função $f'(x)$ em

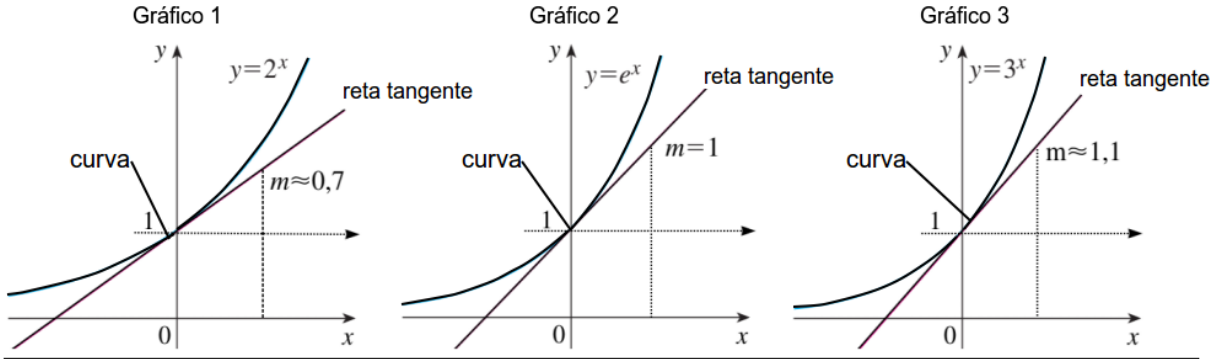
um ponto $a \neq 0$, pelas condições de $a$ está defini o, ou seja, $a \in R^*$, os limites laterais tanto pela direita, $\lim_{h \to 0^+}$, como pela esquerda, $\lim_{h \to 0^-}$, existirem e serem iguais pelo cálculo do módulo absoluto. Então sendo a função diferenciável pode-se encontrar um $h$ que tende a 0 satisfazendo a proximidade.

Explica Stewart (2013) que se uma função é Contínua e Diferenciável em um ponto, então "um aspecto importante da resolução do problema é tentar encontrar uma conexão entre o dado e o desconhecido". Aplicando as RNPM deve-se fazer a indução ou generalização do conhecimento, fixando a entrada de dados e encontrando os pesos e bias desconhecidos. Com os peso e bias encontrados, a saída da RNPM tem um erro "h" com tendência a 0.

Tratando da preferência, segundo Luger (2013), pela função exponencial para compor o perceptron da RNPM, afi ma Stewart (2013, p. 48-52):

> Função Exponencial em geral tem a forma $f(x) = a^x$, onde $a$ é uma constante positiva e $x$ é expoente. [...] as fórmulas do cálculo fica muito simplificada quando escolhemos como base $a$ aquela para a qual resulta uma reta tangente $y = a^x$ em (0, 1) com uma inclinação de exatamente 1. Existe um número assim e ele é denotado pelo caractere $e$ [constante de Euler, ($e \approx 2,71828$)]. Podemos chamar a função $f(x) = e^x$ de função exponencial natural.

Figura 12: Gráfico do Comportamento do Coeficiente Angular "m" da Função Exponencial




Fonte: Adaptado de Stewart (2013, p. 52).

O gráfico da Figura 12-2, em $y = e^x$, comparando com os outros gráficos Figura 12-1 e 12-3, exibe a afirmação de Stewart (2013) sobre função exponencial natural, onde a reta tangente nos pontos (0, 1) tem inclinação de exatamente 1 em relação ao eixo $x$, trazendo uma simplificação para o cálculo: $f(x) = e^x$ tem a propriedade de ser a própria derivada, com isso facilitando o processamento de dados.

Finaliza-se com o entendimento feito em Haykin (2001) e Luger (2013), que a função de ativação deve ter comportamento estritamente crescente, com balanceamento adequado entre comportamento linear e não linear, ser do tipo contínua e diferenciável, e preferencialmente exponencial (a RNPM usa todas essas características, mas principalmente o comportamento não linear). Nessas características é dado ênfase as funções Logística e Tangente Hiperbólica.

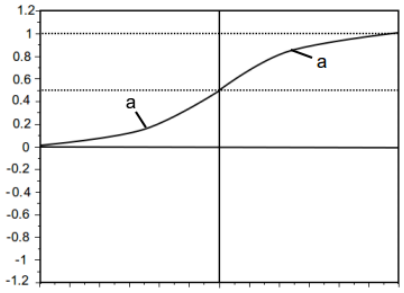
### 3.2.1 Função Logística

Lei de formação conforme Haykin (2001):

$$\text{Logística}(x) = \frac{1}{1 + \exp(-(ax))} \tag{3.7}$$

$a$: parâmetro (pesos) de inclinação da função, com $a \in \mathbb{R}$.
$x$: variável independente (entrada de dados), $-\infty < x < +\infty$.
$exp(expoente)$: Função que calcula a constante de Euler, $e \approx 2,718$, elevada a um expoente.

Figura 13: Gráfico da Função conforme Equação 3.7




Análise ao Gráfico da Função

Ponto Mínimo: 0
Ponto Médio: 0,5
Ponto Máximo: 1
Inclinação definida em $a$.

Fonte: Adaptado de Haykin (2001).

Observando Haykin (2001), Faceli et al. (2011) e Luger (2013), e aplicando a Equação 3.7 ao perceptron para verificação da ativação e saída conforme o gráfico da Figura 13, temos:

Quadro 12: Integração e Uso da Função de Ativação Logística() dentro do Perceptron

| Perceptron | Dados | | | |
|---|---|---|---|---|
| Entrada de Dados ($\vec{x}$) | 0,03;0,62;0,07 | 0,12;0,56;0,04 | 0,56;0,25;0,40 | 0,36;0,35;0,65 |
| Pesos ($\vec{w}$) | 3,078238563489; -5,531851218469; 4,275331338179 | | | |
| Bias ($b$) | -1,300983326208 | | | |
| Junção $\left(\sum_{i=0}^{n}(x_i\ w_i)\right) + b$ | -4,339111 | -3,858418 | 0,750000 | 0,650000 |
| Função Logística Saída | 0,012880 | 0,020665 | 0,679179 | 0,657010 |
| Supervisão | | | | |
| Saída Desejada | 0 | 0 | 1 | 1 |

Fonte: Produção do Autor (2017).

O Quadro 12 integra as várias partes do perceptron e exibe resultados da função Logística(), Equação 3.7. A saída da função Logística() demonstra a atividade neural conforme o gráfico da Figura 13 para a saída do perceptron 0 *ou* 1, conforme dados de treinamento e saída desejada.

Conclui-se pelo uso da função Logística() nos experimentos computacionais para a RNPM ser treinada pelo "Back- Propagation".

### 3.2.2 Função Tangente Hiperbólica

Lei de formação conforme Haykin (2001):

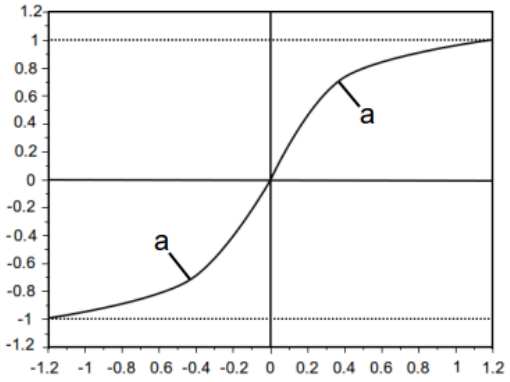
$$\text{Tanh}(x) = \frac{exp(ax) - exp(-(ax))}{exp(ax) + exp(-(ax))} \tag{3.8}$$

$a$: parâmetro (pesos) de inclinação da função, com $a \in \mathbb{R}$.

$x$: variável independente (entrada de dados), $-\infty < x < +\infty$.

$exp(expoente)$: Função que calcula a constante de Euler, $e \approx 2,718$, elevada a um expoente.

Figura 14: Gráfico da Função Tangente Hiperbólica conforme Equação 3.8




Análise ao Gráfico da Função

Ponto Mínimo: -1
Ponto Médio: 0
Ponto Máximo: 1
Inclinação definida em $a$.

Fonte: Adaptado de Haykin (2001).

Observando Haykin (2001), Faceli et al. (2011) e Luger (2013), e aplicando a Equação 3.8 ao perceptron para verificação da ativação e saída conforme o gráfico da Figura 14, temos:

Quadro 13: Integração e Uso da Função de Ativação $Tanh()$ dentro do Perceptron

| Perceptron | Dados | | | |
|---|---|---|---|---|
| Entrada de Dados ($\vec{x}$) | 0,03;0,62;0,07 | 0,12;0,56;0,04 | 0,56;0,25;0,40 | 0,36;0,35;0,65 |
| Pesos ($\vec{w}$) | -9,638554216867; -13,25301204819; -2,409638554217 | | | |
| Bias ($b$) | 9,174698795181 | | | |
| Junção $\left(\sum_{i=0}^{n} x_i\ w_i\right) + b$ | 0,500000 | 7,921687 | -0,500000 | -0,500000 |
| Função Tanh. Saída | 0,462117 | 1,000000 | -0,462117 | -0,462117 |
| Supervisão | | | | |
| Saída Desejada | 1 | 1 | 0 | 0 |

Fonte: Produção do Autor (2017).

O Quadro 13 integra as várias partes do perceptron e exibe resultados da função $Tanh()$, Equação 3.8. A saída da função $Tangh()$ demonstra a atividade neural conforme o gráfico da Figura 14 para a saída do perceptron 0 *ou* 1, conforme dados de treinamento e saída desejada.

Conclui-se pelo uso da função Tangente Hiperbólica nos experimentos computacionais com "Back-Propagation" para treinar RNPM.

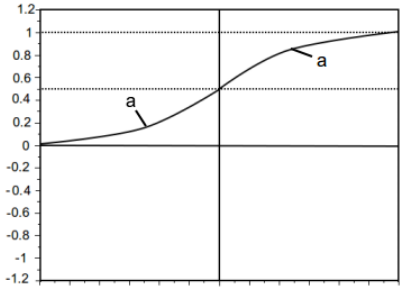
### 3.2.3 Função SoftMax

Lei de formação conforme Goodfellow et al. (2016):

$$\text{Softmax}(z) = \frac{\exp(z)}{\sum_{j=1}^{n} \exp(z_j))} \tag{3.9}$$

$a$: parâmetro (pesos) de inclinação da função, com $a \in \mathbb{R}$.
$x$: variável independente (entrada de dados), $-\infty < x < +\infty$.
$z$: variável de junção de dados de entrada, pesos e bias $z = \left( \sum_{i=0}^{n} x_i \; a_i \right) + b$.
$n$: total de perceptrons na respectiva camada da RNPM de uso da função Softmax.
$b$: Bias (parâmetro de flexibilização para a soma do produto dos vetores $\vec{x}$ e $\vec{w}$).
$exp(expoente)$: Função que calcula a constante de Euler, $e \approx 2,718$, elevada a um expoente.

Figura 15: Gráfico da Função conforme Equação 3.9




Análise ao Gráfico da Função

Ponto Mínimo: 0
Ponto Médio: 0,5
Ponto Máximo: 1
Inclinação definida em $a$.

Fonte: Adaptado de Goodfellow et al. (2016).

O entendimento feito em Dailey et al. (1997) e Goodfellow et al. (2016) remete as propriedades da função Softmax na Equação 3.9, que tem a saída entre o intervalo (0 e 1) conforme Figura 15. Isso faz a função ser adequada para uma interpretação probabilística para o resultado da RNPM.

Portanto, tarefas de reconhecimento de padrões para classificar dados em mais de uma classe, passa a ter uma resposta dentro da Probabilidade conforme a entrada de dados e ajuste dos pesos e bias. Então aplicando a função Softmax a RNPM, pode-se verificar:

$$\vec{x} = \text{entrada}(0{,}55\ 0{,}62\ 0{,}70) \qquad a_{ixj} = pesos \begin{bmatrix} 2,90 & 4,00 & 2,25 \\ 1,85 & 1,00 & 1,20 \end{bmatrix} \qquad b_{1xj} = bias \begin{bmatrix} 1,5 \\ -0,90 \end{bmatrix}$$

$$\begin{bmatrix} \text{Classe 1} : 0,95 = softmax(.) \\ \text{Classe 2} : 0,05 = softmax(.) \end{bmatrix} = \begin{bmatrix} (0,5 * 2,9) + (0,6 * 3,5) + (0,7 * 2,2) + 0,5 \\ (0,5 * 1,8) + (0,6 * 1,0) + (0,7 * 1,9) + (-0,1) \end{bmatrix} = \begin{bmatrix} 5,59 \\ 2,73 \end{bmatrix}$$

A Softmax retornou 0,95 para Classe 1 e 0,05 para a Classe 2, correspondendo a probabilidade da entrada de dados pertencer a uma das duas classes, sendo da Classe 1 com 95% de certeza.

Conclui-se que a função Softmax retorna a probabilidade de determinada entrada de dados pertencer a certa classe de dados, sendo adequada ao uso das RNPM.

## 3.3 Aprendizagem de Máquina (AM) para RNPM

É definido pelo Mitchell (1997, p. 2) que AM é "um programa de computador que aprende a partir da experiência[4] $E$, sobre uma classe de tarefas[5] $T$, com medida de desempenho[6] $P$, se o seu desempenho em $T$, tal como medido por $P$, melhora com $E$". Aplicando o conceito, temos:

$$Desempenho = P(Computador \mapsto \text{RNPM}(E, T)) \tag{3.10}$$

Desempenho $P$ = {Número de iterações para aprendizagem e acurácia na execução da tarefa.}
Experiência E = {[classe ou Serie Numérica] ∈ [conjunto finito de dados]}
Tarefa T = {Classificação ou Previsão}

Entende-se em Mitchell (1997) que a AM pela Equação 3.10 dar-se-á pela indução de uma hipótese ou função, que é aprendida por meio de um conjunto suficiente de exemplos para treinamento, inclusive para indução a exemplos não observados, por causa do conhecimento generalizado após um número "$n$" de iterações e acurácia igual a $f(x) + erro$, onde o $erro$ próximo a 0 define a precisão de acerto da Equação 3.10.

Tratando-se da abordagem de AM Supervisionada para aplicação em problemas do mundo real, observa-se alguns conceitos em Haykin (2001):

Figura 16: Abordagem de Aprendizagem de Máquina para uso com RNPM


► AM Supervisionada: aprendizagem com um professor que conhece o conjunto de dados de treinamento (entradas e saídas).

► Classificação capacidade de classificar em uma classe um dado de entrada com base em um conjunto de treinamento.

► Previsão: identifica um número futuro de forma aproximada ou exata com base em mapeamento de entradas e saídas do conjunto de treinamento.

Fonte: Adaptado de Haykin (2001).

[4] "Experiência: Conhecimento adquirido por prática, estudos, observação, etc.", in Dicionário Priberam da Língua Portuguesa, 2008-2016, <http://www.priberam.pt/dlpo/experiencia> [consultado em 12-03-2016].

[5] "Tarefa: Obra ou porção de trabalho que se deve acabar num determinado prazo.", in Idem, <http://www.priberam.pt/dlpo/tarefa>

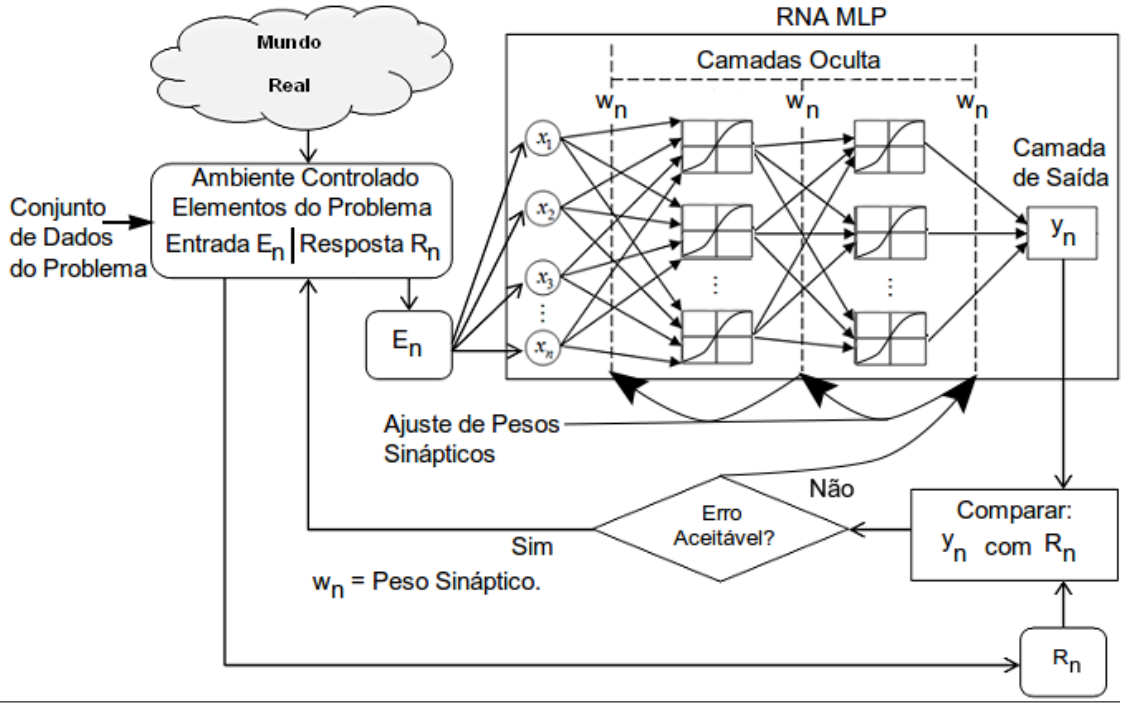
[6] Mitchell (1997, p. 111) Desempenho: Claramente, deve-se usar o número de iterações que produz o menor erro sobre o conjunto de validação, uma vez que esse é o melhor indicador de desempenho da rede ao longo de exemplos invisíveis.

Insere Haykin (2001, p. 75) apud Mendel e McClaren (1970) os conceitos da Figura 16 sobre AM para RNPM a uma teoria ainda mais ampla: "Aprendizagem é um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida".

Sendo assim, e pelas definições da Figura 16, explica Faceli et al. (2011, p. 2): RNPM usa o "processo de indução[7] de uma hipótese[8] (ou aproximação de função) a partir da experiência passada dá-se o nome Aprendizado de Máquina (AM)". Por isso, algoritmos de AM aprendem uma função ou hipótese com base em um conjunto de dados (problema e resposta), sendo capaz de resolver novos problemas não vistos.

Dentro dessa perspectiva, esclarece Russel e Norvig (2013, p. 605) que a RNPM executa iterações "a partir de uma coleção de pares de entrada e saída, [aprendendo] uma função que prevê a saída para novas entradas".

Figura 17: Aprendizado de Máquina para uma RNPM




Fonte: Adaptado de Haykin (2001), Faceli et al. (2011) e Russel e Norvig (2013).

Acompanhado o fluxo da Figura 17, de "Mundo Real"até terminar as "Entradas $E_n$ e "Respostas $R_n$, procede-se a formação do conjunto de dados do problema, transformando-o em ambiente controlado e aplicando-se a RNPM para aprendizado de tarefa, por exemplo, Reconhecimento de Padrões (Classificação e Aproximação de Função (Previsão), executando algoritmo "Retropropagação de Erro", automatizando o conhecimento sobre a situação atual.

[7] "Indução: Consequência tirada dos fatos que se examinam.", in Dicionário Priberam da Língua Portuguesa, 2008-2016, <http://www.priberam.pt/dlpo/inducao> [consultado em 12-03-2016].

[8] "Hipótese: Suposição do que é possível.", in Idem <http://www.priberam.pt/dlpo/hipotese>.

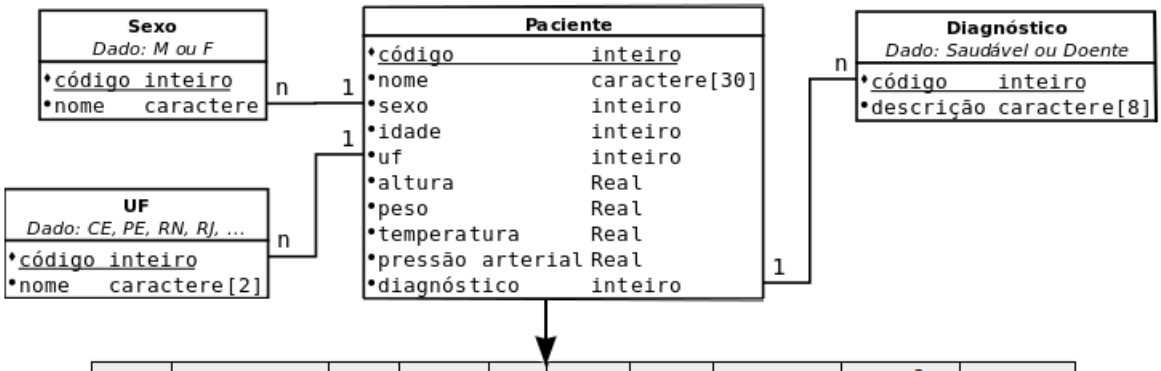
### 3.3.1 Conjunto de Dados

É definido por Faceli et al. (2011, p. 10) que "são formados por objetos que podem representar um objeto físico [...], cada objeto é descrito por um conjunto de atributos de entrada ou vetor de características. [...] Cada atributo está associado a uma propriedade do objeto".

Esse conjunto de dados deve ser estruturados conforme um modelo de dados que dê origem a um banco de dados, para agrupar as tabelas (conjuntos) que são compostas por registros (objetos), contendo linhas divididas em colunas (atributos).

Explica Silberschatz et al. (2006, p. 5) sobre modelo de dados: "uma coleção de ferramentas conceituais para descrever dados, relações de dados, semântica de dados e restrições de consistência". Por exemplo, Álgebra para Conjuntos, Modelo de Diagrama Entidade Relacional (MER) e a Linguagem de Consulta Estruturada - SQL.

Figura 18: Modelo de Entidade Relacional para um Conjunto de Dados com Consulta SQL




| código | nome | sexo | idade | uf | altura | peso | temperatura | pressão arterial | diagnóstico |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Indivíduo A... | M | 38 | PE | 1,75 | 90 | 45 | 14,10 | Doente |
| 2 | Indivíduo B... | F | 40 | PE | 1,62 | 70 | 39 | 15,11 | Doente |
| 3 | Indivíduo C... | M | 45 | RN | 1,65 | 62 | 41 | 13,90 | Doente |
| 4 | Indivíduo D... | F | 50 | CE | 1,59 | 51 | 34 | 13,10 | Saudável |
| 5 | Indivíduo E... | F | 29 | RN | 1,70 | 60 | 32 | 12,80 | Saudável |

Fonte: Adaptado de Silberschatz et al. (2006) e Faceli et al. (2011).

Esclarece Silberschatz et al. (2006) sobre a Figura 18, que um banco de dados "é o conjunto de dados inter-relacionados e um conjunto de programas[9] que permitem aos usuários acessar e modificar esses dados". A organização do conjunto de dados em registros (objetos), que subdividi-se em campos (atributos) é essencial ao treinamento da RNPM.

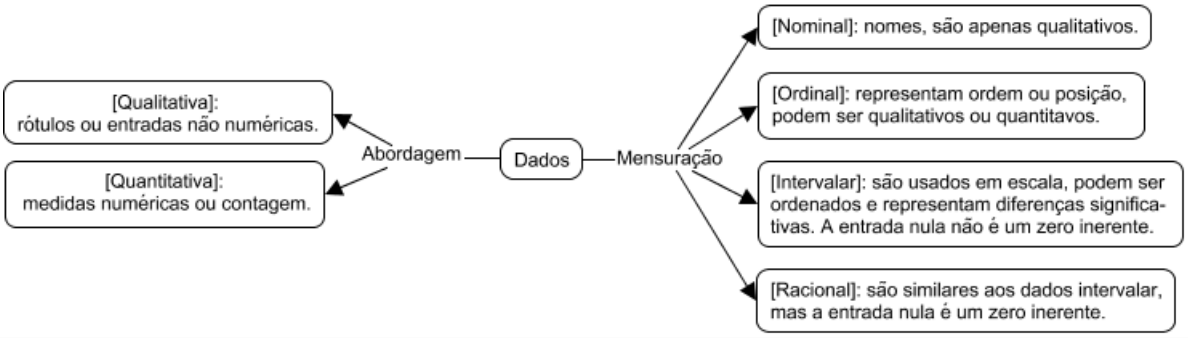
Destaca Faceli et al. (2011) que deve ser feita uma análise formal sobre o conjunto de dados, Figura 18, identificando:

1. Uma Matriz $M$ de objeto "$o$", com atributos "$a$": $M_{oxa}$ para representação dos dados;
2. Cada elemento dessa matriz $x_{ij}$, o i-ésimo objeto que contêm o j-ésimo atributo;
3. O número de atributos que corresponde aos eixos no $R^n$, no espaço de dimensão;
4. O valor do atributo na escala adotada, que determina os pontos nos eixos do $R^n$.

[9] Por exemplo, PostgreSQL, <http://www.postgresql.org/>.

Os atributos do objeto descreve exatamente suas característica, conforme o tipo de dado em questão, por isso destaca Larson e Farber (2010) que ao "realizamos um estudo, é importante saber o tipo de dado envolvido", considerando o zero[10] inerente ou não. Por isso, observa-se:

Figura 19: Abordagem e Mensuração de Tipos de Dados




Fonte: Adaptado de Larson e Farber (2010).

Técnicas de AM usam dados, conforme Figura 19, para aprenderem, mas muitas são limitadas a manipulação de dados somente numéricos. Segundo Faceli et al. (2011), deve ser usado alguma técnica de conversão de valores simbólicos (qualitativos) em números.

Quadro 14: Conversão de Dados Qualitativos ou Simbólico para Numérico

| Conversão Para | Situação |
|---|---|
| 0 e 1. | [O atributo é do tipo nominal], e assume apenas dois valores. |
| 0 e 1. | [Idem], e denota a presença ou ausência de alguma característica. |
| 0 e 1. | [Idem], e apresenta uma relação de ordem onde um dígito é suficiente |
| Pseudoatributos: binário, inteiro ou real. | O atributo qualitativo têm mais de dois valores, não há relação de ordem e são nominal. |
| Sequência de números inteiros. | O atributo qualitativo têm mais de dois valores e existe uma relação de ordem. |

Fonte: Adaptado de Faceli et al. (2011).

Por fim dados quantitativos conforme o Quadro 14 são melhores endereçáveis em $R^n$, com a dimensionalidade necessária à busca de aprendizagem de máquina, conforme explica Faceli et al. (2011): "técnicas de redes neurais artificiais [...] lidam apenas com dados numéricos", então ao usar RNPM deve-se converter dados simbólicos (qualitativos) em números para a adequada assimilação e generalização do conhecimento.

[10] Larson e Farber (2010, p. 10): "Um zero inerente é um zero que significa "nada". Por exemplo, a quantia de dinheiro que você tem em sua poupança pode ser zero dólares. Neste caso, o zero representa nenhum dinheiro; é um zero inerente. Por outro lado, a temperatura $0^oC$ não representa uma condição no qual o aquecimento não está presente. A temperatura de $0^oC$ simplesmente representa uma posição na escala Celsius; não é um zero inerente".

### 3.3.2 Aprendizagem Provavelmente Aproximadamente Correta (PAC)

O modelo de PAC nasce em Valiant (1984), por causa da preocupação de uso de alguma lógica para a escolha correta de um subconjunto de dados, com qualidade probabilística que represente o conhecimento geral de forma generalizada, sobre o domínio de dados, garantindo a acurácia da resposta aprendida.

Nas palavras de Haykin (2001, p. 127), PAC "é uma estrutura probabilística para o estudo de aprendizagem e generalização em sistemas de classificação binária. [...] está intimamente relacionado à aprendizagem supervisionada". Daí a base teórica que determina o tamanho da amostra de dados, para a RNPM aprender provavelmente aproximadamente correto.

A Teoria da Aprendizagem Computacional aborda a questão de saber quantos exemplos em geral são necessário à PAC para a generalização do conhecimento, a isso encontramos a resposta segundo Russel e Norvig (2013, p. 623), pela inequação:

$$N \geq \frac{1}{\varepsilon}\left(ln\frac{1}{\delta} + ln|\mathcal{H}|\right) \tag{3.11}$$

Quadro 15: Significado de Parâmetros da Inequação 3.11 para PAC

| Símbolo | Significado |
|---|---|
| $N$ | $N > 1$. Tamanho da amostra de dados para treinamento supervisionado. |
| $\varepsilon$ | Número constante $(0 < \varepsilon < 0,5]$ que quantifica a taxa de erro de uma hipótese. Define o erro de hipótese da amostra de dados, para generalizar o conhecimento aprendido aplicado a dados desconhecidos em treinamento. |
| ln | Logaritmo Natural: $\log_e n$ |
| $\delta$ | Número constante $(0 < \delta < 0,5]$, quantifica a probabilidade da hipótese errar o alvo, ou seja, da amostra de dados não ser capaz de representar a hipótese corretamente na generalização do conhecimento aprendido. |
| $\|\mathcal{H}\|$ | $h^n$: $h \geq 2$ é a quantidade de hipóteses e $n \geq 1$ é o número de atributos de um registro do conjunto de dados da amostra. Condição: Faz uso de restrição para $\|\mathcal{H}\|$ com conhecimento prévio, delimitando o espaço de hipótese de maneira que o treinamento contenha apenas as hipóteses para aprendizado, e que para a generalização do conhecimento somente induza as hipóteses de treinamento ou mais uma, ou seja, $\|h\| + 1$ (desconhecida). |

Fonte: Adaptado de Haykin (2001) e Russel e Norvig (2013).

Dessa forma, a abordagem para $|\mathcal{H}|$ da Inequação 3.11, pressupõe um subconjunto de dados restrito contendo as hipóteses $h$ que são necessárias a classificação e $|h| + 1$ na iteração de generalização de dados, que satisfaz a possibilidade de algum dado não ser alcançado na indução do conhecimento aprendido. Com o cálculo da Inequação 3.11 encontra o tamanho inicial da amostra para treinamento que retorne uma aprendizagem consistente.

### 3.3.3 Normalização ou Distribuição Normal de Dados

Define Larson e Farber (2010, p. 193):

> Uma distribuição normal é uma distribuição de probabilidade contínua para uma variável aleatória $x$. [Algumas] propriedades: 1. A média, a mediana e a moda são iguais; 2. Uma curva normal tem forma de sino e é simétrica em torno da média; 3. A área total sob a curva normal é igual a um. Distribuições normais podem ser usadas para modelar muitos grupos de mensuração de dados da natureza, indústria e negócios, pressão sanguínea de humanos, etc.

Outrossim, explica Faceli et al. (2011) que faz-se necessário a normalização de dados quando "limites inferiores e superiores de valores dos atributos são muito diferentes", "atributos percentem a escalas diferentes", "evitar que um atributo predomine sobre outro" e "somente o valor do atributo é importante, não o sinal (+/-)".

A normalização pode ser feita pela abordagem de Larson e Farber (2010, p. 196), uma padronização pela distribuição normal padrão com média 0 e desvio padrão 1, denominado Z-Score, calculado pela equação:

$$z = \frac{\text{Valor - Média}}{\text{Desvio Padrão}} \qquad ou \qquad z = \frac{x - \left(\frac{\sum_{i=0}^{n} x_i}{n}\right)}{\sqrt{\frac{\sum_{i=0}^{n}(x_i - \text{Média})^2}{N}}} \qquad ou \qquad z = \frac{x - \mu}{\sigma} \tag{3.12}$$

z: novo valor normalizado.
x: valor a ser normalizado.
N: tamanho total do vetor de dados.
$\mu$: média.
$\sigma$: desvio padrão.

Uma segunda forma de normalização é por reescala, observada em Faceli et al. (2011), denominada MinMax. Inicialmente é definido os valores de mínimo e máximo como uma escala, em seguida o novo valor é gerado, calculado pela equação:

$$V_{novo} = min + \frac{V_{atual} - menor}{maior - menor}(max - min) \tag{3.13}$$

$V_{novo}$: novo valor normalizado.
$V_{atual}$: valor a ser normalizado.
min: menor valor da nova escala escolhida.
max: maior valor da nova escala escolhida.
menor: menor valor dos atributos de dados.
maior: maior valor dos atributos de dados.

Finalizando, esclarece Faceli et al. (2011), sobre qual das Equações 3.12 ou 3.13 usar, enfatizando que "geralmente, é preferível padronizar a reescalar, pois a padronização lida melhor com outliers". Os outliers (valores aberrantes ou atípicos) ocorrem devido a dados mau coletados, diferença de escala, ruído (registro ou atributo fora do domínio do problema), ou dados corretos mas muito diversificados.

### 3.3.4 Reconhecimento de Padrões para Classificar Dados

O Reconhecimento de Padrões é uma tarefa inerente a RNPM, que aprende e induz a Classificação (distribuição por classe) de dados. É dito por Haykin (2001, p. 92) que "o reconhecimento de padrões é formalmente definido como o processo pelo qual um padrão/sinal recebido é atribuído a uma classe dentre um número predeterminado de classes (categorias)".

Então com um número quaisquer de registros "$r$", de uma base de dados, pode-se organizar os dados conforme seus atributos: $r = registro\{atributo_1,\ atributo_2,\ atributo_3,\ atributo_n\}$.

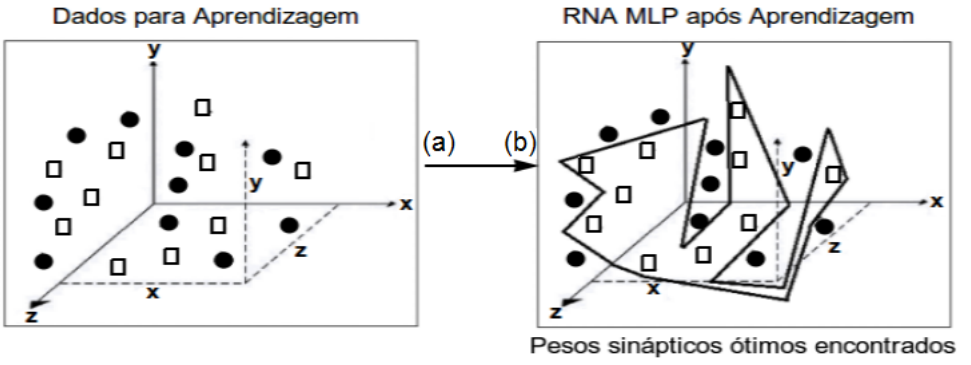
Dado as classes $C_0,\ C_1,\ C_n$ com seus respectivos registros $\{r_1,\ r_2,\ r_3,\ r_n\}\ \forall\ \{r\}$, faz-se a transformação dos dados em números (caso necessário) e a normalização da base de dados. Nisso cada classe possui seu respectivo padrão de comportamento para aprendizagem.

Cada classe é o espaço de instâncias de um domínio de dados, e a RNPM aprende a classificar os registros a sua classe específica $C$. Com esse aprendizado processado, o próximo passo é a generalização do conhecimento, submetendo à RNPM uma entrada $r \in C$ desconhecido, para a classificação correta na classe específica $C$, mesmo sendo $r$ desconhecido no momento da aprendizagem.

Explica Silva et al. (2010) que o treinamento de uma RNPM para a classificação de padrões tem o objetivo de "associar um padrão de entrada (amostra) para uma das classes previamente definidas [...] o problema a ser tratado possui um conjunto discreto e conhecido das possíveis saídas desejadas".

Observa-se na Figura 20 o espaço de instâncias do domínio de dados e a aprendizagem da RNPM após "$n$" iterações:

Figura 20: Visão de Dados Antes e Depois do Treinamento de RNPM para Classificação




Fonte: Adaptado de Silva et al. (2010) e Russel e Norvig (2013).

Chega-se a conclusão pela observação a Silva et al. (2010) e Russel e Norvig (2013), que a RNPM é capaz de aprender a reconhecer padrões demonstrado na Figura 20 e generalizar o conhecimento, para classificar a entrada de dados não conhecida em momento de treinamento, enfatizado a capacidade da RNPM, em tarefa de classificação.

### 3.3.5 Aproximação de Função para Previsão

A aproximação de função extrai do conjunto de dados uma função $f(.)$ que reproduz o comportamento de uma serie de dados. Nas palavras de Haykin (2001) o objetivo é descobrir uma função desconhecida $f(.)$ que se aproxime do comportamento dos dados, descrevendo o mapeamento de entrada e saída.

Entende-se em Haykin (2001), Braga et al. (2007) e Silva et al. (2010) que o problema de Previsão pode ser resolvido com aproximação de função, uma vez que para resolver a Previsão, a RNPM aprende a estimar situações futuras com base em uma série de dados de entrada.

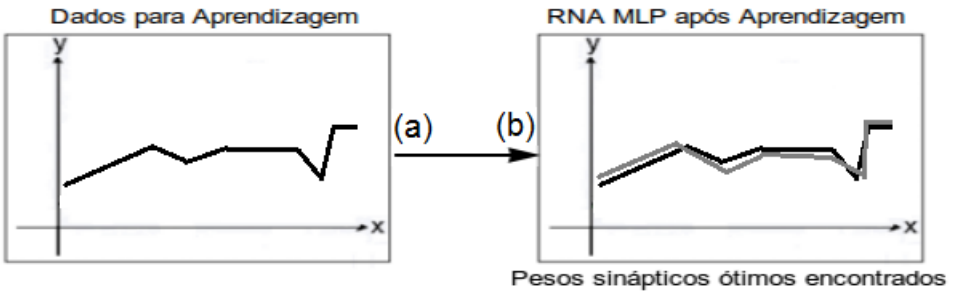
Dados os valores de entrada $r = \{valor_1,\ valor_2,\ valor_n\}$ e de os de saída $s = \{valor_1,\ valor_2,\ valor_n\}$, de uma base de dados, com os conjuntos $D = \{r_1, r_2, r_3,\ ..., r_n\}\ \forall\ \{r\}$ e $S = \{s_1, s_2, s_3,\ ..., s_n\}\ \forall\ \{s\}$, sabendo-se que $S$ é resultado de $D$, submete-se $D$ a RNPM para aprender uma função $f(D)$ para calcular $S$.

Feito isso, o conjunto $D$ é o espaço de instâncias de um domínio de dados, e a RNPM aprende a previsão de $S$ com base em $D$. Com esse aprendizado processado, o próximo passo é a generalização do conhecimento, onde a RNPM encontra o próximo número da serie de $S \in D$, para a previsão correta de uma serie que represente $D$.

Explica Silva et al. (2010) que o objetivo dos sistemas de previsão é estimar valores futuros de um processo, após conhecer medidas prévias sobre o domínio de dados. Pode-se aplicar aproximação de função para previsão na industria, finanças saúde, clima, movimento, etc.

Tem-se então na Figura 21 o espaço de instâncias do domínio de dados e a aprendizagem da RNPM após "*n*" iterações:

Figura 21: Visão de Dados Antes e Depois do Treinamento de RNPM para Previsão




Fonte: Adaptado de Russel e Norvig (2013).

A Figura 21 descreve o conjunto de dados antes da aprendizagem (a) de RNPM, e depois (b). Em Figura 21 (b) as linhas de cores diferentes representas os dados e a aprendizagem da RNPM para previsão de valores.

Por fim verifica-s na demonstração da Figura 21 que a RNPM é capaz de encontrar uma função de aproximação, para calcular a previsão de um valor ou valores futuro com base em uma serie de dados, prevendo a continuidade da sequência aprendida.

### 3.3.6 Aprendizagem Supervisionada (AS)

Segundo Haykin (2001) a aprendizagem de RNPM pode ser efetuada de forma supervisionada (sob a tutela de um professor) por correção e erro, corrigindo os pesos e bias por retropropagação, para obtenção, armazenamento e processamento de conhecimento[11], usando dados de entrada que representam instâncias de um domínio de dados para treinamento e aprendizagem da resolução ao problema proposto. Entende-se em Haykin (2001) e Russel e Norvig (2013) o pseudocódigo para AS:

**Algoritmo 1:** Uso da Aprendizagem Supervisionada($D$, $S$, $\Omega$, $\varepsilon$)

**Entrada:** $D$: Dados do Domínio do Problema;
$S$: Dados (experiência) de comparação (Professor) com a saída da RNPM;
$\Omega$: RNPM;
$\varepsilon$: Tamanho do Erro, por exemplo,$10^{-3}$.

**Saída:** RNPM treinada e retornando uma hipótese como solução.

```
1  início
2      repita
3          Ω_saída = Ω(D);
4          erro = (Ω_saída comparado com S);
5          se (erro > ε) então
6              Atualizar Pesos e Bias de Ω;
7          fim
8      até [(erro ≤ ε) ou (n épocas)];
9      retorna Ω;
10 fim
```

Fonte: Produção do Autor (2017).

Executando o Algoritmo 1 a RNPM aprende, conforme as palavras de Russel e Norvig (2013): "aprendizagem é a busca através do espaço de hipóteses possíveis por aquele [algoritmo] que terá o bom desempenho, mesmo em novos exemplos de treinamento", fazendo menção à aprendizagem da RNPM em um espaço de hipóteses dentro do $R^n$, em busca de pesos e bias ótimos globais.

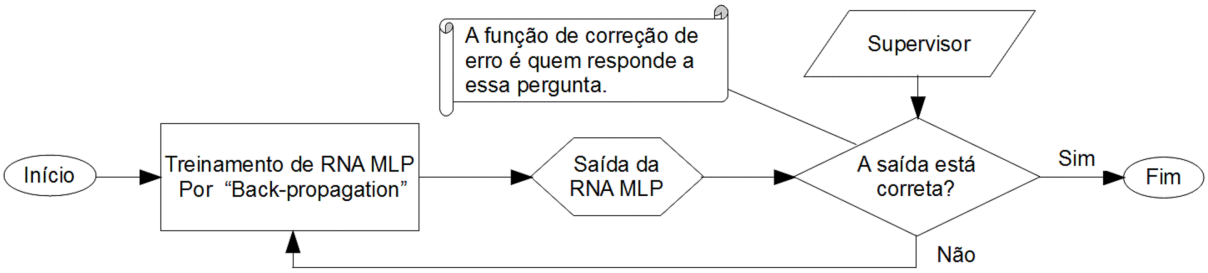
Chega-se a conclusão que o Algoritmo 1 é capaz de realizar o treinamento da RNPM, caso a hipótese verdadeira esteja dentro do domínio de dados, sendo aplicável as tarefas de Classificação (Subseção 3.3.4) e Previsão (Subseção 3.3.5), para generalizar o conhecimento aprendido a problemas diversos não apresentados em treinamento, desde que pertençam ao mesmo domínio de dados.

[11] Haykin (2001, p. 49) apud Fischler e Firschein (1987): "Conhecimento se refere à informação armazenada ou a modelos utilizados por uma pessoa ou máquina para interpretar, prever e responder apropriadamente ao mundo exterior".

## 3.4 Aprendizagem por Correção de Erro

Define Haykin (2001, p. 77) sobre Aprendizagem por Correção de Erro: "o sinal de erro aciona um mecanismo de controle, cujo propósito é aplicar uma sequência de ajustes corretivos aos pesos do neurônio $k$". Nessa aprendizagem uma função de erro é usada para controle do erro de RNPM em fase de treinamento. Então temos o seguinte fluxograma:

Figura 22: Demonstração da Posição da Função de Correção e Erro




Fonte: Adaptado de Haykin (2001) e Silva et al. (2010).

A Figura 22 exibe o fluxograma onde a função de correção erro está inserida para medir o erro do treinamento Supervisionado da RNPM, atualizando os pesos e bias por retropropagação, até que a condição de parada para as épocas seja satisfeita.

Por isso que Silva et al. (2010, p. 99) deixa claro que após a arquitetura da RNPM está pronta, "o próximo passo para derivação do algoritmo "Back-Propagation" consiste em defini a função representativa do erro de aproximação".

### 3.4.1 Função de Erro Quadrático Médio (EQM)

Observa-se em Silva et al. (2010, p. 99) a Função de Erro Quadrático Médio:

$$Erro_{\text{Médio}} = \frac{1}{P}\sum_{k=1}^{P}\left[\frac{1}{2}\sum_{j=1}^{Perceptros}(d_j(k)-Y_j(k))^2\right] \tag{3.14}$$

$(k)$ : número de iterações no processamento de aprendizagem.
$P$ : Número de amostras "população" ou registros do conjunto de dados para treinamento.
$j$ : Índice $\in \mathbb{Z}^+$ para o número de neurônio de saída na RNPM.
$d_j$ : Resposta ou saída desejada para o neurônio de saída $j$.
$Y_j$ : Resposta ou saída do neurônio $j$ calculada pela RNPM na respectiva iteração.
$Perceptros$ : Total de perceptrons na camada de saída da RNPM.

Por fim a Função 3.14 mede o desempenho da AM Supervisionada da RNPM com todos os seus parâmetros, após um número de épocas de treinamento pelo algoritmo "Back-Propagation", minimizando o erro médio quadrático até um pequeno valor, por exemplo, $10^{-3}$.

### 3.4.2 Algoritmo “Back-Propagation”

O algoritmo "Back-Propagation" tradução Retropropagação de Erro, foi criado por Rumelhart et al. (1986) para treinamento Supervisionado de RNPM, buscando os pesos e bias que representam o conhecimento de resolução de determinado problema, comparação a saída da RNPM e o valor correto, retropropagando o erro (diferença) até que a saída e valor desejado sejam aproximados.

Para Haykin (2001), a RNPM é uma forma de codificarmos o conhecimento através de treinamento de algoritmo "Retropropagação de Erro", buscando iterativamente dentro do contexto de dados sobre o problema e resolução, valores que minimize a equação:

$$e_j(n) = d_j - y_j(n) \tag{3.15}$$

$e$ : Erro do neurônio.
$j$ : Índice do neurônio.
$d$ : Saída desejada.
$y$ : Saída da RNPM.
$n$ : Número de épocas.

Na Equação 3.15 o neurônio $j$ é um nó de saída da RNPM, onde o erro deve ser minimizado, para que ocorra aprendizagem por retropropagação de erro.

Explica Silva et al. (2010, p. 95) sobre o ajuste do erro e a retropropagação:

> Assim, em função desses valores de erros, aplica-se, em seguida, a segunda fase do método de *backpropagation*, denominada "propagação reversa"(*backward*). [...] as alterações (ajustes) dos pesos sinápticos e limiares [bias] de todos os neurônios da rede são executados no decorrer desta fase. Em suma, as aplicações sucessivas das fases de *forward* [à frente] e *backward* [para trás] fazem com que os pesos sinápticos e limiares [bias] dos neurônios [perceptrons] se ajustem automaticamente em cada iteração, implicando-se na gradativa diminuição da soma dos erros produzidos pelas respostas da rede frente àquelas desejadas.

Entende-se em Russel e Norvig (2013) o planejamento necessário de uma RNPM, para uso do algoritmo "Back-Propagation":

$1^o$ Defini se o treinamento será para classificação.
$2^o$ Preparar em forma de matriz os dados para treinamento, saída desejada e teste de acurácia.
$3^o$ Arquitetar a RNPM, quantidade de camadas e perceptrons por camada.
$4^o$ Definir: a) o erro geral da rede;
b) o algoritmo de inicialização de pesos e bias;
c) o número máximo de iterações;
d) a função de avaliação de erro de treinamento;
e) a forma de validação da acurácia.

Com esse planejamento definido é repassada a RNPM para ser treinada pelo algoritmo "Back-Propagation".

**Algoritmo 2:** Back-Propagation(exemplos, rede)

**Entrada:** exemplos: dados de entrada, vetor $x$ e de saída vetor $y$;
rede: RNPM com L camadas, pesos $w_{ij}$ e função de ativação $g$;
Variáveis Locais: $\Delta$, um vetor de erros, indexado pelo nó (perceptron) da rede.
$\alpha$: Valor da taxa de aprendizagem.

**Saída:** rede treinada

**início**
  **para** peso $w_{i,j}$ **faça** $w_{i,j} \leftarrow$ número aleatório **fim**
  **repita**
    **para** $exemplo_{i,j}$ **faça** /* Propagar entradas para frente. */
      **para** nó $i$ na camada de entrada **faça**
        $a_i \leftarrow x_i$
      **fim**
      **para** $l = 2$ até $L$ faça
        **para** cada nó $j$ na camada $l$ **faça**
          $in_j \leftarrow \sum_{i}^{n} w_{i,j}\, a_i$
          $a_i \leftarrow g(in_j)$
        **fim**
      **para** nó $j$ na camada de saída **faça** /* Retropropagar correção de erro. */
        $\Delta[j] \leftarrow g'(in_j) * (y_i - a_j)$
      **fim**
      **para** $l = L - 1$ até 1 **faça**
        **para** nó $i$ na camada $l$ **faça**
          $\Delta[i] \leftarrow g'(in_i) \sum_{j}^{n} w_{ij} \Delta[j]$
        **fim**
      **fim**
      /* Atualiza cada peso na rede usando delta. */
      **para** peso $w_{i,j}$ na rede **faça**
        $w_{i,j} \leftarrow w_{i,j} + \alpha * a_i * \Delta[j]$
      **fim**
    **fim**
  **até** *(que algum critério de parada seja satisfeito)*;
  **retorna** *(RNPM treinada)*
**fim**

Fonte: Adaptado de Russel e Norvig (2013, p. 640).

O Algoritmo 2 inicializa os pesos e bias, com números aleatórios, e nas palavras

de Russel e Norvig (2013) entende-se:

1. É feito o calculo de valores de $\Delta$ para os perceptrons de saída usando o erro observado;
2. A partir da camada de saída, até a primeira:
   - Retropropagar os valores de $\Delta$ à camada anterior para corrigir o erro da rede;
   - Atualizar os pesos e bias entre as camadas.

Esse Algoritmo 2 inicia o cálculo dos pesos e bias, conforme Linha 2, a partir de números aleatórios, podendo ser essa linha substituída por uma heurística para o cálculo devalores iniciais, segundo as palavras de Luger (2013, p. 103): "[...] heurísticas são formalizadas como regras para escolher aqueles ramos em um espaço de estados que têm maior probabilidade de levarem a uma solução aceitável para o problema".

A literatura científica lista versão ou tipos diferentes para o algoritmo "Back-Propagation", onde busca-se melhor convergência ao ótimo global de pesos e bias:

Quadro 16: Lista de Algoritmos "Back-Propagation"

| Algoritmos de "Back-Propagation" | Literatura Científica |
|---|---|
| Resiliente Retropropagação (RPROP) | Riedmiller e Braun (1993)<br>Riedmiller e Rprop (1994)<br>Beale et al. (2016) |
| Regularização Bayesiana | Buntine e Weigend (1991)<br>Kamran et al. (2016)<br>Beale et al. (2016) |
| Gradiente Descendente com Taxa de Aprendizagem Adaptativa | Plagianakos et al. (2001)<br>Bottou (2012)<br>Beale et al. (2016) |
| Gradiente Descendente com *Momentum* e Taxa de Aprendizagem Adaptativa | Dao e Vemuri (2002)<br>Beale et al. (2016) |

Fonte: Produção do Autor (2017).
Nota: O diferencial de cada algoritmo é descrita adiante.

O Quadro 16 lista algoritmos de "Back-Propagation", cada um com seu método próprio de convergência de pesos e bias para o ótimo global ou aproximado, entretanto por serem "Back-Propagation" é obrigatória a inicialização prévia de pesos e bias para todos esses algoritmos, para então começar a procura por aprendizagem dentro do espaço de busca dos dados de treinamento.

O Resiliente Retropropagação (RPROP), referenciado no Quadro 16, tem a particularidade de administrar o cálculo da magnitude de derivadas parciais, eliminando o resultado muito pequeno do gradiente de funções sigmóides. Na retropropagação do erro, os pesos e bias são atualizados pela direção do sinal da derivada. É usado um valor separado para determinar

valor de atualização do peso, sendo "incrementado" ou "decrementado", sempre que a derivada da função resulta no mesmo sinal em duas iterações sucessivas. Se a derivada for zero, o valor de atualização permanece o mesmo. Sempre que os pesos são oscilantes, a alteração de peso é reduzida. Se o peso continua a mudar na mesma direção por várias iterações, a magnitude da mudança de peso aumenta.

A Regularização Bayesiana, referenciado no Quadro 16, treina uma RNPM desde que as funções de ativação dos perceptrons sejam deriváveis. Então é feita a combinação linear de erros quadráticos dos pesos e bias comparando com a saída supervisionada da RNPM. Usa o método numérico de Levenberg-Marquardt para calcular a matriz Jacobiana dos pesos e dados de entrada de treinamento.

O Gradiente Descendente com Taxa de Aprendizagem Adaptativa (TAA), referenciado no Quadro 16, calcula a retropropagação dos pesos e bias de acordo com descida do gradiente. Usa uma TAA para variar de acordo com a melhoria ou piora da minimização do erro da RNPM. A variação ocorre "incrementando" ou "decrevementando" o valor da TAA, que pode manter-se constante dependendo do resultado do cálculo do gradiente.

O Gradiente Descendente com o Termo Momentum e Taxa de Aprendizagem Adaptativa (TAA), referenciado no Quadro 16, também usa TAA, mas acrescenta um número chamado Momentum. O Momentum multiplica o resultado do grandiente, os pesos, os bias e a TAA no ajuste do erro pela retropropagação. É uma tentativa do "Back-Propagation" sair de mínimos locais e acelerar a convergência para a minimização do erro quadrático da RNPM.

Desde Rumelhart et al. (1986), criador do algoritmo "Back-Propagation", passando pelas versões referenciadas no Quadro 16, até os dias de hoje, tem-se verificado que esse algoritmo é sensível a inicialização de pesos e bias. Dependendo de quais valores iniciais, é calculado o passo para frente e retropropagado a atualização do erro, sendo que valores diferentes de inicialização causam em muitos casos, acurácia e tempo de treinamento diferente.

É feito o entendimento em Samarasinghe (2006), que o espaço gerado pelo dados de treinamento para a busca por solução aproximada ou exata ao ótimo global, contêm mínimos locais. A inicialização de valores de pesos e bias pode direcionar o "Back-Propagation" a ficar preso em algum mínimo local. Entretanto algum outro valor de pesos e bias direciona o "Back-"Propagation" a alcançar o melhor treinamento de RNPM. Portanto, o resultado de acurácia e tempo de treinamento está diretamente relacionado a qualidade dos valores iniciais dos pesos e bias.

Conclui-se que continua em aberto a pesquisa por uma solução de inicialização de pesos e bias para o ponto de partida do "Back-Propagation" que resulte em máxima acurácia. É possível uma melhoria do cálculo inicial de pesos e bias pelo cômputo de uma heurística, para o "Back-Propagation" a treinar RNPM, com o objetivo de melhoria de resolução de problema de reconhecimento de padrões para classificar dados.

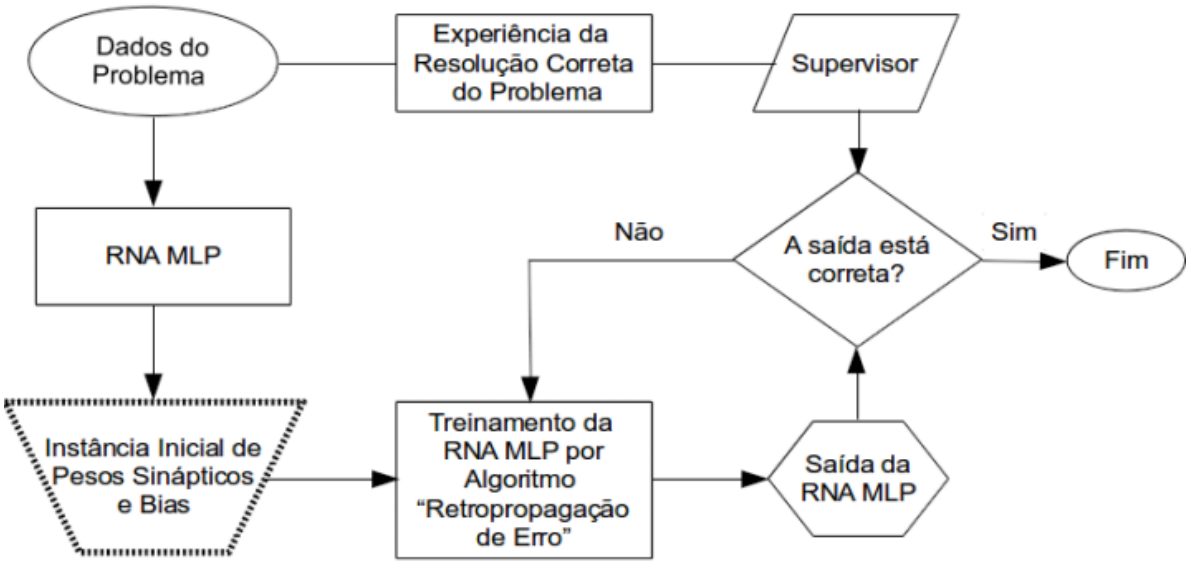
## 3.5 Métodos Tradicionais de Calcular Pesos e Bias Iniciais

Observa-se em Silva et al. (2010), Faceli et al. (2011) e Russel e Norvig (2013), que o modelo matemático de uma RNPM necessita de pesos e bias exatos ou aproximados do ótimos globais, em todas as camadas. Esses pesos e bias são calculados em número ”$n$” de épocas, iniciando de um ponto de partida até alcançar a melhor acurácia, estabelecendo-se um critério de parada para as épocas (iterações).

É esclarecido por Silva et al. (2010) que a RNPM aprende a partir de padrões que exprimem comportamento, iniciando por pesos e bias aleatórios, consistindo em repetir "$n$" épocas até que a generalização acurada seja produzida na saída, cujo treinamento ocorre de forma supervisionada.

É entendido em Faceli et al. (2011) que o processamento do "Back-Propagation" inicia-se com a inicialização de pesos e bias com valores aleatórios, em seguida é apresentado todos os exemplos de treinamento a RNPM e usado o erro médio da saída da RNPM para ajuste dos valores pesos e bias, até que o erro seja pequeno, por exemplo, $10^{-3}$, ou o número de épocas (predefinido seja atingido.

Figura 23: Fluxograma que Exibe o Cálculo Inicial de Pesos e Bias




Fonte: Adaptado de Haykin (2001) e Luger (2013).

É verificado na Figura 23 a passagem de fluxo computacional pela instância inicial de pesos e bias, identificado no trapézio pontilhado, devendo ser no início da execução algoritmo "Retropropagação de Erro", por isso que Luger (2013) diz que é necessário certa quantidade de esforço computacional para "alcançar a convergência para algo próximo ao ótimo global no espaço de peso".

Observando a literatura científica para o cálculo dos pesos e bias iniciais, em instanciar a RNPM para o treinamento supervisionado, verifica-se:

Quadro 17: Métodos para o Cálculo Inicial de Pesos e Bias da Literatura Científica

| Métodos de Inicialização de Pesos e Bias | Literatura Científica |
|---|---|
| NÚMEROS ALEATÓRIOS: um conjunto de números uniformemente distribuídos deforma aleatória no intervalo de $(0 \geqslant n \leqslant 1)$. | Pavelka e Prochazka (2004)<br>Beale et al. (2016)<br>MathWork (2016) |
| NÚMEROS ALEATÓRIOS SIMÉTRICOS:<br>um conjunto de números simétricos aleatórios. Um número simétrico ou oposto de outro é quando representado em uma reta numérica e possuir a mesma distância da origem em relação a outro número. | Pavelka e Prochazka (2004)<br>Singh et al. (2013)<br>Beale et al. (2016)<br>MathWork (2016) |
| NÚMEROS ALEATÓRIOS PEQUENOS<br>:um conjunto de números uniformemente distribuídos deforma aleatória no intervalo de $(-0.1 \geqslant n \leqslant 0.1)$. | Pavelka e Prochazka (2004)<br>Beale et al. (2016)<br>MathWork (2016) |
| ALGORITMO NGUYEN-WIDROW:<br>escolhe valores a fim de distribuir a região ativa de cada neurônio através do espaço gerado pelos dados de entrada. Os valores contêm um grau de aleatoriedade, então não são os mesmos cada vez que o algoritmo é executado. | Ngugen e Widraw (1990)<br>Pavelka e Prochazka (2004)<br>Beale et al. (2016)<br>MathWork (2016) |
| VALOR DO PONTO MÉDIO DO VETOR DE ENTRADA DE DADOS (MIDPOINT):<br>calcula valores de inicialização de pesos pela entrada de dados, calculando a média dos valores de entrada. | Singh et al. (2013)<br>Beale et al. (2016)<br>MathWork (2016) |

Fonte: Produção do Autor (2017).

O Quadro 17 lista métodos diferentes de inicializar pesos e bias conforme a literatura científica para o ponto de partida do "Back-Propagation" treinar RNPM, com treinamento Supervisionado, uma vez que é necessário atribuir algum valor inicial aos pesos e bias, para buscar iterativamente os valores ótimos globais ou aproximados, que represente a aprendizagem com acurácia da resolução do problema em foco.

É bastante usado algum tipo de inicialização randômica para os pesos e bias, ou algoritmos como "Nguyen-Widrow" ou "Midpoint" conforme o Quadro 17, entretanto, o campo de pesquisa científica continua buscado melhoria na inicialização de pesos e bias.

Conclui-se que apesar da literatura científica apresentar diferentes métodos de calcular os pesos e bias iniciais para o "Back-Propagation", não existe um denominador comum que garanta a aproximação ou exatidão do ótimo global em acurácia, estando em aberto o campo de pesquisa por um algoritmo melhor.

## 3.6 Confiabilidade: Métricas de Erros e Acertos para RNPM

Esclarece Sommerville (2011, p. 225) que a "confiabilidado relaciona a implementação do sistema com suas especificações ou seja, o sistema está se comportando de forma confiável se seu comportamento é coerente com o definido na especificação".

A RNPM está pronta à execução de tarefas, por exemplo, Classificação quando o aprendizado é implementado por algoritmo de treinamento para o reconhecimento de padrão sobre os dados. Destaca Sommerville (2011) que as métricas são importantes para aferir confiabilidade.

No que concerne as métricas para aferição de AM, Faceli et al. (2011) enfatiza que a avaliação experimental de AM para RNPM pode ser realizada medindo a acurácia do modelo gerado e o tempo de aprendizado. Para isso usa-se as métricas:

Quadro 18: Métricas de Avaliação de Aprendizado de RNPM

| Tarefa da RNPM e Modelo Matemático da Métrica para Avaliação |
|---|
| Para a Classificação e Previsão avalia-se o tempo de aprendizagem pela seguinte equação:<br>Tempo Total de Iterações TTI = $\sum_{i=1}^{k} \mathrm{I}(O(n^m))$<br>I: Número necessário de iterações para a RNPM aprender;<br>$O(n^m)$: Complexidade Assintótica Big "O" do algoritmo de aprendizagem;<br>$n$: Tamanho da entrada de dados (conjunto de treinamento);<br>$m$: Expoente calculado na complexidade Big "O". |
| Avalia-se a aprendizagem somente em Classificação de dados pela equação:<br>$erro(\Omega) = \frac{1}{\lvert\vec{n}\rvert} \sum_{i=1}^{\lvert\vec{n}\rvert} I[y_i \neq \Omega(x_i)]$<br>$erro$: Percentual de erro da RNPM;<br>$\Omega$: RNPM;<br>$\vec{n}$: Conjunto de dados de entrada (treinamento);<br>$I$: Função que em cada Iteração retorna: [0 se (≠) falsa] ou [1 se (≠) verdadeira];<br>$y_i$: Registro que corresponde a saída correta (Professor) da RNPM;<br>$x_i$: Registro do conjunto de treinamento.<br>Complementando a avaliação, em relação a acurácia, temos:<br>$ac(\Omega) = 1 - erro(\Omega)$<br>$ac$: Percentual de acurácia (acertos) da RNPM. |

Fonte: Adaptado de Faceli et al. (2011).

De mesmo modo, tem-se métricas específicas de avaliação previsão de dados:

Quadro 19: Continuação das Métricas de Avaliação de Aprendizado da RNPM

| Tarefa da RNPM e Modelo Matemático da Métrica para Avaliação |
|---|
| Avalia-se a aprendizagem para Previsão pelo erro quadrático médio:<br>$$MSE(\Omega) = \frac{1}{\lvert\vec{n}\rvert}\sum_{i=1}^{\lvert\vec{n}\rvert}[y_i - \Omega(x_i)]^2$$<br>$MSE$: "Mean Squared Error" (Erro Médio Quadrático) da RNPM;<br>$\Omega$: RNPM;<br>$\vec{n}$: Conjunto de dados de entrada (treinamento);<br>$y_i$: Registro que corresponde a saída correta (Professor) da RNPM;<br>$x_i$: Registro do conjunto de treinamento.<br>Complementando a avaliação, em relação a acurácia, temos:<br>$$ac(\Omega) = 1 - MSE(\Omega)$$<br>$ac$: Percentual de acurácia (acertos) da RNPM. |
| Avalia-se a aprendizagem para Previsão pela distância média absoluta:<br>$$MAD(\Omega) = \frac{1}{\lvert\vec{n}\rvert}\sum_{i=1}^{\lvert\vec{n}\rvert}\lvert y_i - \Omega(x_i)\rvert$$<br>$MAD$: "Mean Absolute Distance" (Distância Média Absoluta) do erro de RNPM;<br>$\Omega$: RNPM;<br>$\vec{n}$: Conjunto de dados de entrada (treinamento);<br>$y_i$: Registro que corresponde a saída correta (Professor) da RNPM;<br>$x_i$: Registro do conjunto de treinamento.<br>Complementando a avaliação, em relação a acurácia, temos:<br>$$ac(\Omega) = 1 - MAD(\Omega)$$<br>$ac$: Percentual de acurácia (acertos) da RNPM. |

Fonte: Adaptado de Faceli et al. (2011).

As equações, nos Quadros 18 e 19, satisfazem a avalização de AM para RNPM, medindo o quanto de iterações, acurácia e erro, sendo obrigatório na averiguação de confiabilidade para os algoritmos de treinamento. Dentro dessa perspectiva de medida de desempenho Faceli et al.(2011) também orienta o uso da Matriz de Confusão.

Entretanto, em AM para RNPM, além de fazer uso de métricas para aferição da saída de dados, necessita-se garantir o máximo exploração dos dados de entrada, visando abrangência de aprendizado. Nisso Silva et al. (2010) e Russel e Norvig (2013) destacam o método de Validação Cruzada de Dados (VCD).

### 3.6.1 Matriz de Confusão

Destaca Witten et al. (2011) o custo de classificações erradas, devido o apoio a tomada de decisão que é feito com base na AM para RNPM, considerando a consequência de acertar ou errar por causa de uma resposta autônoma. Um método para averiguar o quanto o conhecimento foi aprendido ou ficou confuso pela RNPM é a Matriz de Confusão.

Quadro 20: Modelo de Matriz de Confusão para um Classificador com Três Classes

| Classificado | Classe 1 | Classe 2 | Classe 3 |
|---|---|---|---|
| Classe 1 | CC | CE | CE |
| Classe 2 | CE | CC | CE |
| Classe 3 | CE | CE | CC |

Onde: {CC, CE} $\in Z^+$
CC: Classificação Correta
CE: Classificação Errada
Melhor Caso: CC $\geqslant 1$ e CE = 0

Fonte: Adaptado de Witten et al. (2011).

Entende-se em Witten et al. (2011) que a Matriz de Confusão, Quadro 20, concentra os acertos ou erros sobre o subconjunto $V$ de dados para validação, usado como base de cálculo para averiguação da acurácia. A classificação correta é contada na diagonal da matriz, se estão errados são subtraídos da diagonal e quantificado na linha e coluna referente ao erro.

Observa-se em Faceli et al. (2011) apud (Monard e Baranuskas, 2003), uma ampliação de métricas sobre a matriz de confusão, para medir a AM para RNPM:

Quadro 21: Métricas sobre a Matriz de Confusão para Medição da AM para RNPM

| | |
|---|---|
| Taxa de Erro (Er) da Classe:<br>$Er_{Classe\ 1} = \frac{\sum_{i=Classe\ 1}^{Classe\ n} CE}{CC_{Classe\ 1} + \sum_{i=Classe\ 1}^{Classe\ n} CE}$ | Taxa de Acerto (Ac) da Classe:<br>$Ac_{Classe\ 1} = \frac{CC_{Classe\ 1}}{CC_{Classe\ 1} + \sum_{i=Classe\ 1}^{Classe\ n} CE}$ |
| Taxa de Erro (Er) do Classificador:<br>$Er_{Classif.} = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} CE}{\lvert\text{Subconjunto de Validação}\rvert}$ | Taxa de Acerto (Ac) do Classificador:<br>$Ac_{Classif.} = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} CC}{\lvert\text{Subconjunto de Validação}\rvert}$ |

Fonte: Adaptado de Faceli et al. (2011).

Finaliza-se, afirmando que as métricas do Quadro 21 sobre a Matriz de Confusão, conforme o entendimento feito em Witten et al. (2011) e Faceli et al. (2011), são métricas indispensáveis para medir a AM da RNPM, observando a diagonal principal e valores acima e a abaixo, para aferir confiabilidade da resposta e quantificar as taxas de erro e acerto.

### 3.6.2 Validação Cruzada de Dados (VCD) ou "K-Fold Cross-Validation"

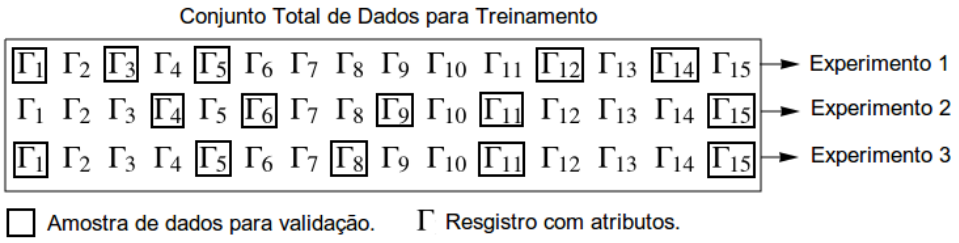
Afirma Silva et al. (2010, p. 147) que a VCD é o "conjunto total de dados (amostras) disponíveis é aleatoriamente dividido em duas partes, isto é, subconjunto de treinamento e subconjunto de teste (validação)".

Observa-se duas técnicas em Silva et al. (2010) e Russel e Norvig (2013): na **1ª** os dados são divididos aleatoriamente por percentual "x" para treinamento e validação; ou, a **2ª** usa um número "$K$" para divisão dos dados, usando repetidas vezes com os registros, alternando entre treino e validação, em repetidos treinamentos e validações dos dados.

Com foco na **1ª** técnica, cria-se dois subconjuntos, $T$ e $V$, fazendo uso da equação:

$$\text{Dados} = T + V, \text{ onde } \begin{cases} \text{(T: para treinamento)} \forall \text{ (60\%, 70\%, 80\% ou 90\%)} \\ \text{(V: para Validação)} \forall \text{ (40\%, 30\%, 20\% ou 10\%)} \end{cases} \tag{3.16}$$

Figura 24: Visão dos Dados para Treinamento e Validação pela Técnica Equação 3.16

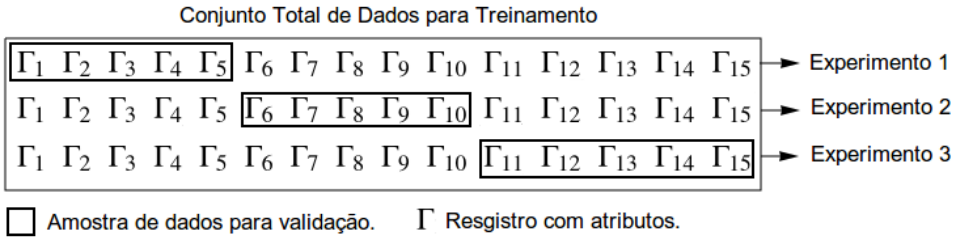



Fonte: Adaptado de Silva et al. (2010).

A **2ª** técnica de divisão de dados, particiona os dados para treinamento e validação em subconjuntos, $T$ e $V$, onde todos os registros alternam entre $T$ e $V$, que nas palavras de Russel e Norvig (2013) podemos "obter uma estimativa precisa usando uma técnica chamada Validação Cruzada com K-Repetições", usando valores de 5 ou 10 para o divisor $K$:

$$\frac{Dados}{K} = S \quad \text{Total de subconjuntos para validação e treinamento.} \tag{3.17}$$

Figura 25: Visão dos Dados em K Treinamentos e Validações pela Equação 3.17




Fonte: Adaptado de Silva et al. (2010).

Na Equação 3.16 e Figura 24 ou Equação 3.17 e Figura 25 - destaca Silva et al. (2010), que a VCD processa melhor junto a RNPM em termos de acurácia, usando posições diferentes

de entrada de dados e validação, encontrando durante o treinamento uma melhor generalização do conhecimento ao problema tratado.

O método de VCD busca validar a acurácia, alternando partes do conjunto de amostra de dados, sendo o "dataset"preferencialmente dividido em K subconjuntos, conforme explica Russel e Norvig (2013): "os valores populares de K são 5 e 10 - o suficiente para dar uma estimativa que é estatisticamente provável que seja precisa". Então, usando de VCD ou "10-Fold Cross-Validation", processando a Equação 3.2 com o conjunto os dados da Figura 6, temos:

$$Dados_{mxn} = \begin{bmatrix} \vec{T}_{1,1} & \vec{V}_{1,2} \\ \vec{T}_{2,1} & \vec{V}_{2,2} \\ \vec{T}_{3,1} & \vec{V}_{3,2} \\ \vdots & \vdots \\ \vec{T}_{10,1} & \vec{V}_{10,2} \end{bmatrix} \tag{3.3}$$

$\vec{T}$: subconjunto de dados para treinamento. $\vec{V}$: subconjunto de dados para validação.

Usando como entrada de dados a Matriz 3.3, o algoritmo para validação de acurácia, VCD com K-validações ou "K-Fold Cross-Validation", compreende:

**Algoritmo 3:** Validação Cruzada com K-Repetições($Dados_{mxn}$, $RNPM$, $K = 10$)

**Entrada:** $Dados_{mxn}$: Matriz de dados do problema dividida em K-Repetições.
RNPM: Rede Neural Perceptron Multicamadas.
$\vec{\Omega}$: Vetor de saída de acurácia da RNPM.
**Saída:** Acurácia Validada com K-Repetições.

**início**
  $\vec{\Omega}_{i..K} = 0$;
  **para** $(i = 0;\ i \leq K;\ i++)$ **faça**
    $RNPM$ = Back-Propagation($RNPM$, $Dados(T[K,1])$);
    $\Omega_K$ = Acurácia($RNPM$, $Dados(V[K,2])$);
  **fim**
  Acurácia_Validada = $\left(\frac{1}{K}\sum_{i=1}^{K}\Omega_i\right)$;
**fim**
**retorna** *Acurácia_Validada.*

Fonte: Produção do Autor (2017).

Conclui-se que o Algoritmo 3 usa subconjuntos de dados de treinamento para validação da acurácia de RNPM, executando a "K-Fold Cross-Validation", buscando diversificar os dados para treinamento e validação, visando a generalização da aprendizagem com permuta de registros entre os subconjuntos de dados.

## 3.7 Algoritmo de Uso da RNPM Após o Aprendizado

A solução de um problema já aprendido por um RNPM, deve ser generalizado ou induzido a outras instâncias de problemas desconhecidos, mas pertencente ao mesmo domínio de dados, onde é capaz, segundo Witten et al. (2011, p. 29), de "buscar através de um espaço de conceito possível, alguma descrição de conceito que se adapta aos dados".

Por isso que Luger (2013) destaca que os algoritmos de aprendizagem de máquina e uma RNPM, através de uma abordagem conexionista, é treinada para capturar adequadamente a resolução de alguma tarefa, em vez de ser programada diretamente para dar a resposta.

O uso de uma RNPM após o aprendizado, para a solução de novos problemas ocorre pela expressão $Y_{[\text{saída}]} = \Omega_{[\text{RNPM}]}(\vec{R}_{[\text{entrada de dados}]})$, conforme o Algoritmo 4:

**Algoritmo 4:** Uso de RNPM Pós-Treinamento($\vec{R}$)

**Entrada:** $\vec{R}$: Registro com atributos do problema [$x_0$, $x_1$, $x_2$, …, $x_n$].
**Saída:** $Y_{\text{saída}}$: Resposta da RNPM ao problema.

**início**
  $\vec{\theta}_1$, $\vec{\theta}_2$, $\vec{\theta}_3$; // Camadas ocultas $1^a$ *e* $2^a$, e de saída $3^a$ com os perceptrons.
  $\vec{\times}_1$, $\vec{\times}_2$, $\vec{\times}_3$; // Saída dos perceptrons das camadas $1^a$, $2^a$ *e* $3^a$.
  $w_{3xn}$ = abrir aprendizagem($\Omega_{\text{pesos}}$); // Para as camadas $1^a$, $2^a$ *e* $3^a$.
  $b_{3xn}$ = abrir aprendizagem($\Omega_{\text{bias}}$); // Para as camadas $1^a$, $2^a$ *e* $3^a$.
  **para** ($i = 0$; i < $|\vec{\theta}_1|$; $i++$) $(\vec{\theta}_1).(\vec{\times})[i] = tanh\left(\sum_{j=0}^{|\vec{D}|}(\vec{x}_j \vec{w}_{(0,j)}) + b_{(0,i)}\right)$; **fim**
  **para** ($i = 0$; i < $|\vec{\theta}_2|$; $i++$) $(\vec{\theta}_2).(\vec{\times})[i] = tanh\left(\sum_{j=0}^{|\vec{\theta}_1|}((\vec{\theta}_1).(\vec{\times}_j \vec{w}_{(1,j)})) + b_{(1,i)}\right)$; **fim**
  $(\vec{\theta}_3).(\vec{\times})[0] = tanh\left(\sum_{j=0}^{|\vec{\theta}_3|}((\vec{\theta}_2).(\vec{\times}_j \vec{w}_{(2,j)})) + b_{(2,0)}\right)$;
  **se** $((\vec{\theta}_3).(\vec{\times})[0] > 0)$ Y = 1 **senão** Y = 0; **fim**
  **retorna** $Y_{saída}$;
**fim**

Fonte: Produção do Autor (2017).

O Algoritmo 4 descreve o uso de uma RNPM em ambiente de produção. Uma vez treinada, a RNPM consegue resolver outros problemas não vistos em época de treinamento, desde que pertencente ao mesmo domínio de problema.

Conclui-se entendendo que a RNPM treinada, é capaz de generalizar o conhecimento aprendido para resolver novos problemas, com a capacidade de solucionar tarefas de classificação de dados.

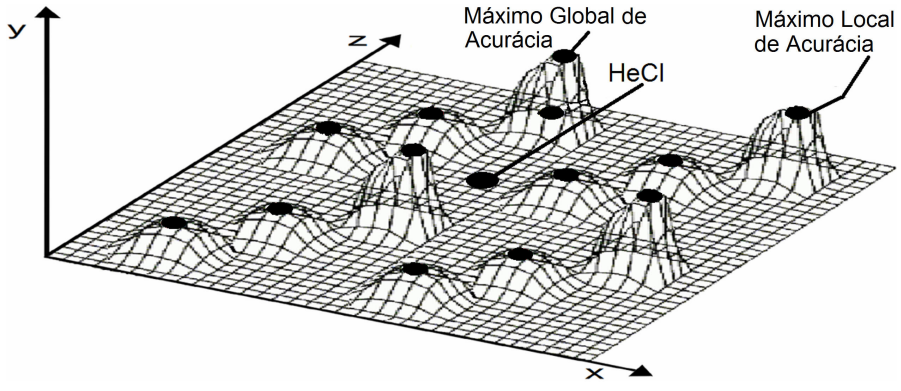
# 4 Proposta: Heurística para Calcular Pesos e Bias Iniciais (HeCI)

O algoritmo heurístico desenvolvido nesta pesquisa é nomeado de Heurística para Calcular Pesos e Bias Iniciais (HeCI). A HeCI computa os dados de treinamento de RNPM e retorna pesos e bias iniciais para o ponto de partida do "Back-Propagation", controla parcialmente o número de épocas de treinamento da RNPM e altera pesos para sair de mínimos locais - diversificando a busca por mais acurácia. A idéia central da HeCI:

- Ler dados de treinamento de RNPM e retornar pesos e bias em local estratégico dentro do espaço $R^n$ para o ponto de partida do "Back-Propagation".
- Controlar parcialmente o número de épocas de treinamento de RNPM pelo "Back-Propagation", para sair de mínimos locais e diversificar a busca por mais acurácia.
- Provocar o aumento de acurácia de RNPM, aproximando ou encontrando o ótimo global, pelo cálculo inicial de pesos e bias, e pela diversidade de busca sobre os dados de treino.

Sendo os dados de treinamento de RNPM descrito em forma de $Matriz_{mxn}$, projetados no espaço $R^n$, a HeCI lê esses dados e retorna pesos e bias em local estratégico. A Figura 26 ilustra o ponto de retorno da HeCI dentro do espaço $R^n$:

Figura 26: Ideia do Cálculo da HeCI para Retorno de Pesos e Bias




Fonte: Produção do Autor (2017).

A Figura 26 ilustra os dados de treinamento da RNPM plotados no espaço $R^n$, apresenta os mínimos locais e a posição de retorno de pesos e bias calculados pela HeCI.

A HeCI é composta por dois Métodos: HeCI_Start (calcula e retorna pesos e bias iniciais) e HeCI_Control (controla parcialmente o número de épocas de treinamento e altera pesos para sair de mínimos locais - diversificando a busca por mais acurácia).

O pseudocódigo do algoritmo da HeCI:

**Algoritmo 5:** HeCI(set: inteiro, dataset: $matrix_{[ixj]}$)

**Entrada:** set: Variável de configuração para uso das Estratégias 1 ou 2.
*dataset*: Dados para o treinamento de RNPM.

1 Comentário: { $Hweights[][]$ *e* $Hbias[][]$ : Matrizes de Pesos e Bias iniciais.
*minXi* e *maxXi* : Valores de mínimo e máximo de épocas de treinamento.
*divXi* : Divisor de "maxXi" para controle parcial de épocas de treinamento.

2 [ $Hweights[][]$ ; $Hbias[][]$ ] = HeCI_Start(set: inteiro, dataset: $matrix_{[ixj]}$)

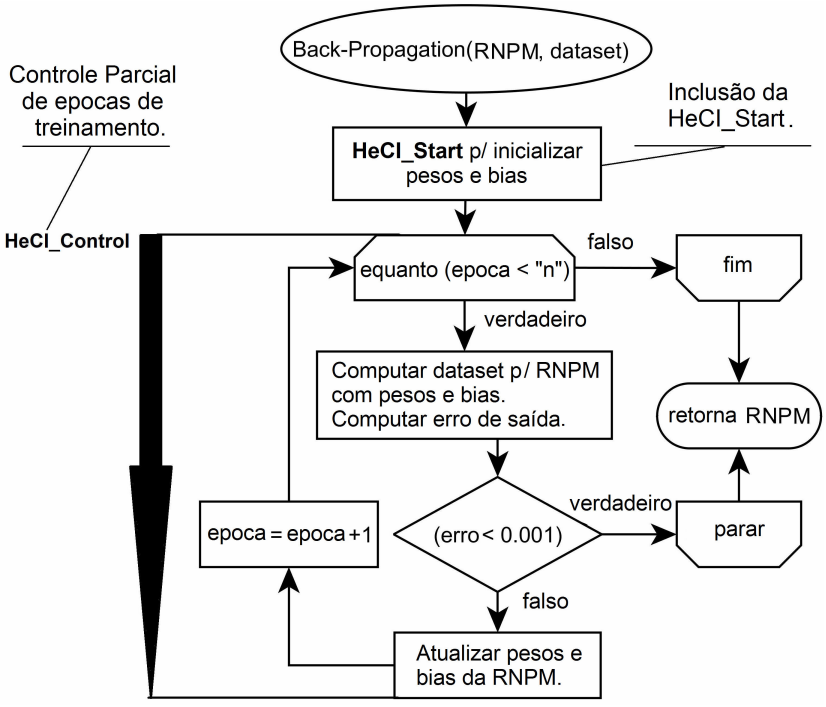
3 RNPM = HeCI_Control(Hweights[][], Hbias[][], minXi, divXi, maxXi)

4 **retorna** *RNPM treinada por Back-Propagation com HeCI*

Fonte: Produção do Autor, 2017.

O Algoritmo 5 descreve a HeCI com dois Métodos: HeCI_Start e HeCI_Control. A HeCI é desenvolvida para calcular os pesos e bias iniciais para o "Back-Propagation" treinar a RNPM, controlar o número de épocas de treinamento e sair demínimos locais - diversificando a busca por mais acurácia. A HeCI é inserida no algoritmo "Back-Propagation" em dois pontos destintos, conforme descrito no fluxograma:

Figura 27: Descreve o Fluxograma do "Back-Propagation" com a HeCI




Fonte: Adaptado de Rumelhart et al. (1986) e Faceli et al. (2011) com a inserção da HeCI.

A Figura 27 descreve a inicialização de Pesos e Bias pela HeCI para o "Back-Propagation" treinar a RNPM e o controle parcial do número de épocas de treinamento.

O detalhamento da HeCI em conceitos, modelagem matemática, fluxograma, pseudocódigo e métodos são descritos adiante para entendimento da heurística proposta.

A HeCI usa os conceitos de:

1. Análise de Componentes Principais (PCA);
2. Método dos Mínimos Quadrados (MMQ);
3. Duas configurações estratégicas:
   3.1 Pesos não normalizados por Distribuição Normal Gaussiana;
   3.2 Pesos e Bias normalizados por Distribuição Normal Gaussiana.
4. Controle Parcial do número de épocas de treinamento de RNPM.

A HeCI combina esses conceitos e computa e calcula o ponto inicial de pesos e bias para o "Back-Propagation", controla parcialmente o treinamento de RNPM, de forma que resulte em aproximação ou excepcionalmente o ótimo global, e modifica pesos para sair de mínimos locais - diversificando a busca por mais acurácia.

## 4.1 Análise de Componentes Principais (PCA)

A PCA segundo Jolliffe (2002), Johnstone e Lu (2009), e Jolliffe e Cadima (2016) lê uma $Matriz_{mxn}$ numérica e retorna os elementos principais com máxima variação e não correlacionados. A Decomposição por Valor Singular (DVS) é calculada pela Álgebra Linear conforme Lay (2013), para encontrar os componentes principais de uma matriz.

A HeCI usa PCA por causa da definição de Haykin (2001, p. 433):

> Um problema comum de reconhecimento estatístico de padrões é a seleção das características ou extração de características. A seleção de características se refere a um processo no qual o espaço de dados é transformado em um espaço de características que, em teoria, tem exatamente a mesma dimensão que o espaço original de dados. A análise de componentes principais (...) é, portanto, a escolha correta.

A HeCI computa a PCA para uma matriz $B_{mxn}$ de dados de treinamento de RNPM. Para esse processamento é usado os Processos 1, 2 e 3:

- **Processo 1.** Computa a normalização de dados por Z-Score:

Média Aritmética: $\bar{x} = \sum_{j=1}^{n} x_j$

Desvio Padrão: $s = \sqrt{\frac{\sum_{j=1}^{n} \left(x_{ij} - \bar{x}\right)^2}{n - 1}}$

$$zscore(B_{mxn}) : z_{ij} = \frac{x_{ij} - \bar{x}}{s} \qquad (4.1)$$

$z_{ij}$ : Elemento da matriz normalizado.
$x_{ij}$: Elemento da matriz.
$n$: Total de elementos.

- **Processo 2.** Computa a transformação da matriz $B_{m\text{x}n}$ em uma matriz $A_{n\text{x}n}$ quadrática de tamanho das colunas da matriz $B$ e sem covariância dos dados:

  $B$: Matriz de dados.
  $B^T_{n\text{x}m}[a_{j,i}] = B_{m\text{x}n}[a_{i,j}]$: Matriz Transposta de $B$.
  $N$: Total de linhas da matriz $B$ (dividir por $N-1$ extrai a covariância dos dados).

  $$A = \frac{B^T B}{N-1} \tag{4.2}$$

  Multiplicação das matrizes $B^T B$:

$$B^T_{3x4}\begin{bmatrix} x_a & x_a & x_a & x_a \\ x_b & x_b & x_b & x_b \\ x_c & x_c & x_c & x_c \end{bmatrix} * B_{4x3}\begin{bmatrix} x_a & x_b & x_c \\ x_a & x_b & x_c \\ x_a & x_b & x_c \\ x_a & x_b & x_c \end{bmatrix} = A_{3x3}\begin{bmatrix} \sum_{\substack{a=1}}^{4} x_a x_a & \sum_{\substack{a=1\\b=1}}^{4} x_a x_b & \sum_{\substack{a=1\\c=1}}^{4} x_a x_c \\ \sum_{\substack{b=1\\a=1}}^{4} x_b x_a & \sum_{\substack{b=1}}^{4} x_b x_b & \sum_{\substack{b=1\\c=1}}^{4} x_b x_c \\ \sum_{\substack{c=1\\a=1}}^{4} x_c x_a & \sum_{\substack{c=1\\b=1}}^{4} x_c x_b & \sum_{\substack{c=1}}^{4} x_c x_c \end{bmatrix}$$

- **Processo 3.** Computa a Decomposição de Valor Singular (DVS) do resultado da Equação 4.2, sendo a matriz $A_{n\text{x}n}$ os dados de entrada para a DVS:

  $$DVS(A_{n\text{x}n}) = \mathbb{U}\mathbb{D}\mathbb{V}^T \tag{4.3}$$

  $\mathbb{U}$: uma Matriz Ortogonal.
  $\mathbb{D}$: uma Matriz Diagonal.
  $\mathbb{V}^T$: uma Matriz Ortogonal Transposta.

Segue-se com o cálculo da matriz Ortogonal $\mathbb{U}$. Sendo o resultado da Equação 4.2 a matriz $A_{3x3} = \begin{bmatrix} a & b & c \\ d & a & e \\ f & g & a \end{bmatrix}$, calcula-se o determinante:

$$det(A - \lambda I) = \begin{bmatrix} a & b & c \\ d & a & e \\ f & g & a \end{bmatrix} = \begin{bmatrix} a-\lambda & b & c \\ d & a-\lambda & e \\ f & g & a-\lambda \end{bmatrix} \tag{4.4}$$

$\lambda$: incógnita.

$I$: Matriz Identidade $I_{3x3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, de mesmo tamanho da matriz $A_{3x3}$.

Aplicando a regra de Sarrus, segundo Kolman e Hill (2014), na Matriz (4.4), repete-se as duas primeiras colunas no lado direito da matriz e calcula-se as diagonais Principal e Secundária:

$$S = Sarrus(A - \lambda I)_{Diag.Principal} - Sarrus(A - \lambda I)_{Diag.Secundria} \tag{4.5}$$

Resultado da regra de Sarrus:

$$S = Sarrus(det(A - \lambda I)) = \left[\begin{array}{ccc|cc} a-\lambda & b & c & a-\lambda & b \\ d & a-\lambda & e & d & a-\lambda \\ f & g & a-\lambda & f & g \end{array}\right] \tag{4.6}$$

$$S_{Diag.Principal} = \sum_{principal=1}^{n} x_i y_j z_k \tag{4.7}$$

$$S_{Diag.Secundria} = \sum_{secundaria=1}^{n} x_i y_j z_k \tag{4.8}$$

$$det(S) = S_{Diag.Principal} - S_{Diag.Secundria} \tag{4.9}$$

O determinante da Equação 4.9 é igual a um polinômio de ordem três, e o resultado são as raízes $\lambda_1,\ \lambda_2,\ \lambda_3$.

Consequentemente, $\lambda_1,\ \lambda_2,\ \lambda_3$ são Autovalores da matriz $A$. Pela teoria dos Sistemas Lineares, é feito o cálculo dos Autovetores associando os Autovalores, ou seja, para o vetor $\vec{v} = [x,\ y,\ z]$, calcula-se:

$$(A - \lambda I)\vec{v} = 0 \tag{4.10}$$

O desenvolvendo a Equação 4.10, resulta em um sistema linear:

$$A_{3x3} = \begin{bmatrix} a-\lambda & b & c \\ d & a-\lambda & e \\ f & g & a-\lambda \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.11}$$

Com isso, calcula-se os Autovetores $v_1,\ v_2,\ v_3$ associados a cada um dos Autovalores $\lambda_1,\ \lambda_2,\ \lambda_3$. Para demonstração do modelo, considere que $\lambda_1 = 2a$, então calcula-se:

$$A_{3x3} = \begin{bmatrix} a-2a & b & c \\ d & a-2a & e \\ f & g & a-2a \end{bmatrix} = \left\{ \begin{bmatrix} -a & b & c \\ d & -a & e \\ f & g & -a \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right. \tag{4.12}$$

A Equação 4.12 resulta em um Sistema Linear:

$$\begin{cases} -ax + by + cz = 0 \\ dx - ay + ez = 0 \\ fx + gy - az = 0 \end{cases} \tag{4.13}$$

Resolvendo o Sistema 4.13, temos: $x = \alpha_1,\ y = \beta_1,\ z = \delta_1$. Então, o $\vec{v1} = [\alpha_1,\ \beta_1,\ \delta_1]$ é o Autovetor associado ao Autovalor $\lambda_1$.

Em seguida calcula-se a norma do vetor $\vec{v1}$:

$$\theta = ||v1|| = \sqrt{\sum_{i=1}^{n} (v1)_i^2} \tag{4.14}$$

Para calcular a primeira linha $\vec{u1}$ da matriz Ortogonal $\mathbb{U}$, referente a matriz $A$, divide-se cada elemento de $v1$ pelo $\theta$ da Equação 4.14, que é norma de $v1$. Então temos:

$$\vec{u1} = \left[ \frac{v1_1}{\theta}, \frac{v1_2}{\theta}, \frac{v1_3}{\theta} \right] \tag{4.15}$$

As outras linhas, vetores $\vec{u2}$ e $\vec{u3}$, são calculadas pela mesma sequência de cálculo de $\vec{u1}$, pelas Equações 4.4 até a 4.15. Assim procedendo, temos a matriz Ortogonal:

$$\mathbb{U} = \begin{bmatrix} \vec{u1} \\ \vec{u2} \\ \vec{u3} \end{bmatrix} \tag{4.16}$$

Para calcular a matriz Diagonal $\mathbb{D}$, referente a matriz $A$, multiplica-se:

$$\mathbb{D} = \mathbb{U}^{-1} A \mathbb{U} \tag{4.17}$$

$\mathbb{U}$: Matriz Ortogonal calculada na Equação 4.16.
$A$: Matriz quadrática calculada na Equação 4.2.

Para calcular a matriz Ortogonal Transporta $\mathbb{V}^T$ referente a matriz $A$, faz-se:

$$\mathbb{V}^T = U^T \tag{4.18}$$

$\mathbb{U}$: Matriz Ortogonal calculada na Equação 4.16.

Com a conclusão dos Processos 1, 2 e 3, sobre a matriz $A_{nxn} = \mathbb{U}\mathbb{D}\mathbb{V}^T$, então tem-se que os Componentes Principais são os valores da matriz $\mathbb{V}^T$ da Equação 4.18.

A HeCI computa a PCA para cada uma das classes, buscando reconhecer o padrão para classificar os dados. Para exemplificar o processamento da HeCI, considere três Classes K, Y e $W$, sendo seus componentes principais as matrizes $K_{3x3}$, $Y_{3x3}$ e $W_{3x3}$:

$$K_{3x3} = \begin{bmatrix} a_k & b_k & c_k \\ d_k & e_k & f_k \\ g_k & h_k & i_k \end{bmatrix} \qquad Y_{3x3} = \begin{bmatrix} a_y & b_y & c_y \\ d_y & e_y & f_y \\ g_y & h_y & i_y \end{bmatrix} \qquad W_{3x3} = \begin{bmatrix} a_w & b_w & c_w \\ d_w & e_w & f_w \\ g_w & h_w & i_w \end{bmatrix}$$

A HeCI separa cada uma das classes em matrizes separadas, acrescenta duas colunas a direita de cada matriz, uma coluna igual a 1 (valor para cálculo do Bias) e outra coluna igual a 4

(porque a Tangente Hiperbólica de 4 é igual a 0,999) para a resposta do perceptron. O perceptron retorna $[0 \geq x \leq 1]$, então 0,999 é igual a classificação dos dados na respectiva Classe.

As matrizes $K_{3x3}$, $Y_{3x3}$ e $W_{3x3}$ passam a ter a estrutura numérica:

$$K_{3x3} = \begin{bmatrix} a_k & b_k & c_k & 1 & 4 \\ d_k & e_k & f_k & 1 & 4 \\ g_k & h_k & i_k & 1 & 4 \end{bmatrix} \quad Y_{3x3} = \begin{bmatrix} a_y & b_y & c_y & 1 & 4 \\ d_y & e_y & f_y & 1 & 4 \\ g_y & h_y & i_y & 1 & 4 \end{bmatrix} \quad W_{3x3} = \begin{bmatrix} a_w & b_w & c_w & 1 & 4 \\ d_w & e_w & f_w & 1 & 4 \\ g_w & h_w & i_w & 1 & 4 \end{bmatrix}$$

Na sequência, faz-se a união das três matrizes para cada Classe referente as matrizes $K$, $Y$ e $W$, sendo que registros de outras classes recebe o valor zero na última coluna (porque a Tangente Hiperbólica de 0 é igual a 0) para a resposta do perceptron. As matrizes são transformadas em:

$$K_{5x9} = \begin{bmatrix} a_k & b_k & c_k & 1 & \mathbf{4} \\ d_k & e_k & f_k & 1 & \mathbf{4} \\ g_k & h_k & i_k & 1 & \mathbf{4} \\ a_y & b_y & c_y & 1 & 0 \\ d_y & e_y & f_y & 1 & 0 \\ g_y & h_y & i_y & 1 & 0 \\ a_w & b_w & c_w & 1 & 0 \\ d_w & e_w & f_w & 1 & 0 \\ g_w & h_w & i_w & 1 & 0 \end{bmatrix} \quad Y_{5x9} = \begin{bmatrix} a_k & b_k & c_k & 1 & 0 \\ d_k & e_k & f_k & 1 & 0 \\ g_k & h_k & i_k & 1 & 0 \\ a_y & b_y & c_y & 1 & \mathbf{4} \\ d_y & e_y & f_y & 1 & \mathbf{4} \\ g_y & h_y & i_y & 1 & \mathbf{4} \\ a_w & b_w & c_w & 1 & 0 \\ d_w & e_w & f_w & 1 & 0 \\ g_w & h_w & i_w & 1 & 0 \end{bmatrix} \quad W_{5x9} = \begin{bmatrix} a_k & b_k & c_k & 1 & 0 \\ d_k & e_k & f_k & 1 & 0 \\ g_k & h_k & i_k & 1 & 0 \\ a_y & b_y & c_y & 1 & 0 \\ d_y & e_y & f_y & 1 & 0 \\ g_y & h_y & i_y & 1 & 0 \\ a_w & b_w & c_w & 1 & \mathbf{4} \\ d_w & e_w & f_w & 1 & \mathbf{4} \\ g_w & h_w & i_w & 1 & \mathbf{4} \end{bmatrix}$$

O próximo passo é calcular os coeficiente comuns para cada matriz, onde destaca-se a classificação de uma classe entres as outras classes. O cálculo dos coeficiente para a aproximação de $[4 \forall \mathit{Registros} \in \mathit{Classe}]$ e $[0 \forall \mathit{Registros} \notin \mathit{Classe}]$ é processando pelo Método dos Mínimos Quadrados.

## 4.2 Método dos Mínimos Quadrados (MMQ)

Segundo Markovsky e Huffel (2007), Chapra e Canale (2009) e Silva et al. (2015), o MMQ calcula os coeficiente multiplicadores comuns para as colunas de uma matriz $X_{mxn}$ de dados $\in \mathbb{R}$. As operações de multiplicação, transposta e inversa são calculadas com a matriz de dados, e um vetor $R \in \mathbb{R}$ de igualdade ou rótulo (resposta do perceptron) para os dados.

A HeCI usa MMQ por causa da possibilidade de calcular coeficiente de matrizes não quadráticas, ou seja, $Matriz_{mxn}$. O cálculo desses coeficientes é processado pela Equação:

$$Coeficientes = (X^T * X)^{-1} * (X^T * R) \tag{4.19}$$

$X$: Matriz de dados.
$R$: Vetor com o rótulo das classes para os dados (Resposta do perceptron).

Então, preparando os dados das matrizes referente as Classes $K$, $Y$ e $W$, para o cálculo do MMQ, transforma-se cada matriz em duas. Transformando a matriz $K_{5x9}$ em $X_{mxn}$ e $R_{mx1}$:

$$X_{mxn} = \begin{bmatrix} a_k & b_k & c_k & 1 \\ d_k & e_k & f_k & 1 \\ g_k & h_k & i_k & 1 \\ a_y & b_y & c_y & 1 \\ d_y & e_y & f_y & 1 \\ g_y & h_y & i_y & 1 \\ a_w & b_w & c_w & 1 \\ d_w & e_w & f_w & 1 \\ g_w & h_w & i_w & 1 \end{bmatrix} \quad (4.20) \qquad R_{mx1} = \begin{bmatrix} 4 \\ 4 \\ 4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (4.21)$$

Aplicando a Equação 4.19 a matriz $X_{4x9}$ (4.20) e ao vetor $R_{1x9}$ (4.21), temos:

$$coeficientes\ K = \vec{\Delta}_1 = [c_1,\ c_2,\ c_3,\ c_4], \qquad \text{Sendo o } Bias\ =\ c_4 \quad (4.22)$$

Repetindo o cálculo do MMQ, para as matrizes $Y_{5x9}$ e $W_{5x9}$, aplicando a Equação 4.19, temos:

$$coeficientes\ Y = \vec{\Delta}_2 = [c_1,\ c_2,\ c_3,\ c_4], \qquad \text{Sendo o } Bias\ =\ c_4 \quad (4.23)$$

$$coeficientes\ W = \vec{\Delta}_3 = [c_1,\ c_2,\ c_3,\ c_4], \qquad \text{Sendo o } Bias\ =\ c_4 \quad (4.24)$$

Conclui-se com a montagem da matriz de Pesos e Bias de cada vetor $\vec{\Delta}$:

$$Pesos\ e\ Bias = \begin{bmatrix} \vec{\Delta}_1 \\ \vec{\Delta}_2 \\ \vec{\Delta}_3 \end{bmatrix} \quad (4.25)$$

Portanto, a matriz de Pesos e Bias 4.25 contêm os Pesos e Bias iniciais.

## 4.3 Duas Configurações Estratégicas

Segundo Gebser et al. (2013) e Russel e Norvig (2013), a heurística pode usar alguma configuração estratégica para melhor customização da busca por solução próxima ou exata ao ótimo global. Partindo dessa definição, a HeCI usa duas estratégias distintas, para a matriz de Pesos na camada oculta da RNPM:

1. Não normalizar a matriz de Pesos com Distribuição Normal Gaussiana, usando a Função 4.26 de Densidade de Probabilidade; ou

2. Normalizar a matriz de Pesos com Distribuição Normal Gaussiana, usando a Função 4.26 de Densidade de Probabilidade.

Essas duas estratégias foram desenvolvidas por causa da observação em laboratório do comportamento geométrico e estatístico dos dados de treinamento da RNPM.

Para alguns "Dataset's", os coeficientes do MMQ sobre a matriz de Componentes Principais, igualando-os aos Pesos e Bias iniciais, resultaram em alta acurácia da RNPM treinada por "Back-Propagation".

Entretanto, para outros "Dataset's", somente foi alcançada a alta acurácia de RNPM treinada por "Back-Propagation", quando os coeficientes igualando-os aos Pesos e Bias iniciais, foram normalizados pela Distribuição Normal Gaussiana.

A Distribuição Normal Gaussiana entendida em Walpole et al. (2011) e MathWorks (2016), corresponde ao comportamento do efeito agregado de amostras aleatórias e independentes de dados, modelando um conjunto de números em torno de uma tendência. Nesses dados é possível calcular a probabilidade relativa de uma variável aleatória acontecer, com o uso da Função de Densidade de Probabilidade (PDF):

$$PDF(\vec{x}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x_i-\mu)^2} \tag{4.26}$$

$\vec{x}$ e $x_i$: Vetor numérico $\in \mathbb{R}$ e Elemento do vetor.
$e$: Constante Matemática de Euler = $2,7182$.
$\pi$: Constante Matemática Pi = $2,1415$.

$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$: Média Aritmética dos dados do vetor.

$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n\ -\ 1}}$: Desvio padrão dos dados do vetor.

A HeCI usa a Função 4.26 porque transforma os dados mantendo a sua probabilidade, diminuindo a média, o desvio padrão e a norma do vetor de pesos e bias. Com isso foi evitado a saturação da função de ativação do perceptron, e também capturado a concentração tendenciosa dos dados. Por exemplo, um vetor:

$\vec{v1}$ = [0,0201 0,3104 0,7882 0,4701 0,7933 0,3420 0,7727 0,4908], características:
Média Aritm.: 0,4984 Desvio Padrão: 0,2766 Norma: 1,5885 Média Geom.: 0,3524

Agora, $\vec{v1}$ normalizado pela Função 4.26:

$\vec{v1}$ = [0,3989 0,3802 0,2924 0,3572 0,2912 0,3763 0,2960 0,3537], características:
Média Aritm: 0,3432 Desvio Padrão: 0,0437 Norma: 0,9777 Média Geom.: 0,3407

A normalização do vetor $\vec{v1}$ pela Função 4.26, transforma os dados, diminuindo a média Aritmética e Geométrica, a Norma e o Desvio Padrão, causando melhor concentração dos dados para o descobrimento de um padrão. O uso da normalização dos pesos e bias iniciais resultou em maior acurácia em alguns "Dataset's" usados nos experimentos, e por outro lado a normalização somente de pesos provocou a saída de mínimos locais.

### 4.3.1 Estratégia 1: Pesos Não Normalizados por Distribuição Normal

A Estratégia "Set 1" da HeCI usa a matriz de Pesos e Bias $\begin{bmatrix} \vec{\Delta}_1 \\ \vec{\Delta}_2 \\ \vec{\Delta}_3 \end{bmatrix}$ calculada na Equação 4.25, sendo os pesos para a camada oculta da RNPM não normalizados pela Função 4.26, a última coluna dos vetores corresponde aos Bias.

Após esses cálculos, são retornados os Pesos não normalizados para a camada oculta, e os Bias normalizados, para o ponto de partida do "Back-Propagation" no treinamento de RNPM.

### 4.3.2 Estratégia 2: Pesos e Bias Normalizados por Distribuição Normal

A Estratégia "Set 2" da HeCI usa uma matriz de Pesos e Bias $\begin{bmatrix} \vec{\Delta}_1 \\ \vec{\Delta}_2 \\ \vec{\Delta}_3 \end{bmatrix}$ calculada na Equação 4.25, sendo todos os Pesos de todas as camadas da RNPM normalizados pela Função 4.26, a última coluna dos vetores corresponde aos Bias.

Após esses cálculos, são retornados os Pesos e os Bias normalizados para todas as camadas, para o ponto de partida do "Back-Propagation" no treinamento de RNPM.

## 4.4 O Retorno de Pesos e Bias para as Camadas da RNPM

A HeCI retorna Pesos e Bias, respectivamente para as Camadas Oculta e de Saída da RNPM:

$$Perceptrons_{mxn} = \begin{bmatrix} peso_{1,1} & peso_{1,2} & \cdots & peso_{1,n-1} & bias_{1,n} \\ peso_{1,1} & peso_{1,2} & \cdots & peso_{1,n-1} & bias_{1,n} \\ peso_{1,1} & peso_{1,2} & \cdots & peso_{1,n-1} & bias_{1,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ peso_{n,1} & peso_{n,2} & peso_{n,3} & peso_{n,n-1} & bias_{n,n} \end{bmatrix} = \text{Pesos e Bias} \begin{bmatrix} \vec{\Delta}_1 \\ \vec{\Delta}_2 \\ \vec{\Delta}_3 \end{bmatrix}$$

Segundo entendimento feito em Haykin (2001, p. 188-205), o "Back-Propagation" atualiza os perceptron da Camada de Saída diretamente, e os da Camada Oculta são atualizados por estimativa de erro, sendo fácil o ajuste numérico na Camada de Saída e difícil na Camada Oculta.

Na Camada de Saída (aprendizagem Supervisionada), é feito o ajuste direto para o valor de resposta do perceptron, pelo cálculo da função de erro; entretanto para os perceptrons da Camada Oculta, o ajuste é feito por retropropagação, estimando um valor conforme os dados de entrada, aproximando uma resposta adequada aos perceptrons da camada seguinte.

Por essa razão, a HeCI optou por retornar os pesos e bias para a Camada Oculta da RNPM, e para a Camada de Saída o valor da Função 4.26, com o parâmetro zero, $PDF(0) = 0,3989$,

referente a densidade de probabilidade da Distribuição Normal de Gauss. O valor "0" foi usado por causa da matriz de Dados de treinamento está normalizado com Z-Score, Equação 4.1, e porque é o centro no eixo das Abcissa do gráfico em formato de Sino da Distribuição Gaussiana.

O Valor de retorno da HeCI para os perceptron da Camada de Saída:

$$Perceptrons_{mxn} = \begin{bmatrix} PDF(0)_{1,1} & PDF(0)_{1,2} & \cdots & PDF(0)_{1,n-1} & PDF(0)_{1,n} \\ PDF(0)_{1,1} & PDF(0)_{1,2} & \cdots & PDF(0)_{1,n-1} & PDF(0)_{1,n} \\ PDF(0)_{1,1} & PDF(0)_{1,2} & \cdots & PDF(0)_{1,n-1} & PDF(0)_{1,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ PDF(0)_{n,1} & PDF(0)_{n,2} & PDF(0)_{n,3} & PDF(0)_{n,n-1} & PDF(0)_{n,n} \end{bmatrix}$$

$PDF(0)$: Função de Densidade de Probabilidade da Distribuição Normal de Gauss, com entrada "0" e saída igual a 0,39894228, conforme Função 4.26.

Conlui-se, que a HeCI de acordo com a estratégia configurada, retorna os Pesos e Bias iniciais normalizados ou não normalizados, para o ponto de partido do "Back-Propagation" processar o treinamento de RNPM, para reconhecimento de padrões e classificar dados.

## 4.5 Controle Parcial do Número de Épocas de Treinamento da RNPM

O objetivo do Controle Parcial é diversificar a busca por solução aproximada ou exata ao ótima global, usando números fracionários do total de épocas de treinamento.

Esse Controle Parcial é por causa do grande número de épocas de treinamento usado pelo "Back-Propagation" e convergência para mínimos locais, segundo entendimento feito em Haykin (2001), Faceli et al. (2011), Russel e Norvig (2013) e Luger (2013).

O padrão do "Back-Propagation" é configurar um número "$n$" de épocas de treinamento e um erro pequeno, por exemplo, Épocas = 9000 e Erro $\leq 10^{-3}$, e treinar a RNPM até que uma das duas condições sejam satisfeitas, então parar o treinamento.

No entanto, o Controle Parcial na HeCI cria um vetor de épocas e verifica a melhor acurácia a cada "$n$" épocas. Esse vetor é calculado dividindo o total de épocas por ”1000”, para parceladamente verificar se a acurácia melhorou ou piorou, modificar os pesos para diversificar a busca por mais acurácia e sair de mínimos locais.

O uso do divisor = ”1000” é devido os experimentos apontarem que até "2000" épocas de treinamento, a função de erro é minimizada, e que depois até ”1000”, o "Back-Propagation" quase sempre fica preso em algum mínimo local. Por isso, após as primeiras "2000" épocas e ao passo de ”1000”, é avaliada a acurácia e diversificado os pesos para sair de mínimos locais. Portanto, em Épocas = "9000" de treinamento para o "Back-Propagation", temos o cálculo do

controle parcial na HeCI:

$$\frac{9000}{1000} = 9 - 1, \text{ Então um vetor de 8 elementos para a HeCI} = [2, 3, 4, 5, 6, 7, 8, 9] * 1000 \quad (4.27)$$

(-1): Menos Um por causa das primeiras 2000 épocas configuradas no "Back-Propagation".

A HeCI, a partir de "2000" e a cada "1000" épocas, conforme a Equação 4.27, verifica a acurácia e diversifica os pesos das Camadas Oculta e de Saída da RNPM, dividindo todos os pesos de todas as camadas por "100" , normalizando com a Função PDF(), Equação 4.26, e retornando o treinamento ao "Back-Propagation" para mais "1000" épocas.

O uso do divisor = "100" é para evitar a saturação do perceptron, diminuindo o valor dos pesos em duas casas decimais.

Com isso, a diversidade de procura por mais acurácia é controlada parcialmente por épocas de treinamento, e o benefício de valores pequenos distribuídos normalmente pela função de Densidade de Probabilidade é usada com êxito.

## 4.6 Notação da HeCI

A notação usada pela HeCI:

Quadro 22: Notação, símbolos, abreviaturas e acronômos usado pela HeCI

| Símbolos, Abreviaturas, Acronômos ou Nomes | Significado |
|---|---|
| RNPM | Variável da Rede Neural Perceptron Multicamadas. |
| RNPM.trainEpochs | Variável de épocas de treinamento. |
| RNPM.acc, RNPM.allLayers.allWeights | Acurácia da RNPM e Pesos de todas as camadas. |
| minXi (mínimo) e maxXi (máximo) | Valores para épocas de treinamento. |
| divXi | Divisor para controle parcial de épocas de treinamento. |
| set | Variável de configuração para uso das Estratégias 1 ou 2. |
| dataset | Dados para o treinamento de RNPM. |
| class, data | Índices usado nas classes rotuladas. |
| dataTrain[], pcaClass[] | Vetor de matrizes de dados de treinamento de RNPM. |
| PCA(), $\Psi$ | Função de Análises de Componentes Principais. |
| SVD | Decomposição Singular de Valor da matriz de dados. |
| $\mu$, $\sigma$ | Média Aritmética e desvio padrão. |
| pcaClass[] | Vetor de matrizes de Análises de Componentes Principais. |
| MMQ, $\Gamma$, $\Theta$ | Método dos Mínimos Quadrados e parâmetros de entrada. |
| PDF | Função de Densidade de Probabilidade da Distribuição. |
| $\Phi$ | Parâmetro de entrada da função PDF(). |
| Hweights, Hbias | Matrizes de Pesos e Bias das camadas da RNPM. |
| vectorXi, idXi | Vetor de épocas de treinamento e índice do vetor. |
| configure | Função para atribuir os pesos e bias iniciais na RNPM. |
| trainEpochs | Épocas de treinamento do "Back-Propagation". |
| epochs, diversify, loop, n, valueAcc | Variáveis dinâmicas do tipo real, inteiro e lógico. |

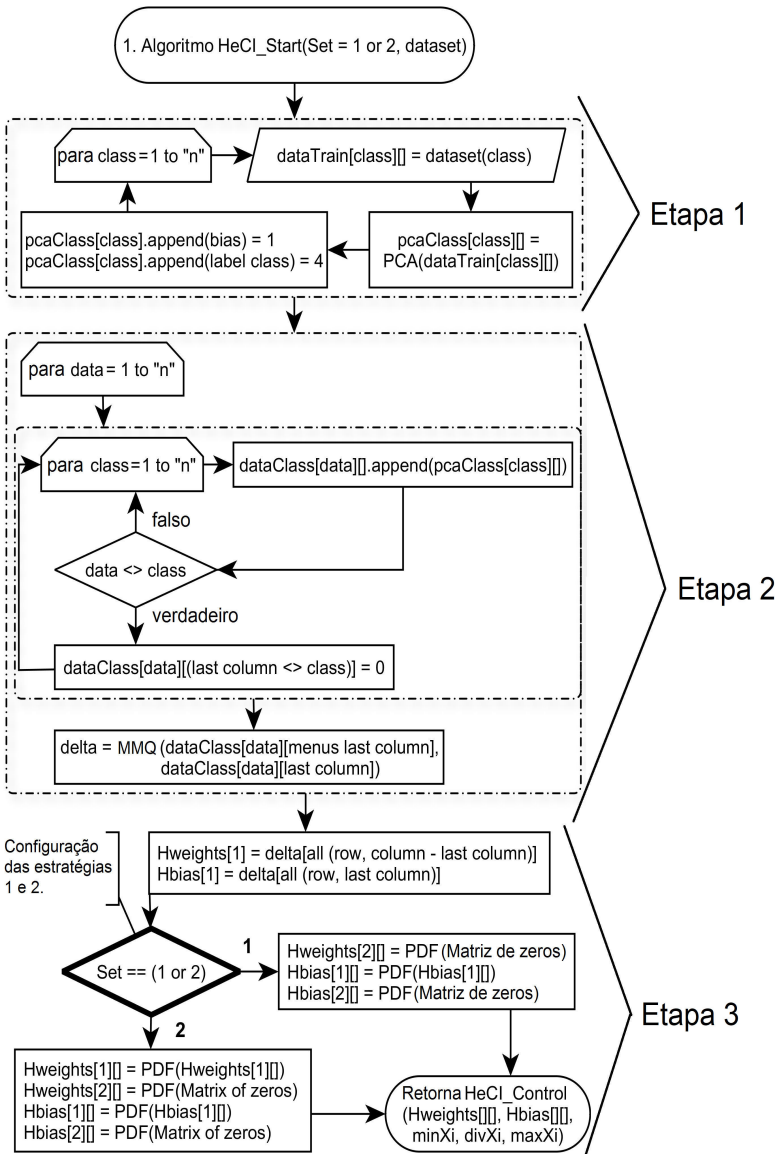
Fonte: Produção do Autor, 2017.

O Quadro 22 descreve a notação usada pela HeCI em fluxogramas e pseudocódigos.

## 4.7 Fluxogramas da HeCI

Os fluxogramas do processamento de dados da HeCI são:

Figura 28: Descrição do Fluxograma da HeCI: Método HeCI_Start

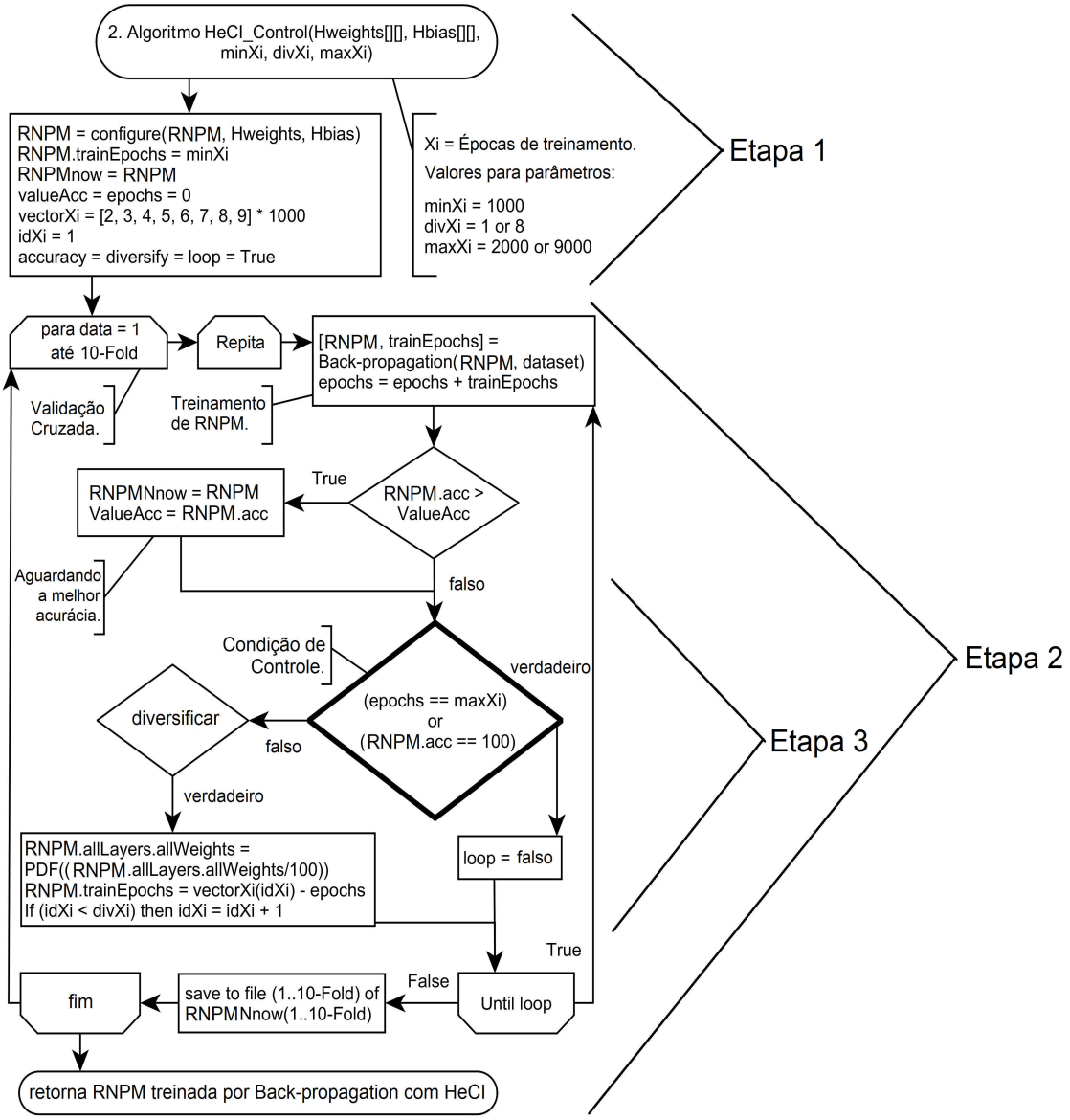



Fonte: Produção do Autor, 2017.

A Figura 28 apresenta o fluxograma do Método HeCI_Start:

- **Etapa 1.** Inicia a separação das classes em matrizes, calcula a Análise de Componentes Principais e acrescenta duas colunas a matriz de cada classe: o Bias e o rótulo dos dados.
- **Etapa 2.** O vetor bidimencial "dataClass" recebe todas as classes. Cada matriz de uma classe recebe todas as outras classes igualando o rótulo dos dados das outras classes a zero e é feito o cálculo dos coeficientes usando o Método dos Mínimos Quadrados.
- **Etapa 3.** Executa-se a Estratégia "Set = (1 ou 2)" e retorna o Método HeCI_Control.

Figura 29: Descrição do Fluxograma da HeCI: Método HeCI_Control




Fonte: Produção do Autor, 2017.

A Figura 29 apresenta o fluxograma do Método HeCI_Control:

- **Etapa 1.** Inicializa-se variáveis, a "RNPM" com os pesos e bias iniciais e o vetor de controle parcial de épocas de treinamento;
- **Etapa 2.** Executa-se a validação cruzada 10-Fold. Após "*n*" épocas de treinamento é decidido parar ou continuar, com base no valor total de épocas ou máxima acurácia. Caso continue a busca por mais acurácia, os pesos de todas as camadas são divididos por 100, normalizados e repassados a RNPM para o "Back-Propagation" continuar o treino.
- **Etapa 3.** Por fim, salva os arquivos de dados de pesos e bias, e retorna uma "RNPM" treinada por "Back-Propagation" usando a HeCI.

## 4.8 Métodos da HeCI: HeCI_Start e HeCI_Control

O pseudocódigo do Método HeCI_Start:

**Algoritmo 6:** HeCI_Start(set: inteiro, dataset: $matriz_{[iXj]}$)

**Retorna:** Carrega a HeCI_Control()

**para** *class* ← 1 **até** *total classes rotuladas* **faça**
  dataTrain[class][$matrix_{[iXj]}$] = dataset(∀ row, ∀ column) ∈ class;
  *função* `Principal_Component_Analysis(`
  Ψ : $dataTrain[class][matrix_{[iXj]}$)
    // A função PCA() usa internamente SVD.
    pcaClass[class][$matrix_{[jxj]}$] = PCA(Ψ);
  **retorna pcaClass**
  // Inserir nos dados de treinamento um multiplicador para o "bias" e
  // a resposta = 4 para o perceptron (porque tanh(4)=0.999).
  pcaClass[class][(∀ row)].append(rigth column = 1);
  pcaClass[class][(∀ row)].append(rigth column = 4);
**fim**
**para** *data* ← 1 **até** *total classes rotuladas* **faça**
  **para** *class* ← 1 **até** *total classes rotuladas* **faça**
    dataClass[data] .append(pcaClass[class][$matriz_{[jxj]}$]);
    **Se** *data* ≠ *class* **Então**
      // Desired answer to other classes
      dataClass[data][(∀ (row, (last column))) ≠ class] = 0;
    **fim**
  **fim**
  *fun*`ção Method_Lest_Squares(`
  Θ*: dataClass[data][∀ (row, column - last column)],*
  *Γ: dataClass[data][∀ (row, last column)]*)
    $\Delta_{[iXj]} = (\Theta^T * \Theta)^{-1} * (\Theta^T * \Gamma)$;
  **retorna** Δ
**fim**
$Hweights[1][matriz_{[iXj]}] = \Delta_{[iXj]}$(∀ (row, column - last column));
$Hbias[1][matriz_{[iXj]}] = \Delta_{[iXj]}$(∀ (row, last column));
*fun*`ção PDF = Probability_Density_Function(`Φ*:* $matriz_{[iXj]}$)
  $\Omega_{matriz_{[iXj]}} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(\Phi-\mu)^2}$, sendo $-\infty < \Phi < +\infty$.
**retorna** Ω
//Hweights[1] and Hbias[1] are of the hidden layer.
//Hweights[2] and Hbias[2] are of the output layer.
**Se** *(Set = 1)* **então**
  // Não normalizar $Hweights[1][matriz]$
**senão** `// (Set = 2)`
  $Hweights[1][matriz]$ = PDF($Hweights[1][matriz]$);
**end**
$Hweights[2][matriz]$ = PDF((*matriz de zeros*));
$Hbias[1][matriz]$ = PDF($Hbias[1][matriz]$);
$Hbias[2][matriz]$ = PDF((*matriz de zeros*));
**retorna** *HeCI_Control(Hweights[][], Hbias[][], minXi, divXi, maxXi);*

Fonte: Produção do Autor, 2017.

O Algoritmo 6 descreve o pseudocódigo do Método HeCI_Start, sendo:

I. As Linhas (1-12) Bloco de repetição para a separação em matrizes das classes rotuladas:

I.I. A Linha (2) as classes rotuladas são separadas em matrizes;

I.II. As Linhas (3-7) extrai os componentes principais de cada classe;

I.III. A Linha (10) cria uma coluna para multiplicação de Bias;

I.IV. A Linha (11) cria uma coluna para a resposta desejada do perceptron.

II. As Linhas (13-20) Bloco de repetição para atribuir zero ao rótulo da classe diferente da classe atual:

II.I. A Linha (15) Vetor de matrizes de todas as classes, cada matriz destaca a classificação de uma classe;

II.II. A Linha (18) Para o vetor de matrizes de todas as classes, a classe diferente da classe em destaque recebe "0" zero.

III. As Linhas (21-25) Bloco de repetição para calcular o Método dos Mínimos Quadrados:

III.I. As Linhas (21-26) Função de cálculo do Método de Mínimos Quadrados.

IV. A Linha (27) Matriz somente com os Pesos para o inicio da "Back-Propagation".

V. A Linha (28) Matriz somente com os Bias para o inicio da "Back-Propagation".

VI. As Linhas (29-31) Função Gaussiana para a normalização de Pesos e Bias.

Dependendo da escolha da Estratégia, configuração "Set $\in$ (1 ou 2)", o Método HeCI_Start computa o processamento:

I. As Linhas (34-38) Função de Densidade de Probabilidade para normalização de pesos da camada oculta da RNPM;

II. As Linhas (39-41) os pesos da camada de saída e todos os bias de todas as camadas são normalizados pela Função de Densidade de Probabilidade Gaussiana.

Na Linha (42), é executado o Método HeCI_Control, repassando os parâmetros:
1) $Hweights[][]$: Matriz de Pesos iniciais.
2) $Hbias[][]$: Matriz de Bias iniciais.
3) $minXi$: Valor mínimo de épocas de treinamento.
4) $maxXi$: Valor máximo de épocas de treinamento.
5) $divXi$: Divisor de "$maxXi$" para controle parcial de épocas de treinamento.

Esses parâmetros são exigidos pela HeCI_Control para atribuição dos valores de Pesos e Bias iniciais a RNPM, controle parcial do número de épocas de treinamento e sair de mínimos locais - diversificando a busca por mais acurácia. O pseudocódigo do Método HeCI_Control:

**Algoritmo 7:** HeCI_Control($Hweights[][]_{[iXj]}$, $Hbias[][]_{[iXj]}$, minXi, divXi, maxXi)

**Retorna:** RNPM trainada por "Back-Propagation" com HeCI.

```
1  RNPM = configure(RNPM, Hweights[][], Hbias[][]);
2  RNPM.trainEpochs = minXi;
3  RNPMNnow = RNPM;
4  valueAcc = epochs = 0;
5  vectorXi = [2, 3, 4, 5, 6, 7, 8, 9] * 1000;
6  idXi = 1;
7  diversify = loop = True;
8  // Validação cruzada igual a 10-Fold.
9  para Fold = 1 até 10 faça
10     repita
11         [RNPM, trainEpochs] = Back-Propagation(RNPM, dataset);
12         epochs = epochs + trainEpochs;
13         se RNPM.acc > valueAcc então
14             valueAcc = RNPM.acc;
15             RNPMNnow = RNPM;
16         fim
17         se (epochs = maxXi) or (RNPM.acc = 100) então
18             loop = False;
19         senão
20             se diversify = True então
21                 RNPM.allLayers.allWeights = PDF(( RNPM.allLayers.allWeights / 100 ));
22                 RNPM.trainEpochs = vectorXi(idXi) - epochs;
23                 se idXi < divXi então
24                     idXi = idXi + 1;
25                 fim
26             fim
27         fim
28     até (loop = True);
29     save file(1..10) de RNPMNnow(1..10-Fold);
30 fim
31 retorna RNPM trainada por Back-Propagation with HeCI
```

Fonte: Produção do Autor, 2017.

O Algoritmo 7 descreve o pseudocódigo do Método HeCI_Control, sendo:

I. As Linhas (1-7) configura variáveis e a RNPM com os Pesos e Bias iniciais.

II. A Linha (11) executa o "Back-Propagation" com validação cruzada 10 vezes para a acurácia.

III. As Linhas (13-16) guarda a RNPM de melhor acurácia.

IV. As Linhas (17-27) controla parcialmente o número de época de treinamento.

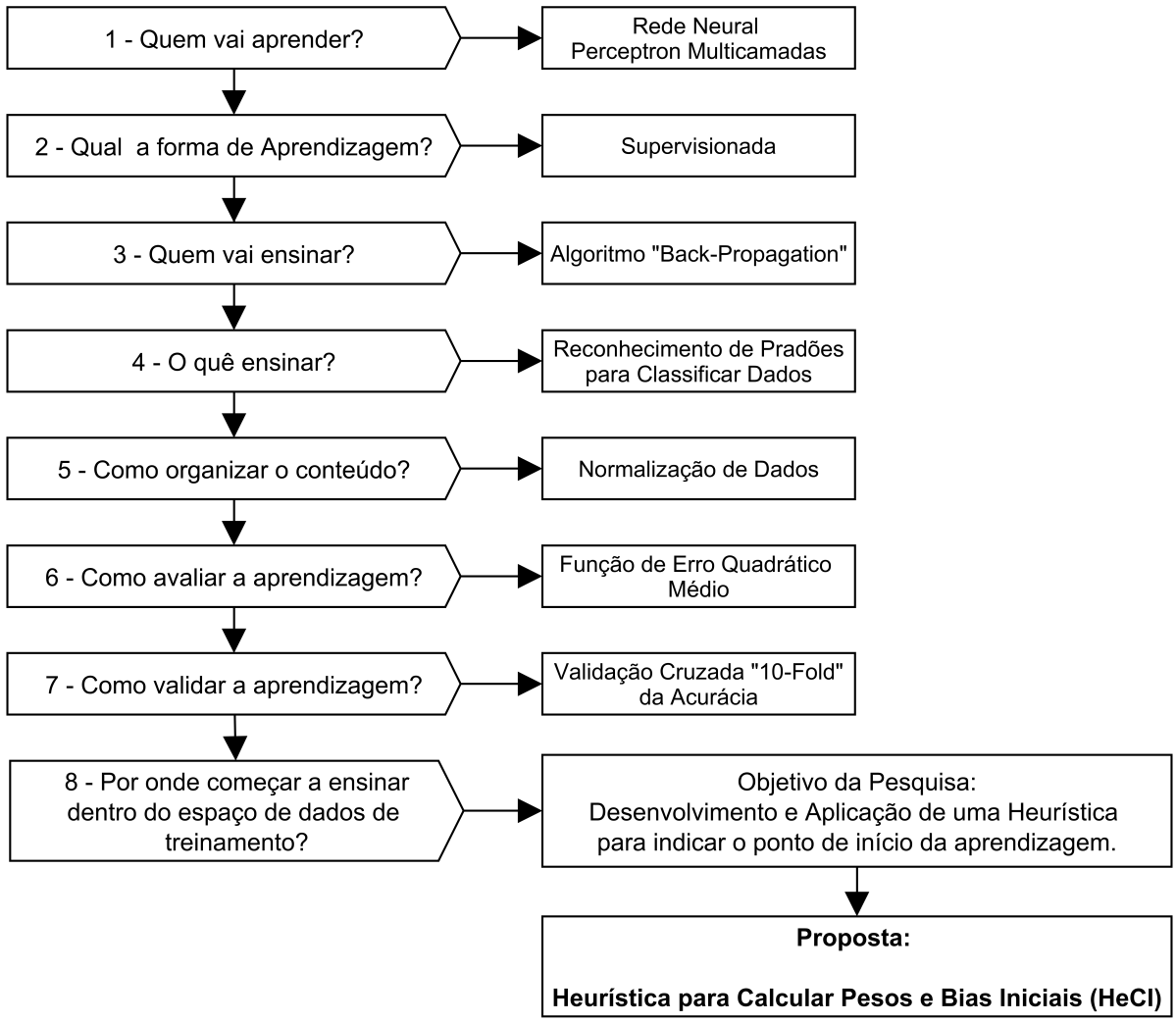
V. A Linha (29) salva em arquivo os pesos e bias da RNPM treinada.

IV. A Linha (31) retorna uma RNPM treinada por "Back-Propagation" + HeCI.

## 4.9 Inclusão da HeCI dentro da Aprendizagem de Máquina (AM)

Fluxo descritivo de contexto de Aprendizagem de Máquina e inclusão da HeCI dentro desse contexto:

Figura 30: Inclusão da HeCI em um Contexto de Aprendizagem de Máquina




Fonte: Adaptado de Haykin (2001), Faceli et al. (2011), Russel e Norvig (2013) e Luger (2013).

A Figura 30 descreve um contexto de Aprendizagem de Máquina para RNPM, conforme o entendimento feito em Haykin (2001), Faceli et al. (2011), Russel e Norvig (2013) e Luger (2013), e a inclusão da HeCI nesse contexto.

Por fim, a HeCI é inserida nesse contexto para computar os dados de treinamento de RNPM e retornar pesos e bias iniciais para o ponto de partida do "Back-Propagation", controlando parcialmente o número de épocas de treinamento da RNPM e alterando pesos para sair de mínimos locais - diversificando a busca por mais acurácia.

# 5 Experimentos Computacionais

Os Experimentos Computacionais foram realizados para investigar e comparar resultados de algoritmos da literatura científica com o algoritmo desenvolvido nessa pesquisa. Nas palavras de Mcgeoch (2012) experimentos com algoritmos é uma abordagem feita em laboratório, enfatizando o controle de parâmetros, o isolamento de componentes-chave, a construção de modelos e a análise estatística.

A investigação realizou o trabalho de laboratório em busca de resposta ao problema de pesquisa conforme Seção 1.1:

> Quais valores inicializar pesos e bias para o "Back-Propagation" convergir para a máxima acurácia?

Sendo a resposta investigada sob a ótica das hipóteses $\mathcal{H}_0$ e $\mathcal{H}_1$, levantadas na Seção 1.3:

> $\mathcal{H}_0$: Inicializar pesos e bias com números Aleatórios ou método Heurístico é equivalente em acurácia e tempo de treinamento.
>
> $\mathcal{H}_1$: Inicializar pesos e bias com números Aleatórios ou método Heurístico é diferente em acurácia e tempo de treinamento.

Dada a conclusão do processamento de dados, os resultados são analisados e comparados pela Estatística Descritiva e Inferencial.

## 5.1 Materiais Usados nos Experimentos

### 5.1.1 O Ambiente Computacional de "Hardware" e "Software" Usado

1. Computador Intel i7-5500U 2.4 GHZ, Memória RAM 16 GB.
2. Sistema Operacional Linux compilado para 64 bits.
3. MATLAB R2016b Licença Estudante, <http://www.mathworks.com>.

### 5.1.2 Normalização de Dados e os "Dataset's" Usados

1. Os dados forma normalizados com Z-Score conforme Faceli et al. (2011), Zaki e Jr (2014) e MathWorks (2016).
2. Os "dataset's" usados nos estudos de caso são "Acute Inflammations" "Banknote Autenticação", "Cancer Breast Wisconsin Diagnostic", "Iris", "Parkinson's"e "Seeds" conforme Czerniak e Zarzycki (2003), Little et al. (2007) e Lichman (2013).

Os "dataset's" são da "University of California, Irvine (UCI)" e detalhados no Quadro 23:

Quadro 23: Conjunto de "Dataset's" Usados nos Experimentos

| Nome do "Dataset" e Área | Descrição | Dados para Aprendizagem de Máquina | Classes |
|---|---|---|---|
| 1. Acute Inflammation (SAÚDE) | Inflamação Aguda em pacientes:<br>Classe 1: Bexiga (30 registros)<br>Classe 2: Neflite (Pelve Renal) (40 registros)<br>Classe 3: Bexiga e Neflite (31 registros)<br>Classe 4: Sem Inflamação (19 registros)<br>**Problema___: Classificação de paciente.** | Matriz 120x6 $\in \mathbb{R}$<br>Origem dos Dados:<br>Sintomas da doença. | 04 |
| 2. Banknote Authentication (NEGÓCIO) | Notas de dinheiro (Euro) falsa ou verdadeira:<br>Classe 1: Nota Verdadeira (762 registros)<br>Classe 2: Nota Falsa (610 registros)<br>**Problema___: Classificação de dinheiro.** | Matriz 1372x4 $\in \mathbb{R}$<br>Origem dos Dados:<br>Imagens digitalizadas de notas. | 02 |
| 3. Breast Cancer Wisconsin Diagnostic (SAÚDE) | Dados de pacientes para diagnóstico de Câncer de Mama:<br>Classe 1: Maligno (212 registros)<br>Classe 2: Benígno (357 registros)<br>**Problema___: Classificação de paciente.** | Matriz 569x30 $\in \mathbb{R}$<br>Origem dos Dados:<br>Imagens digitalizadas de massa da mama. | 02 |
| 4. Iris (BOTÂNICA) | Dados sobre plantas do tipo flores<br>Classe 1: Iris Setosa (50 registros)<br>Classe 2: Iris Versicolor (50 registros)<br>Classe 3: Iris Virgínica (50 registros)<br>**Problema___: Classificação do tipo de planta.** | Matriz 150x4 $\in \mathbb{R}$<br>Origem dos Dados:<br>Dimensões geométricas das plantas. | 03 |
| 5. Parkinson's (SAÚDE) | Dados sobre a Doença de "Parkinson's":<br>Classe 1: Saudável (48 registros)<br>Classe 2: Doente (147 registros)<br>**Problema___: Classificação de paciente.** | Matriz 197x22 $\in \mathbb{R}$<br>Origem dos Dados:<br>Voz dos pacientes (sinais na fala). | 02 |
| 6. Seeds (NEGÓCIO) | Grãos de trigo de três variedades:<br>Classe 1: Kama (70 registros)<br>Classe 2: Rosa (70 registros)<br>Classe 3: Canadian (70 registros)<br>**Problema___: Classificação do tipo de trigo.** | Matriz 210x7 $\in \mathbb{R}$<br>Origem dos Dados:<br>Raio "X" dos grãos. | 03 |

Fonte: Czerniak e Zarzycki (2003), Little et al. (2007) e Lichman (2013).<https://archive.ics.uci.edu/ml>

O Quadro 23 apresenta os "dataset's" usados nos experimentos. Foram realizados seis estudos de casos entre as áreas de Botânica, Negócio e Saúde. Todos esses "dataset's" tem em comum o problema de Classificação de Dados em classes distintas, são em formato de matriz numérica $\in \mathbb{R}$ e possuem específicas amostras de dados de sua respectiva área.

### 5.1.3 Algoritmos de Inicialização de Pesos e Bias

Conforme referenciado no Quadro 17, esta pesquisa usa os seguintes algoritmos de inicialização de pesos e bias para o "Back-Propagation":

1. Heurística para Calcular de Pesos e Bias Iniciais (HeCI) desenvolvida nesta pesquisa e descrita no Capítulo 4.
2. Valores Aleatórios Simétricos (RandS) conforme Pavelka e Prochazka (2004), Singh et al. (2013) e Beale et al. (2016).
3. "Nguyen-Widrow" (InitNW) conforme Ngugen e Widraw (1990) e Pavelka e Prochazka (2004).
4. Valores Aleatórios (Rand) conforme Pavelka e Prochazka (2004), Beale et al. (2016) e MathWork (2016).
5. Valores Aleatórios Pequenos (RandSmall) conforme Pavelka e Prochazka (2004) e Beale et al. (2016).
6. "MidPoint" para os pesos e "RandS" para os bias conforme Beale et al. (2016) e MathWork (2016).

As siglas "RandS", "InitNW", "Rand", "RandSmall" e "MidPoint" são correspondentes aos respectivos algoritmos no ambiente matemático do MATLAB R2016b Licença Estudante. Foram usados esses algoritmos para comparação com a HeCI devido o frequente uso (para inicialização de pesos e bias) registrado na literatura científica.

### 5.1.4 Algoritmos de "Back-Propagation"

Conforme referenciado no Quadro 16, esta pesquisa usa os seguintes algoritmos "Back-Propagation's":

1. "Resiliente Propagation" (TrainRP) conforme Riedmiller e Braun (1993), Riedmiller e Rprop (1994) e Beale et al. (2016).
2. "Bayesian Regularization" (TrainBR) conforme Buntine e Weigend (1991), Kamran et al. (2016) e Beale et al. (2016).
3. "Gradient Descent with Adaptive Learning Rate" (TrainGDA) conforme Plagianakos et al. (2001), Bottou (2012) e Beale et al. (2016).
4. "Gradient Descent with Momentum and Adaptive Learning Rate" (TrainGDX) conforme Dao e Vemuri (2002) e Beale et al. (2016).

As siglas "TrainRP", "TrainBR", "TrainGDA" e "TrainGDX" são correspondentes aos respectivos algoritmos do ambiente matemático do MATLAB R2016b (pacote específico: Aprendizagem de Máquina e Estatística) Licença Estudante.

Configuração usada em todos os "Back-Propagation":

1. Taxa de Aprendizagem igual a "0,01";
2. Condição de parada (erro < "0,001" ou número máximo de épocas de treinamento);
3. Termo "Momentum" igual a "0,09";
4. Validação Cruzada igual a "10-Fold Cross-Validation" conforme descrito em Haykin (2001).

### 5.1.5 HeCI (Parâmetros) e Arquiteturas para a RNPM

A configuração e parâmetros da HeCI (set = configuração estratégica, minXI = número mínimo de iterações, divXi = divisor do número máximo de iterações, maxXi = número maximo de iterações), as respectivas arquiteturas das RNPM's treinadas e os "dataset's" para classificação de dados são descritos a seguir:

1. HeCI(set=1, minXi=1000, divXi=1, maxXi=2000) e RNPM 06-5-4 para classificar o "Dataset" 1.
2. HeCI(set=1, minXi=1000, divXi=1, maxXi=2000) e RNPM 04-4-2 para classificar o "Dataset" 2.
3. HeCI(set=2, minXi=1000, divXi=8, maxXi=9000) e RNPM 30-6-2 para classificar o "Dataset" 3.
4. HeCI(set=2, minXi=1000, divXi=8, maxXi=9000) e RNPM 04-4-3 para classificar o "Dataset" 4.
5. HeCI(set=1, minXi=1000, divXi=8, maxXi=9000) e RNPM 22-9-2 para classificar o "Dataset" 5.
6. HeCI(set=2, minXi=1000, divXi=8, maxXi=9000) e RNPM 07-4-3 para classificar o "Dataset" 6.

Os "Dataset's" 1, 2, 3, 4, 5 e 6 estão descritos no Quadro 23.

A HeCI retorna pesos e bias para uma RNPM de uma camada oculta. A configuração de parâmetros é obtida através da execução de experimentos e observação de resultados.

Todas as RNPM contêm uma Camada Oculta e uma Camada de Saída conforme Haykin (2001) e Samarasinghe (2006), e usam as funções de ativação segundo Beale et al. (2016) e MathWork (2016) respectivamente na Camada Oculta e de Saída: em (1, 2, 3, 6) Tangente Hiperbólica e Sigmóide, em (4) Tangente Hiperbólica e Softmax, e em (5) Softmax e Softmax. O total de perceptron em cada camada da RNPM foi obtido fixando na saída o número de perceptron igual a quantidade de classes e na camada oculta igualando a camada de saída, observando o resultado de acurácia, e depois incrementado ou decrementando 01 perceptron.

### 5.1.6 Função para Medição do Erro em Treinamento

É usada a função:

1. Erro Quadrático Médio conforme explicação de Haykin (2001), Samarasinghe (2006) e MathWork (2016).

### 5.1.7 Métricas para Medição de Resultados

Os resultados são medidos pelas métricas:

1. Médias de Acurácia e de Tempo de Treinamento. O tempo é contado pelos comandos "tic" (início) e "toc" (fim) dentro do ambiente do MATLAB R2016b.
2. Testes de Hipóteses: Teste de "Friedman" com Pós-Teste de "Tukey HSD" e Teste de "Wilcoxon-M-W" descritos em Hammer (2013), Beale et al. (2016) e Nahm (2016).

### 5.1.8 Contabilidade de Amostras de Dados e Algoritmos para os Experimentos

Volume de dados usados e produzidos pelos experimentos computacionais:

1. Total de "Dataset's": 06.
2. Total de algoritmos de inicialização de pesos e bias: 06.
3. Total de algoritmos de "Back-Propagation": 04.
4. Total de Validação Cruzada 10, repetida 30 vezes: 300.
5. Total de variáveis amostrais (Acurácia e Tempo de Treinamento): 02.
6. Total de amostras de aprendizagem de máquina para 01 "dataset": 14400.
7. Total de amostras de aprendizagem de máquina para todos os "dataset's": 86400.

### 5.1.9 Maneira de Apresentação dos Resultados

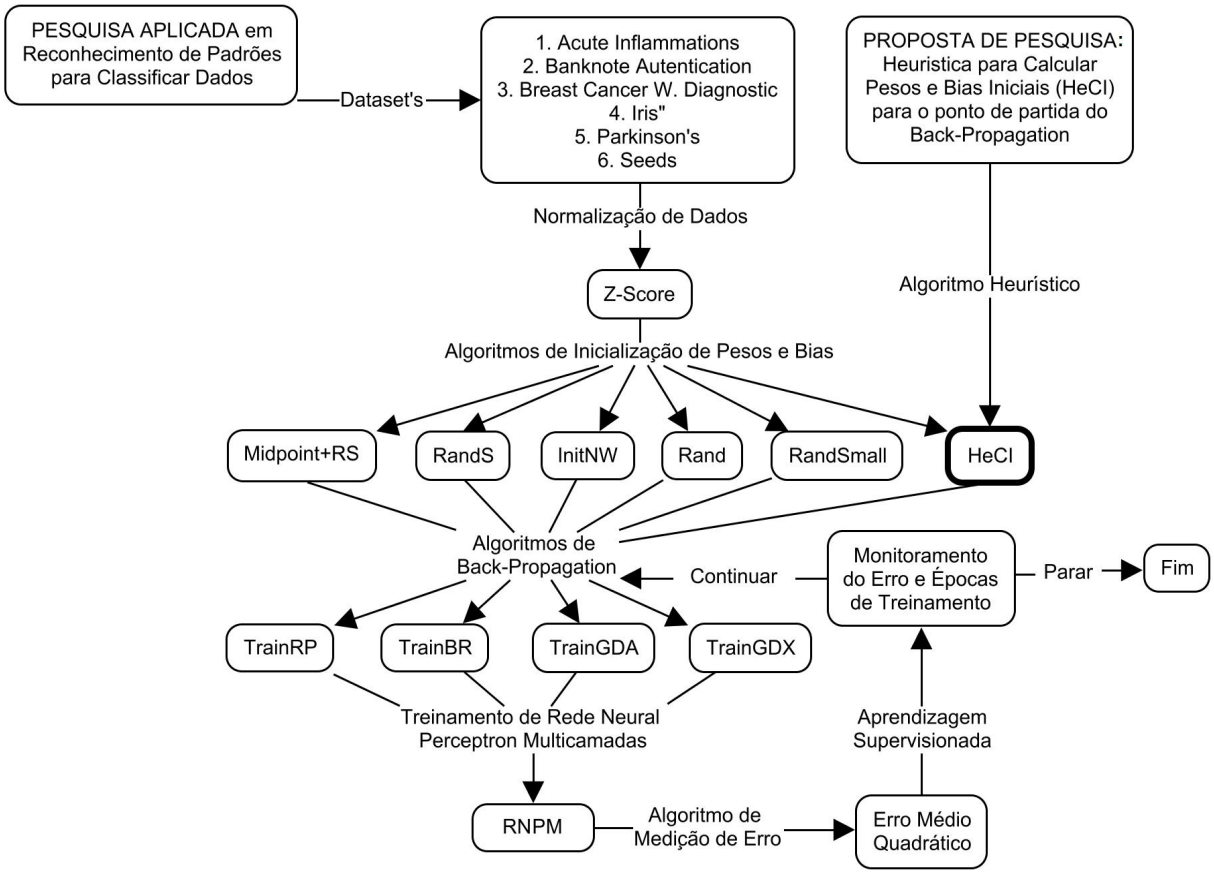
Esta pesquisa analisa os resultados de aprendizagem de máquina da RNPM por "dataset" e pelo conjunto universal de resultados de todos os "dataset's". Então procede-se:

1. Apresentação da Tabela 2 com os resultados individualmente por "dataset's".
2. Realização de testes de hipóteses para os resultados de cada um dos "dataset's".
3. Realização de testes de hipóteses para o conjunto de resultados de todos os "dataset's".
4. Dedução de melhoria significativa, ou seja, um resultado médio sendo subtraído o desvio padrão, e comparando com outros valores, ainda assim é o maior que os valores.

## 5.2 Fluxograma do Trabalho em Laboratório

Os materiais, algoritmos e métricas usados nos experimentos computacionais:

Figura 31: Fluxograma de Trabalho para Execução em Laboratório




Fonte: Produção do Autor (2017).
Nota: As siglas "Rand", "RandS", "RandSmall", "InitNW", "MidPoint+RS (MidPoint+RandSmall)", "TrainRP", "TrainBR", "TrainGDA", "TrainGDX" são algoritmos do ambiente MATLAB 2017b.
A sigla HeCI refere-se ao algoritmo heurístico desenvolvido nesta pesquisa.

A Figura 31 descreve os algoritmos usados nos experimentos e a sequência que foram combinados para a obtenção de resultados. Todos os "dataset's" são usados com todos os algoritmos de inicialização de pesos e bias e todos os "Back-Propagation". Os "dataset's" são normalizados pelo cálculo estatístico de "Z-Score", os algoritmos de inicialização são "Rand", "RandS", "RandSmall", "InitNW", "MidPoint+RS (MidPoint+RandSmall)" e "HeCI"; e os "Back-Propagation" são "TrainRP", "TrainBR", "TrainGDA", "TrainGDX". O trabalho em laboratório é metodicamente executado para cada "dataset" e respectivos algoritmos.

## 5.3 Resultado dos Experimentos: Todos os "Dataset's"

Todos os resultados são apresentados na Tabela 2.

Tabela 2: Resultado dos Experimentos: "10-Fold Cross-Validation" Repetidos 30 Vezes

| Média de Acurácia e de Tempo de Treinamento: Back-Propagation + Algoritmos de Inicialização de Pesos e Bias | | | | | | |
|---|---|---|---|---|---|---|
| "Back-Propagation" | **1.Proposta: HeCI** | 1.RandS | 2.InitNW | 3.Rand | 4.RandSmall | 5.Midpoint+RS |
| | **1. Inflamação Aguda "Acute Inflammations dataset"** | | | | | |
| 1. TrainRP | **100,00± 0,00 0s:061ms** | **100,00± 0,00 0s:048ms** | 99,14 ± 0,23 0s:080ms | 99,78 ± 0,13 0s:075ms | **100,00± 0,00 0s:048ms** | 99,33 ± 0,21 0s:078ms |
| 2. TrainBR | **100,00 ± 0,00 0s:061ms** | 100,00 ± 0,00 0s:053ms | 100,00 ± 0,00 0s:058ms | 100,00 ± 0,00 0s:053ms | 100,00 ± 0,00 0s:062ms | 100,00 ± 0,00 0s:058ms |
| 3. TrainGDA | **100,00 ± 0,00 0s:187ms** | 100,00 ± 0,00 0s:191ms | 99,64 ± 0,15 0s:272ms | 100,00 ± 0,00 0s:182ms | 100,00 ± 0,00 0s:179ms | 99,50 ± 0,18 0s:282ms |
| 4. TrainGDX | **100,00 ± 0,00 0s:189ms** | 100,00 ± 0,00 0s:176ms | 99,39 ± 0,20 0s:293ms | 100,00 ± 0,00 0s:185ms | 100,00 ± 0,00 0s:174ms | 99,64 ± 0,16 0s:286ms |
| | **2. Autenticação Bancária "Banknote Authentication dataset"** | | | | | |
| 1. TrainRP | **99,93 ± 0,01 0s:440ms** | 99,85 ± 0,03 0s:084ms | 99,86 ± 0,03 0s:140ms | 99,81 ± 0,03 0s:085ms | 99,63 ± 0,06 0s:177ms | 99,81 ± 0,04 0s:131ms |
| 2. TrainBR | **100,00 ± 0,00 0s:080ms** | 99,99 ± 0,01 0s:061ms | 99,95 ± 0,01 0s:068ms | 99,98 ± 0,01 0s:064ms | 99,97 ± 0,01 0s:079ms | 99,97 ± 0,01 0s:067ms |
| 3. TrainGDA | 99,85 ± 0,02 0s:733ms | 99,97 ± 0,01 0s:593ms | 99,89 ± 0,02 0s:836ms | 99,97 ± 0,01 0s:625ms | **99,99 ± 0,00 1s:033ms** | 99,85 ± 0,03 0s:777ms |
| 4. TrainGDX | **100,00 ± 0,00 0s:334ms** | 99,94 ± 0,01 0s:277ms | 99,92 ± 0,02 0s:458ms | 99,95 ± 0,01 0s:296ms | 99,98 ± 0,01 0s:447ms | 99,90 ± 0,02 0s:422ms |
| | **3. Diagnóstico de Câncer de Mama "Breast Cancer Wisconsin Diag. dataset"** | | | | | |
| 1. TrainRP | **98,39 ± 0,09 5s:996ms** | 96,59 ± 0,14 0s:196ms | 96,75 ± 0,14 0s:218ms | 96,70 ± 0,14 0s:228ms | 97,49 ± 0,13 0s:076ms | 96,51 ± 0,15 0s:209ms |
| 2. TrainBR | **99,11 ± 0,05 34s:027ms** | 96,45 ± 0,16 0s:379ms | 96,58 ± 0,15 0s:319ms | 96,58± 0,15 0s:277ms | 96,55 ± 0,16 0s:750ms | 96,41 ± 0,16 0s:268ms |
| 3. TrainGDA | **98,04 ± 0,09 6s:130ms** | 96,50 ± 0,16 2s:536ms | 96,45 ± 0,16 3s:273ms | 96,63 ± 0,14 1s:798ms | 96,61 ± 0,11 6s:397ms | 96,57 ± 0,16 3s:466ms |
| 4. TrainGDX | **98,39 ± 0,10 5s:419ms** | 96,47 ± 0,16 0s:500ms | 96,53 ± 0,17 0s:691ms | 96,48 ± 0,15 0s:595ms | 96,06 ± 0,16 1s:184ms | 96,61± 0,16 0s:714ms |
| | **4. Planta "Iris dataset"** | | | | | |
| 1. TrainRP | **98,00 ± 0,25 1s:711ms** | 95,16 ± 0,32 1s:563ms | 95,31 ± 0,34 1s:562ms | 95,24 ± 0,33 1s:794ms | 94,78 ± 0,33 2s:311ms | 95,18 ± 0,32 1s:698ms |
| 2. TrainBR | **99,33 ± 0,12 2s:927ms** | 95,78 ± 0,28 0s:185ms | 95,98 ± 0,27 0s:180ms | 95,69 ± 0,28 0s:147ms | 33,82 ± 0,19 0s:042ms | 95,71 ± 0,30 0s:168ms |
| 3. TrainGDA | 97,33 ± 0,31 1s:601ms | 96,78 ± 0,27 3s:522ms | 96,47 ± 0,27 3s:536ms | 96,82 ± 0,27 3s:787ms | **97,51 ± 0,26 4s:893ms** | 96,40 ± 0,28 3s:743ms |
| 4. TrainGDX | **98,00 ± 0,25 1s:645ms** | 96,78 ± 0,27 2s:535ms | 96,51 ± 0,28 2s:895ms | 96,58 ± 0,28 2s:835ms | 96,49 ± 0,27 3s:517ms | 96,44 ± 0,27 3s:020ms |
| | **5. Doença de Parkinson "Parkinson's dataset"** | | | | | |
| 1. TrainRP | **93,16 ± 0,33 6s:623ms** | 83,58 ± 0,60 0s:082ms | 83,68 ± 0,65 0s:083ms | 83,79 ± 0,65 0s:090ms | 84,61 ± 0,59 0s:103ms | 83,03 ± 0,62 0s:087ms |
| 2. TrainBR | **97,37 ± 0,37 29s:390ms** | 82,96 ± 0,59 0s:188ms | 83,09 ± 0,65 0s:179ms | 83,39 ± 0,66 0s:215ms | 82,98 ± 0,65 0s:252ms | 82,93 ± 0,60 0s:244ms |
| 3. TrainGDA | **88,42 ± 0,38 5s:863ms** | 83,72 ± 0,59 0s:953ms | 83,46 ± 0,58 0s:888ms | 84,35 ± 0,61 1s:270ms | 83,89 ± 0,63 0s:865ms | 84,70 ± 0,59 0s:835ms |
| 4. TrainGDX | **88,42 ± 0,49 5s:865ms** | 83,37 ± 0,61 0s:581ms | 83,35 ± 0,62 0s:533ms | 83,82 ± 0,63 0s:730ms | 83,89 ± 0,64 0s:582ms | 84,47 ± 0,61 0s:438ms |
| | **6. Trigo "Seeds dataset"** | | | | | |
| 1. TrainRP | **95,24 ± 0,43 2s:628ms** | 92,52 ± 0,46 2s:306ms | 92,67 ± 0,48 2s:886ms | 92,25 ± 0,48 2s:350ms | 93,11 ± 0,43 2s:106ms | 92,06 ± 0,54 2s:902ms |
| 2. TrainBR | **99,05 ± 0,16 3s:902ms** | 94,57 ± 0,37 0s:083ms | 94,19 ± 0,39 0s:083ms | 94,46 ± 0,38 0s:083ms | 94,79 ± 0,36 0s:075ms | 94,40 ± 0,39 0s:135ms |
| 3. TrainGDA | **96,19 ± 0,24 3s:682ms** | 94,03 ± 0,36 1s:911ms | 94,02 ± 0,38 2s:595ms | 93,95 ± 0,34 2s:050ms | 94,56 ± 0,34 1s:221ms | 93,68 ± 0,39 2s:716ms |
| 4. TrainGDX | **97,14 ± 0,28 2s:856ms** | 93,81 ± 0,37 0s:973ms | 93,79 ± 0,39 1s:910ms | 93,94 ± 0,36 0s:889ms | 94,62 ± 0,34 0s:621ms | 93,86 ± 0,37 1s:881ms |

Fonte: Produção do Autor (2017).

A Tabela 2 apresenta a média de Acurácia e de Tempo de Treinamento de todos os experimentos de reconhecimento de padrões para classificar dados. Esses resultados são de todas as RNPM com todos os "dataset's" e todos os "Back-Propagation" com os respectivos algoritmos de inicialização de pesos e bias.

Na Tabela 2 são comparados os resultados por linha entre os algoritmos de inicialização de pesos e bias. Os melhores resultados estão em negrito. E o melhor resultado entre todos (maior Acurácia com o Tempo de Treinamento) é sombreado em cinza.

A HeCI empata em 100% de acurácia com os outros algoritmos de inicialização de pesos e bias para o "dataset Acute Inflammations conforme a Tabela 2.

Entretanto, para os "dataset's" "Banknote Authentication", "Breast Cancer Wisconsin Diagnostic", "Iris", "Parkinson's" e "Seeds" a HeCI alcançou a melhor acurácia.

Filtrando apenas as melhores médias de resultados de todos os algoritmos, pode-se observar a precisão de Treinamento e Acurácia da HeCI:

Tabela 3: Melhores Resultados de Precisão de Treinamento e Acurácia de RNPM

| "DATASET'S" | Algoritmos de Inicialização de Pesos e Bias para o "Back-Propagation" Treinar RNPM | | | |
|---|---|---|---|---|
| | TREINAMENTO (%) | | ACURÁCIA (%) | |
| | **Proposta HeCI** | Outros Algoritmos | **Proposta HeCI** | Outros Algoritmos |
| 1. Acute Inflammation | 100,00 ± 0,00 | 100,00 ± 0,00 | 100,00 ± 0,00 | 100,00 ± 0,00 |
| 2. Banknote Autentication | 100,00 ± 0,00 | 100,00 ± 0,00 | 100,00 ± 0,00 | 99,99 ± 0,00 |
| 3. Breast Cancer Wisconsin Diagnostic | 99,92 ± 0,01 | 99,20 ± 0,01 | 99,11 ± 0,05 | 97,49 ± 0,13 |
| 4. Iris | 100,00 ± 0,00 | 99,48 ± 0,02 | 99,33 ± 0,12 | 97,51 ± 0,26 |
| 5. Parkinson's | 100,00 ± 0,00 | 99,99 ± 0,00 | 97,37 ± 0,37 | 84,70 ± 0,59 |
| 6. Seeds | 100,00 ± 0,00 | 100,00 ± 0,00 | 99,05 ± 0,16 | 94,79 ± 0,36 |

Fonte: Produção do Autor (2017).

A Tabela 3 apresenta resultados da HeCI que empata em 100% ou alcança melhor acurácia, quando comparada com os outros algoritmos abordados nesta pesquisa, Subseção 5.1.3.

Diante dos resultados das Tabelas 2 e 3 procedeu-se a avaliação dos resultados por testes de hipóteses.

## 5.4 Aplicação de Testes de Hipóteses aos Resultados

Explica Walpole et al. (2011) que a declaração formal de hipóteses estatísticas, sobre dados amostrais, fornece uma prova de aceitação ou rejeição de evidência experimental para a tomada de decisão, ou seja, que o algoritmo é significativamente melhor, pior ou equivalente a todos os resultados com base em testes de hipóteses.

São aplicados o Teste de "Friedman" com o pós Teste de "Tukey Post-Hoc HSD" e Teste de "Wilcoxon-M-W" descritos em Hammer (2013), Beale et al. (2016) e Nahm (2016). Esses

mesmos testes foram usados por Demšar (2006) (na área de Aprendizado de Máquina), Hazra e Gogtay (2016) (na área da Bioestatística) e Saremi et al. (2017) (na área de Otimização).

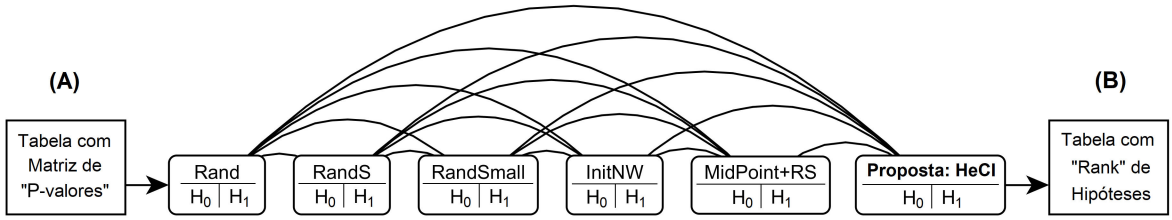
Ensina Walpole et al. (2011) que o resultado do teste de hipótese é o "P-Valor" (Valor Probabilístico) que atesta significativamente melhoria, piora ou equivalência de algoritmo aplicado aos dados amostrais. E segundo Hammer (2013), Beale et al. (2016) e Nahm (2016), entende-se:

I. O Teste de "Friedman" verifica a existência de diferença significativa entre resultados de diferentes algoritmos, mas não aponta o algoritmo.
II. O Teste de "Tukey HSD Post-hoc" calcula a diferença significativa de maneira exata de média entre todos os algoritmos tomados dois a dois, apontando qual ou quais algoritmo provocou a diferença.
III. O Teste de "Wincoxon-M-W" calcula um "Rank" de valores significativos obtidos pela comparação dois a dois de todos os algoritmos.

Aplica-se esses testes, medindo a diferença significativa sobre os resultados pelo cálculo do "P-Valor": (I) do Teste de "Friedman" para verificar a diferença entre os algoritmos; (II) do Teste de "Tukey Post-Hoc HSD" para apontar o algoritmo ou algoritmos com diferença; e (III) do Teste de "Wincoxon-M-W" para calcular um "Rank" dos melhores resultados.

O processo de aplicação dos testes de hipóteses é ilustrado na Figura 32:

Figura 32: Descrição da Aplicação dos Testes de Hipóteses e Comparação de Resultados




Fonte: Produção do Autor (2017).

A Figura 32 descreve a comparação entre todos os resultados da Tabela 2 - Acurácia e Tempo de Treinamento de "Back-Propagation" com todos os algoritmos de inicializar pesos e bias. Em Figura 32-A, a matriz de "P-Valores" do Teste de "Tukey Post-Hoc HSD" e Teste de "Wilcoxon-M-W" é gerada. Cada linha da matriz de "P-Valores" contém dois valores respectivamente ao Teste de "Tukey Post-Hoc HSD" e ao Teste de "Wilcoxon-M-W". Todos os resultados são testados e comparados sobre as hipóteses $\mathcal{H}_0$ e $\mathcal{H}_1$. Em Figura 32-B, é apresentada a posição em "Rank" de melhor algoritmo para inicializar pesos e bias.

Os parâmetros usados para os testes de hipóteses são:

1. Magnitude $\alpha = [0 \leq \text{P-Valor} \leq 1]$.
2. Significância $\alpha \leq 0,01$.
3. Região Crítica: Bilateral.
4. Confiança $1 - \alpha \geq 0,99$ ou 99%, Então $\mathcal{H}_1$ Senão $\mathcal{H}_0$.

Segue-se então a aplicação dos testes de hipóteses aos resultados dos experimentos.

### 5.4.1 Testes de Hipóteses nos Resultados com "Acute Inflammations"

Os testes de hipóteses avaliam a Acurácia com o Tempo de Treinamento de RNPM:

Tabela 4: Resultado dos Experimentos: "dataset Acute Inflammations"

| Média de Acurácia e Tempo de Treinamento: "Back-Propagation" + Algoritmos de Inicialização de Pesos e Bias | | | | | | |
|---|---|---|---|---|---|---|
| "Back-Propagation" | **1.Proposta: HeCI** | 2.RandS | 3.InitNW | 4.Rand | 5.RandSmall | 6.Midpoint+RS |
| 1. TrainRP | 100,00 ± 0,00<br>0s:061ms | 100,00 ± 0,00<br>0s:048ms | 99,14 ± 0,23<br>0s:080ms | 99,78 ± 0,13<br>0s:075ms | 100,00 ± 0,00<br>0s:048ms | 99,33 ± 0,21<br>0s:078ms |
| 2. TrainBR | 100,00 ± 0,00<br>0s:061ms | 100,00 ± 0,00<br>0s:053ms | 100,00 ± 0,00<br>0s:058ms | 100,00 ± 0,00<br>0s:053ms | 100,00 ± 0,00<br>0s:062ms | 100,00 ± 0,00<br>0s:058ms |
| 3. TrainGDA | 100,00 ± 0,00<br>0s:187ms | 100,00 ± 0,00<br>0s:191ms | 99,64 ± 0,15<br>0s:272ms | 100,00 ± 0,00<br>0s:182ms | 100,00 ± 0,00<br>0s:179ms | 99,50 ± 0,18<br>0s:282ms |
| 4. TrainGDX | 100,00 ± 0,00<br>0s:189ms | 100,00 ± 0,00<br>0s:176ms | 99,39 ± 0,20<br>0s:293ms | 100,00 ± 0,00<br>0s:185ms | 100,00 ± 0,00<br>0s:174ms | 99,64 ± 0,16<br>0s:286ms |

Fonte: Produção do Autor (2017).

A Tabela 4 avaliada pelo Teste de "Friedman", apresenta diferença significativa "P-Valor = 5,67E-169". Segue-se com o Teste de "Tukey HSD Post-hoc" e o Teste de "Wilcoxon-M-W":

Tabela 5: Matriz de "P-Valores" de Resultados com o "dataset Acute Inflammations"

| "A" Teste Tukey HSD Post-hoc e "B" Teste Wilcoxon-M-W: ambos os testes com Magnitude de $\alpha = [0 \leq \text{P-Valor} \leq 1]$<br>Região Crítica Bilateral: $\alpha \leq 0,01$ e Nível de Confiança 1 $\alpha \geq 0.99$ ou 99%, Então $\mathcal{H}_1$ Senão $\mathcal{H}_0$ | | | | | | |
|---|---|---|---|---|---|---|
| **P-Valor** | | Rand | RandS | RandSmall | InitNW | Midpoint+RS | **Proposta: HeCI** |
| Rand | Tukey | | 0,0001 | 0,0879 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | | 0,3001 | 0,1721 | 0,0062 | 0,0039 | 0,0374 |
| RandS | Tukey | 0,0001 | | 0,4346 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,3001 | | 0,8163 | 0,0003 | 0,0002 | 0,0022 |
| RandSmall | Tukey | 0,0879 | 0,4346 | | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,1721 | 0,8163 | | 0,0011 | 0,0006 | 0,0045 |
| InitNW | Tukey | 0,0000 | 0,0000 | 0,0000 | | 0,9611 | 0,5377 |
| | Wilcoxon | 0,0062 | 0,0003 | 0,0011 | | 0,8861 | 0,2395 |
| Midpoint+RS | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,9611 | | 0,1166 |
| | Wilcoxon | 0,0039 | 0,0002 | 0,0006 | 0,8861 | | 0,1666 |
| **Proposta: HeCI** | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,5377 | 0,1166 | |
| | Wilcoxon | 0,0374 | 0,0022 | 0,0045 | 0,2395 | 0,1666 | |

Fonte: Produção do Autor (2017).

Tabela 6: Dedução Lógica Inferencial com Base na Tabela 5 Matriz de "P-Valores" "dataset Acute Inflammations" (Inflamação Aguda)

| Algoritmos de Inicialização de Pesos e Bias | Acurácia Média de 04 Algoritmos de "Back-Propagation" | "Rank" | "A" Teste Tukey HSD Post-hoc | "B" Teste Wilcoxon-M-W | Dedução Lógica Inferencial Se $(A \wedge B) \rightarrow \mathcal{H}_1$ Senão $\mathcal{H}_0$ |
|---|---|---|---|---|---|
| **1. Proposta: HeCI** | **100,00 ± 0,00** | **1º** | $\boldsymbol{\mathcal{H}_0}$ | $\boldsymbol{\mathcal{H}_0}$ | $\mathcal{H}_0 \therefore$ **Equivalente** |
| 2. RandS | 100,00 ± 0,00 | 1º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |
| 3. RandSmall | 100,00 ± 0,00 | 1º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |
| 4. Rand | 99,94 ± 0,03 | 2º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |
| 5. InitNW | 99,62 ± 0,08 | 3º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |
| 6. Midpoint+RS | 99,54 ± 0,09 | 4º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |

Fonte: Produção do Autor (2017).

As Tabelas 5 e 6 apresentam os resultados dos testes de hipóteses.

### 5.4.2 Testes de Hipóteses nos Resultados com “Banknote Authentication”

Os testes de hipóteses avaliam a Acurácia com o Tempo de Treinamento de RNPM:

Tabela 7: Resultado dos Experimentos: "dataset Banknote Authentication"

| Média de Acurácia e Tempo de Treinamento: "Back-Propagation" + Algoritmos de Inicialização de Pesos e Bias | | | | | | |
|---|---|---|---|---|---|---|
| "Back-Propagation" | **1.Proposta: HeCI** | 2.RandS | 3.InitNW | 4.Rand | 5.RandSmall | 6.Midpoint+RS |
| 1. TrainRP | **99,93 ± 0,01**<br>**0s:440ms** | 99,85 ± 0,03<br>0s:084ms | 99,86 ± 0,03<br>0s:140ms | 99,81 ± 0,03<br>0s:085ms | 99,63 ± 0,06<br>0s:177ms | 99,81 ± 0,04<br>0s:131ms |
| 2. TrainBR | **100,00 ± 0,00**<br>**0s:080ms** | 99,99 ± 0,01<br>0s:061ms | 99,95 ± 0,01<br>0s:068ms | 99,98 ± 0,01<br>0s:064ms | 99,97 ± 0,01<br>0s:079ms | 99,97 ± 0,01<br>0s:067ms |
| 3. TrainGDA | **99,85 ± 0,02**<br>**0s:733ms** | 99,97 ±0,01<br>0s:593ms | 99,89 ± 0,02<br>0s:836ms | 99,97 ± 0,01<br>0s:625ms | 99,99 ± 0,00<br>1s:033ms | 99,85 ± 0,03<br>0s:777ms |
| 4. TrainGDX | **100,00 ± 0,00**<br>**0s:334ms** | 99,94 ± 0,01<br>0s:277ms | 99,92 ± 0,02<br>0s:458ms | 99,95 ± 0,01<br>0s:296ms | 99,98 ± 0,01<br>0s:447ms | 99,90 ± 0,02<br>0s:422ms |

Fonte: Produção do Autor (2017).

A Tabela 7 avaliada pelo Teste de "Friedman", apresenta diferença significativa "P-Valor = 0,00E-000". Segue-se com o Teste de "Tukey HSD Post-hoc" e o Teste de "Wilcoxon-M-W":

Tabela 8: Matriz de "P-Valores" de Resultados com o "dataset Banknote Authentication"

| "A" Teste Tukey HSD Post-hoc e "B" Teste Wilcoxon-M-W: ambos os testes com Magnitude de $\alpha$ = [0 ≤ P-Valor ≤ 1]<br>Região Crítica Bilateral: $\alpha \leq 0,01$ e Nível de Confiança 1 $\alpha \geq 0.99$ ou 99%, Então $\mathcal{H}_1$ Senão $\mathcal{H}_0$ | | | | | | |
|---|---|---|---|---|---|---|
| **P-Valor** | | Rand | RandS | RandSmall | InitNW | Midpoint+RS | **Proposta: HeCI** |
| Rand | Tukey | | 0,0018 | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | | 0,4869 | 0,0000 | 0,3472 | 0,6315 | 0,0031 |
| RandS | Tukey | 0,0018 | | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,4869 | | 0,0000 | 0,1263 | 0,2957 | 0,0005 |
| RandSmall | Tukey | 0,0000 | 0,0000 | | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,0000 | 0,0000 | | 0,0002 | 0,0000 | 0,0660 |
| InitNW | Tukey | 0,0000 | 0,0000 | 0,0000 | | 0,4531 | 0,0000 |
| | Wilcoxon | 0,3472 | 0,1263 | 0,0002 | | 0,6164 | 0,0476 |
| Midpoint+RS | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,4531 | | 0,0000 |
| | Wilcoxon | 0,6315 | 0,2957 | 0,0000 | 0,6164 | | 0,0114 |
| **Proposta: HeCI** | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |
| | Wilcoxon | 0,0031 | 0,0005 | 0,0660 | 0,0476 | 0,0114 | |

Fonte: Produção do Autor (2017).

Tabela 9: Dedução Lógica Inferencial com Base na Tabela 8 Matriz de "P-Valores" "dataset Banknote Authentication" (Autenticação Bancária)

| Algoritmos de Inicialização de Pesos e Bias | Acurácia Média de 04 Algoritmos de "Back-Propagation" | "Rank" | "A" Teste Tukey HSD Post-hoc | "B" Teste Wilcoxon-M-W | Dedução Lógica Inferencial Se $(A \wedge B) \rightarrow \mathcal{H}_1$ Senão $\mathcal{H}_0$ |
|---|---|---|---|---|---|
| **1. Proposta: HeCI** | **99,95 ± 0,01** | **1º** | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_0}$ | $\mathcal{H}_0 \therefore$ **Equivalente** |
| 2. RandS | 99,94 ± 0,01 | 2º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |
| 3. Rand | 99,93 ± 0,01 | 3º | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |
| 4. InitNW | 99,90 ± 0,01 | 5º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |
| 5. RandSmall | 99,90 ± 0,02 | 4º | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |
| 6. Midpoint+RS | 99,89 ± 0,01 | 6º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0 \therefore$ Equivalente |

Fonte: Produção do Autor (2017).

As Tabelas 8 e 9 apresentam os resultados dos testes de hipóteses.

### 5.4.3 Testes de Hipóteses nos Resultados com “Breast Cancer W. Diagnostic”

Os testes de hipóteses avaliam a Acurácia com o Tempo de Treinamento de RNPM:

Tabela 10: Resultado dos Experimentos: "dataset Breast Cancer W. Diagnostic"

| Média de Acurácia e Tempo de Treinamento: "Back-Propagation" + Algoritmos de Inicialização de Pesos e Bias | | | | | | |
|---|---|---|---|---|---|---|
| "Back-Propagation" | **1.Proposta: HeCI** | 2.RandS | 3.InitNW | 4.Rand | 5.RandSmall | 6.Midpoint+RS |
| 1. TrainRP | **98,39 ± 0,09**<br>**5s:996ms** | 96,59 ± 0,14<br>0s:196ms | 96,75 ± 0,14<br>0s:218ms | 96,70 ± 0,14<br>0s:228ms | 97,49 ± 0,13<br>0s:076ms | 96,51 ± 0,15<br>0s:209ms |
| 2. TrainBR | **99,11 ± 0,05**<br>**34s:027ms** | 96,45 ± 0,16<br>0s:379ms | 96,58 ± 0,15<br>0s:319ms | 96,58± 0,15<br>0s:277ms | 96,55 ± 0,16<br>0s:750ms | 96,41 ± 0,16<br>0s:268ms |
| 3. TrainGDA | **98,04 ± 0,09**<br>**6s:130ms** | 96,50 ± 0,16<br>2s:536ms | 96,45 ± 0,16<br>3s:273ms | 96,63 ± 0,14<br>1s:798ms | 96,61 ± 0,11<br>6s:397ms | 96,57 ± 0,16<br>3s:466ms |
| 4. TrainGDX | **98,39 ± 0,10**<br>**5s:419ms** | 96,47 ± 0,16<br>0s:500ms | 96,53 ± 0,17<br>0s:691ms | 96,48 ± 0,15<br>0s:595ms | 96,06 ± 0,16<br>1s:184ms | 96,61± 0,16<br>0s:714ms |

Fonte: Produção do Autor (2017).

A Tabela 10 avaliada pelo Teste de "Friedman", apresenta diferença significativa "P-Valor = 0,00E-000". Segue-se com o Teste de "Tukey HSD Post-hoc" e o Teste de "Wilcoxon-M-W":

Tabela 11: Matriz de "P-Valores" de Resultados com o "dataset Breast Cancer W. Diagnostic"

| "A" Teste Tukey HSD Post-hoc e "B" Teste Wilcoxon-M-W: ambos os testes com Magnitude de $\alpha = [0 \leq \text{P-Valor} \leq 1]$<br>Região Crítica Bilateral: $\alpha \leq 0,01$ e Nível de Confiança $1-\alpha \geq 0.99$ ou 99%, Então $\mathcal{H}_1$ Senão $\mathcal{H}_0$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| **P-Valor** | | Rand | RandS | RandSmall | InitNW | Midpoint+RS | **Proposta: HeCI** |
| Rand | Tukey | | 0,7866 | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | | 0,3220 | 0,3880 | 0,2799 | 0,4069 | 0,0000 |
| RandS | Tukey | 0,7866 | | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,3220 | | 0,2524 | 0,0463 | 0,0857 | 0,0000 |
| RandSmall | Tukey | 0,0000 | 0,0000 | | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,3880 | 0,2524 | | 0,9008 | 0,9688 | 0,0000 |
| InitNW | Tukey | 0,0000 | 0,0000 | 0,0000 | | 0,9985 | 0,0000 |
| | Wilcoxon | 0,2799 | 0,0463 | 0,9008 | | 0,8061 | 0,0000 |
| Midpoint+RS | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,9985 | | 0,0000 |
| | Wilcoxon | 0,4069 | 0,0857 | 0,9688 | 0,8061 | | 0,0000 |
| **Proposta: HeCI** | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |
| | Wilcoxon | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |

Fonte: Produção do Autor (2017).

Tabela 12: Dedução Lógica Inferencial com Base na Tabela 11 Matriz de "P-Valores" "dataset Breast Cancer W. Diagnostic" (Diagnóstico de Câncer de Mama)

| Algoritmos de Inicialização de Pesos e Bias | Acurácia Média de 04 Algoritmos de "Back-Propagation" | "Rank" | "A" Teste Tukey HSD Post-hoc | "B" Teste Wilcoxon-M-W | Dedução Lógica Inferencial Se $(A \wedge B) \rightarrow \mathcal{H}_1$ Senão $\mathcal{H}_0$ |
|---|---|---|---|---|---|
| **1. Proposta: HeCI** | **98,48 ± 0,04** | **1º** | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ $\therefore$ **Melhoria Significativa** |
| 2. RandSmall | 96,62 ± 0,07 | 2º | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 3. Rand | 96,60 ± 0,07 | 3º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 4. InitNW | 96,58 ± 0,08 | 4º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 5. Midpoint+RS | 96,53 ± 0,08 | 5º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 6. RandS | 96,50 ± 0,08 | 6º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |

Fonte: Produção do Autor (2017).

As Tabelas 11 e 12 apresentam os resultados dos testes de hipóteses.

### 5.4.4 Testes de Hipóteses nos Resultados com “Iris”

Os testes de hipóteses avaliam a Acurácia com o Tempo de Treinamento de RNPM:

Tabela 13: Resultado dos Experimentos: "dataset Iris"

| Média de Acurácia e Tempo de Treinamento: "Back-Propagation" + Algoritmos de Inicialização de Pesos e Bias | | | | | | |
|---|---|---|---|---|---|---|
| "Back-Propagation" | **1.Proposta: HeCI** | 2.RandS | 3.InitNW | 4.Rand | 5.RandSmall | 6.Midpoint+RS |
| 1. TrainRP | **98,00 ± 0,25**<br>**1s:711ms** | 95,16 ± 0,32<br>1s:563ms | 95,31 ± 0,34<br>1s:562ms | 95,24 ± 0,33<br>1s:794ms | 94,78 ± 0,33<br>2s:311ms | 95,18 ± 0,32<br>1s:698ms |
| 2. TrainBR | **99,33 ± 0,12**<br>**2s:927ms** | 95,78 ± 0,28<br>0s:185ms | 95,98 ± 0,27<br>0s:180ms | 95,69 ± 0,28<br>0s:147ms | 33,82 ± 0,19<br>0s:042ms | 95,71 ± 0,30<br>0s:168ms |
| 3. TrainGDA | **97,33 ± 0,31**<br>**1s:601ms** | 96,78 ± 0,27<br>3s:522ms | 96,47 ± 0,27<br>3s:536ms | 96,82 ± 0,27<br>3s:787ms | 97,51 ± 0,26<br>4s:893ms | 96,40 ± 0,28<br>3s:743ms |
| 4.TrainGDX | **98,00 ± 0,25**<br>**1s:645ms** | 96,78 ± 0,27<br>2s:535ms | 96,51 ± 0,28<br>2s:895ms | 96,58 ± 0,28<br>2s:835ms | 96,49 ± 0,27<br>3s:517ms | 96,44 ± 0,27<br>3s:020ms |

Fonte: Produção do Autor (2017).

A Tabela 13 avaliada pelo Teste de "Friedman", apresenta diferença significativa "P-Valor = 2,25E-48". Segue-se com o Teste de "Tukey HSD Post-hoc" e o Teste de "Wilcoxon-M-W":

Tabela 14: Matriz de "P-Valores" de Resultados com o "dataset Iris"

| "A" Teste Tukey HSD Post-hoc e "B" Teste Wilcoxon-M-W: ambos os testes com Magnitude de $\alpha$ = [0 ≤ P-Valor ≤ 1]<br>Região Crítica Bilateral: $\alpha \leq 0,01$ e Nível de Confiança $1-\alpha \geq 0.99$ ou 99%, Então $\mathcal{H}_1$ Senão $\mathcal{H}_0$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| **P-Valor** | | Rand | RandS | RandSmall | InitNW | Midpoint+RS | **Proposta: HeCI** |
| Rand | Tukey | | 1,0000 | 0,0000 | 0,1333 | 0,0320 | 0,0000 |
| | Wilcoxon | | 0,8231 | 0,0017 | 0,9300 | 0,9002 | 0,0010 |
| RandS | Tukey | 1,0000 | | 0,0000 | 0,1877 | 0,0501 | 0,0000 |
| | Wilcoxon | 0,8231 | | 0,0033 | 0,8002 | 0,7697 | 0,0004 |
| RandSmall | Tukey | 0,0000 | 0,0000 | | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,0017 | 0,0033 | | 0,0008 | 0,0008 | 0,0000 |
| InitNW | Tukey | 0,1333 | 0,1877 | 0,0000 | | 0,9948 | 0,0000 |
| | Wilcoxon | 0,9300 | 0,8002 | 0,0008 | | 0,9899 | 0,0003 |
| Midpoint+RS | Tukey | 0,0320 | 0,0501 | 0,0000 | 0,9948 | | 0,0000 |
| | Wilcoxon | 0,9002 | 0,7697 | 0,0008 | 0,9899 | | 0,0005 |
| **Proposta: HeCI** | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |
| | Wilcoxon | 0,0010 | 0,0004 | 0,0000 | 0,0003 | 0,0005 | |

Fonte: Produção do Autor (2017).

Tabela 15: Dedução Lógica Inferencial com Base na Tabela 14 Matriz de "P-Valores" "dataset Iris" (Plantas)

| Algoritmos de Inicialização de Pesos e Bias | Acurácia Média de 04 Algoritmos de "Back-Propagation" | "Rank" | "A" Teste Tukey HSD Post-hoc | "B" Teste Wilcoxon-M-W | Dedução Lógica Inferencial Se $(A \wedge B) \rightarrow \mathcal{H}_1$ Senão $\mathcal{H}_0$ |
|---|---|---|---|---|---|
| **1. Proposta: HeCI** | **98,17 ± 0,12** | **1º** | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ ∴ **Melhoria Significativa** |
| 2. RandS | 96,12 ± 0,14 | 2º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ ∴ Equivalente |
| 3. Rand | 96,08 ± 0,15 | 3º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ ∴ Equivalente |
| 4. InitNW | 96,07 ± 0,15 | 4º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ ∴ Equivalente |
| 5. Midpoint+RS | 95,93 ± 0,15 | 5º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ ∴ Equivalente |
| 6. RandSmall | 80,65 ± 0,79 | 6º | $\mathcal{H}_1$ | $\mathcal{H}_1$ | $\mathcal{H}_1$ ∴ Queda Significativa |

Fonte: Produção do Autor (2017).

As Tabelas 14 e 15 apresentam os resultados dos testes de hipóteses.

### 5.4.5 Testes de Hipóteses nos Resultados com “Parkinson's”

Os testes de hipóteses avaliam a Acurácia com o Tempo de Treinamento de RNPM:

Tabela 16: Resultado dos Experimentos: "dataset Parkinson's"

| Média de Acurácia e Tempo de Treinamento: "Back-Propagation" + Algoritmos de Inicialização de Pesos e Bias | | | | | | |
|---|---|---|---|---|---|---|
| "Back-Propagation" | **1.Proposta: HeCI** | 2.RandS | 3.InitNW | 4.Rand | 5.RandSmall | 6.Midpoint+RS |
| 1. TrainRP | **93,16 ± 0,33**<br>**6s:623ms** | 83,58 ± 0,60<br>0s:082ms | 83,68 ± 0,65<br>0s:083ms | 83,79 ± 0,65<br>0s:090ms | 84,61 ± 0,59<br>0s:103ms | 83,03 ± 0,62<br>0s:087ms |
| 2. TrainBR | **97,37 ± 0,37**<br>**29s:390ms** | 82,96 ± 0,59<br>0s:188ms | 83,09 ± 0,65<br>0s:179ms | 83,39 ± 0,66<br>0s:215ms | 82,98 ± 0,65<br>0s:252ms | 82,93 ± 0,60<br>0s:244ms |
| 3. TrainGDA | **88,42 ± 0,38**<br>**5s:863ms** | 83,72 ± 0,59<br>0s:953ms | 83,46 ± 0,58<br>0s:888ms | 84,35 ± 0,61<br>1s:270ms | 83,89 ± 0,63<br>0s:865ms | 84,70 ± 0,59<br>0s:835ms |
| 4. TrainGDX | **88,42 ± 0,49**<br>**5s:865ms** | 83,37 ± 0,61<br>0s:581ms | 83,35 ± 0,62<br>0s:533ms | 83,82 ± 0,63<br>0s:730ms | 83,89 ± 0,64<br>0s:582ms | 84,47 ± 0,61<br>0s:438ms |

Fonte: Produção do Autor (2017).

A Tabela 16 avaliada pelo Teste de "Friedman", apresenta diferença significativa "P-Valor = 0,00E-000". Segue-se com o Teste de "Tukey HSD Post-hoc" e o Teste de "Wilcoxon-M-W":

Tabela 17: Matriz de "P-Valores" de Resultados com o "dataset Parkinson's"

| "A" Teste Tukey HSD Post-hoc e "B" Teste Wilcoxon-M-W: ambos os testes com Magnitude de $\alpha = [0 \leq \text{P-Valor} \leq 1]$<br>Região Crítica Bilateral: $\alpha \leq 0,01$ e Nível de Confiança $1 - \alpha \geq 0.99$ ou 99%, Então $\mathcal{H}_1$ Senão $\mathcal{H}_0$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| **P-Valor** | | Rand | RandS | RandSmall | InitNW | Midpoint+RS | **Proposta: HeCI** |
| Rand | Tukey | | 0,0139 | 0,0000 | 0,0009 | 0,1187 | 0,0000 |
| | Wilcoxon | | 0,1899 | 0,0756 | 0,2512 | 0,5540 | 0,0000 |
| RandS | Tukey | 0,0139 | | 0,0000 | 0,9776 | 0,9759 | 0,0000 |
| | Wilcoxon | 0,1899 | | 0,0011 | 0,9019 | 0,4524 | 0,0000 |
| RandSmall | Tukey | 0,0000 | 0,0000 | | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,0756 | 0,0011 | | 0,0014 | 0,0110 | 0,0000 |
| InitNW | Tukey | 0,0009 | 0,9776 | 0,0000 | | 0,6752 | 0,0000 |
| | Wilcoxon | 0,2512 | 0,9019 | 0,0014 | | 0,5420 | 0,0000 |
| Midpoint+RS | Tukey | 0,1187 | 0,9759 | 0,0000 | 0,6752 | | 0,0000 |
| | Wilcoxon | 0,5540 | 0,4524 | 0,0110 | 0,5420 | | 0,0000 |
| **Proposta: HeCI** | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |
| | Wilcoxon | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |

Fonte: Produção do Autor (2017).

Tabela 18: Dedução Lógica Inferencial com Base na Tabela 17 Matriz de "P-Valores" "dataset Parkinson's" (Doença de Parkinson's)

| Algoritmos de Inicialização de Pesos e Bias | Acurácia Média de 04 Algoritmos de "Back-Propagation" | "Rank" | "A" Teste Tukey HSD Post-hoc | "B" Teste Wilcoxon-M-W | Dedução Lógica Inferencial Se $(A \wedge B) \rightarrow \mathcal{H}_1$ Senão $\mathcal{H}_0$ |
|---|---|---|---|---|---|
| **1. Proposta: HeCI** | **91,84 ± 0,23** | **1º** | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ $\therefore$ **Melhoria Significativa** |
| 2. RandSmall | 83,85 ± 0,32 | 2º | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 3. Rand | 83,84 ± 0,32 | 3º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 4. Midpoint+RS | 83,79 ± 0,30 | 4º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 5. RandS | 83,41 ± 0,30 | 5º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 6. InitNW | 83,39 ± 0,31 | 6º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |

Fonte: Produção do Autor (2017).

As Tabelas 17 e 18 presentam os resultados dos testes de hipóteses.

### 5.4.6 Testes de Hipóteses nos Resultados com “Seeds”

Os testes de hipóteses avaliam a Acurácia com o Tempo de Treinamento de RNPM:

Tabela 19: Resultado dos Experimentos: "dataset Seeds"

| Média de Acurácia e Tempo de Treinamento: "Back-Propagation" + Algoritmos de Inicialização de Pesos e Bias | | | | | | |
|---|---|---|---|---|---|---|
| "Back-Propagation" | **1.Proposta: HeCI** | 2.RandS | 3.InitNW | 4.Rand | 5.RandSmall | 6.Midpoint+RS |
| 1. TrainRP | **95,24 ± 0,43**<br>**2s:628ms** | 92,52 ± 0,46<br>2s:306ms | 92,67 ± 0,48<br>2s:886ms | 92,25 ± 0,48<br>2s:350ms | 93,11 ± 0,43<br>2s:106ms | 92,06 ± 0,54<br>2s:902ms |
| 1. TrainBR | **99,05 ± 0,16**<br>**3s:902ms** | 94,57 ± 0,37<br>0s:083ms | 94,19 ± 0,39<br>0s:083ms | 94,46 ± 0,38<br>0s:083ms | 94,79 ± 0,36<br>0s:075ms | 94,40 ± 0,39<br>0s:135ms |
| 2. TrainGDA | **96,19 ± 0,24**<br>**3s:682ms** | 94,03 ± 0,36<br>1s:911ms | 94,02 ± 0,38<br>2s:595ms | 93,95 ± 0,34<br>2s:050ms | 94,56 ± 0,34<br>1s:221ms | 93,68 ± 0,39<br>2s:716ms |
| 3. TrainGDX | **97,14 ± 0,28**<br>**2s:856ms** | 93,81 ± 0,37<br>0s:973ms | 93,79 ± 0,39<br>1s:910ms | 93,94 ± 0,36<br>0s:889ms | 94,62 ± 0,34<br>0s:621ms | 93,86 ± 0,37<br>1s:881ms |

Fonte: Produção do Autor (2017).

A Tabela 19 avaliada pelo Teste de "Friedman", apresenta diferença significativa "P-Valor = 6,20E-231". Segue-se com o Teste de "Tukey HSD Post-hoc" e o Teste de "Wilcoxon-M-W":

Tabela 20: Matriz de "P-Valores" de Resultados com o "dataset Seeds"

| "A" Teste Tukey HSD Post-hoc e "B" Teste Wilcoxon-M-W: ambos os testes com Magnitude de $\alpha = [0 \leq \text{P-Valor} \leq 1]$<br>Região Crítica Bilateral: $\alpha \leq 0,01$ e Nível de Confiança $1 - \alpha \geq 0.99$ ou 99%, Então $\mathcal{H}_1$ Senão $\mathcal{H}_0$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| **P-Valor** | | Rand | RandS | RandSmall | InitNW | Midpoint+RS | **Proposta: HeCI** |
| Rand | Tukey | | 0,2735 | 0,0913 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | | 0,7306 | 0,7729 | 0,0202 | 0,0445 | 0,0000 |
| RandS | Tukey | 0,2735 | | 0,9962 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,7306 | | 0,8709 | 0,0413 | 0,0902 | 0,0000 |
| RandSmall | Tukey | 0,0913 | 0,9962 | | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,7729 | 0,8709 | | 0,0228 | 0,0671 | 0,0000 |
| InitNW | Tukey | 0,0000 | 0,0000 | 0,0000 | | 0,0243 | 0,0000 |
| | Wilcoxon | 0,0202 | 0,0413 | 0,0228 | | 0,7466 | 0,0000 |
| Midpoint+RS | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,0243 | | 0,0000 |
| | Wilcoxon | 0,0445 | 0,0902 | 0,0671 | 0,7466 | | 0,0000 |
| **Proposta: HeCI** | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |
| | Wilcoxon | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |

Fonte: Produção do Autor (2017).

Tabela 21: Dedução Lógica Inferencial com Base na Tabela 20 Matriz de "P-Valores" "dataset Seeds" (Variedades de Trigo)

| Algoritmos de Inicialização de Pesos e Bias | Acurácia Média de 04 Algoritmos de "Back-Propagation" | "Rank" | "A" Teste Tukey HSD Post-hoc | "B" Teste Wilcoxon-M-W | Dedução Lógica Inferencial Se $(A \wedge B) \rightarrow \mathcal{H}_1$ Senão $\mathcal{H}_0$ |
|---|---|---|---|---|---|
| **1. Proposta: HeCI** | **96,90 ± 0,15** | **1º** | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ $\therefore$ **Melhoria Significativa** |
| 2. RandSmall | 94,27 ± 0,18 | 2º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 3. RandS | 93,73 ± 0,20 | 3º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 4. InitNW | 93,67 ± 0,21 | 4º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 5. Rand | 93,65 ± 0,20 | 5º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 6. Midpoint+RS | 93,50 ± 0,22 | 6º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |

Fonte: Produção do Autor (2017).

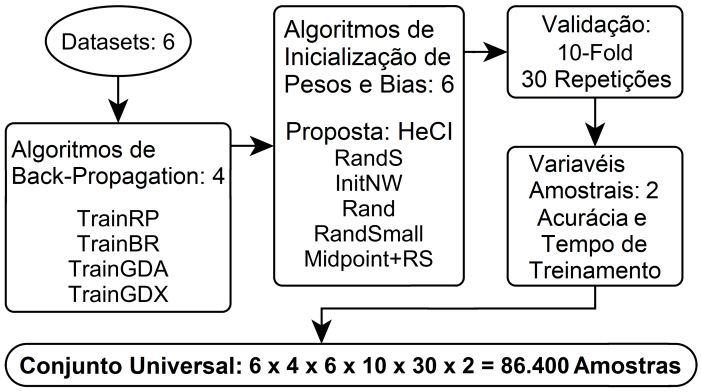
As Tabelas 20 e 21 apresentam os resultados dos testes de hipóteses.

### 5.4.7 Testes de Hipóteses no Conjunto Universal de Resultados

Agora, procede-se uma avaliação universal dos resultados experimentais, de maneira independente de "dataset's" e tipo de "Back-Propagation". O objetivo é verifica a melhoria universal sobre todas as amostras de aprendizado de RNPM. Para tanto, foi criado um conjunto universal com todos os resultados juntos, para uso da Regra Geral Seção 1.3.

Todos os resultados dos Experimentos Computacionais descritos na Tabela 2 foram medidos juntos. O procedimento de junção dos resultados é ilustrado na Figura 33:

Figura 33: Conjunto Universal de Amostras de Aprendizado de RNPM




Fonte: Produção do Autor (2017).

A Figura 33 descreve a junção de quantitativos de todos os experimentos da Tabela 2. As 86400 amostras de aprendizagem (Acurácia com o Tempo de Treinamento) de RNPM são avaliadas em um único conjunto de dados pelos testes de hipóteses

O Teste de "Friedman" apresenta "P-Valor = 0,0E-00" na avaliação dos resultados do conjunto universal de amostras de aprendizagem (Acurácia com o Tempo de Treinamento). Prossegue-se com os Testes de "Tukey HSD Post-hoc" e de "Wilcoxon-M-W":

Tabela 22: Matriz de "P-Valores": Resultado da Avaliação dos Testes de Hipóteses para o Conjunto Universal de Amostras de Aprendizado e RNPM

"A" Teste Tukey HSD Post-hoc e "B" Teste Wilcoxon-M-W: ambos os testes com Magnitude de $\alpha$ = [0 ≤ P-Valor ≤ 1]
Região Crítica Bilateral: $\alpha \leq 0,01$ e Nível de Confiança $1-\alpha \geq 0.99$ ou 99%, Então $\mathcal{H}_1$ Senão $\mathcal{H}_0$

| **P-Valor** | | Rand | RandS | RandSmall | InitNW | Midpoint+RS | **Proposta: HeCI** |
|---|---|---|---|---|---|---|---|
| Rand | Tukey | | 0,0008 | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | | 0,4204 | 0,8737 | 0,0638 | 0,1268 | 0,0000 |
| RandS | Tukey | 0,0008 | | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,4204 | | 0,5762 | 0,0085 | 0,0212 | 0,0000 |
| RandSmall | Tukey | 0,0000 | 0,0000 | | 0,0008 | 0,0000 | 0,0000 |
| | Wilcoxon | 0,8737 | 0,5762 | | 0,0750 | 0,1342 | 0,0000 |
| InitNW | Tukey | 0,0000 | 0,0000 | 0,0008 | | 0,9368 | 0,0000 |
| | Wilcoxon | 0,0638 | 0,0085 | 0,0750 | | 0,7348 | 0,0000 |
| Midpoint+RS | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,9368 | | 0,0000 |
| | Wilcoxon | 0,1268 | 0,0212 | 0,1342 | 0,7348 | | 0,0000 |
| **Proposta: HeCI** | Tukey | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |
| | Wilcoxon | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | |

Fonte: Produção do Autor (2017).

Tabela 23: Dedução Lógica Inferencial com Base na Tabela 22 Matriz de "P-Valores" Conjunto Universal de Amostras de Aprendizado de RNPM

| Algoritmos de Inicialização de Pesos e Bias | Acurácia Média de 04 Algoritmos de "Back-Propagation" | "Rank" | "A" Teste Tukey HSD Post-hoc | "B" Teste Wilcoxon-M-W | Dedução Lógica Inferencial Se $(A \wedge B) \rightarrow \mathcal{H}_1$ Senão $\mathcal{H}_0$ |
|---|---|---|---|---|---|
| **1. Proposta: HeCI** | **97,57 ± 0,06** | **1º** | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ | $\boldsymbol{\mathcal{H}_1}$ $\therefore$ **Melhoria Significativa** |
| 2. Rand | 95,01 ± 0,09 | 2º | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 3. RandS | 94,95 ± 0,09 | 3º | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 4. Midpoint+RS | 94,87 ± 0,09 | 4º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 5. InitNW | 94,86 ± 0,10 | 5º | $\mathcal{H}_0$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |
| 6. RandSmall | 92,55 ± 0,17 | 6º | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ $\therefore$ Equivalente |

Fonte: Produção do Autor (2017).

A Tabela 22 apresenta os menores "P-Valores" para a HeCI. Na Tabela 23, a $\mathcal{H}_1$ é aceita pelos testes de hipóteses para a HeCI, atestando significativa diferença de melhoria. Os outros algoritmos não passaram em ambos os testes, sendo então rejeitados, $\mathcal{H}_0$.

## 5.5 Análise de Resultados

Os experimentos de Reconhecimento de Padrões para Classificar Dados foram realizados em mesmo ambiente computacional. Os resultados demonstram claramente o alcance de 100% de acurácia ou uma diferença matemática de melhoria com o uso da HeCI.

O fato de 100% ou melhor acurácia na classificação de dados, para os "dataset's", é avaliado com testes de hipóteses nas Tabelas 6, 9, 12, 15, 18 e 21. E também é avaliado o conjunto universal de resultados da Tabela 23.

Por diferença matemática para mais acurácia ou pela inferência estatística, a HeCI alcançou melhoria significativa de acurácia conforme apresentado na Tabela 23. A Tabela 24 apresenta a dedução lógica pela Estatística Inferencial, para uso da Regra Geral Seção 1.3.

Tabela 24: Dedução Lógica com Base em Testes de Hipóteses para Seleção de Algoritmo de Acordo com o Conjunto Universal de Amostras de Aprendizado de RNPM

| Algoritmos de Inicialização de Pesos e Bias | Teste de Friedman | Teste de Tukey HSD Post-hoc | Teste de Wilcoxon-M-W | Dedução Lógica | Melhoria |
|---|---|---|---|---|---|
| **Proposta: HeCI** | $\mathbf{H_1}$ | $\mathbf{H_1}$ | $\mathbf{H_1}$ | **Melhoria Significativa** | **97,57 ± 0,06** |
| Rand | $\mathcal{H}_1$ | $\mathcal{H}_1$ | $\mathcal{H}_0$ | Equivalente | 95,01 ± 0,09 |
| RandS | $\mathcal{H}_1$ | $\mathcal{H}_1$ | $\mathcal{H}_0$ | Equivalente | 94,95 ± 0,09 |
| Midpoint+RS | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ | Equivalente | 94,87 ± 0,09 |
| InitNW | $\mathcal{H}_1$ | $\mathcal{H}_0$ | $\mathcal{H}_0$ | Equivalente | 94,86 ± 0,10 |
| Randsmall | $\mathcal{H}_1$ | $\mathcal{H}_1$ | $\mathcal{H}_0$ | Equivalente | 92,55 ± 0,17 |

Fonte: Produção do Autor (2017).

A Tabela 24 apresenta o resultado dos testes de hipóteses sobre os dados dos experimentos. Com Confiança $\geq 99\%$ é deduzido por inferência estatística a escolha pela proposta HeCI.

Todos os testes de hipóteses atestaram para a HeCI, respectivamente:

1. O Teste de "Friedman" atestou diferença significativa de procedimento ou caminho para chegar ao resultado, afirmando que a HeCI processou os dados de maneira diferente dos outros algoritmos usados na comparação.
2. O Teste de "Tukey HSD Post-hoc" atestou diferença significativa de média em melhoria de acurácia com o uso da HeCI, sobre os outros algoritmos usados na comparação.
3. O Teste de "Wincoxon-M-W" atestou melhor "Rank" de acurácia para o uso da HeCI, sobre os outros algoritmos usados na comparação.

Uma vez que a HeCI foi aprovada nos três testes de hipóteses, analisa-se os resultados de acurácia em faixa de valores, com base no conjunto universal de resultados dos experimentos, conforme descrito na Figura 33. A Tabela 25 descreve o range de acurácia:

Tabela 25: Range de Resultados da HeCI com 04 Tipos de "Back-Propagation", validados com "10-Fold Cross-Validation" Repetidos 30 Vezes

| "Dataset's" | Range de Acurácia da HeCI (%) | | |
|---|---|---|---|
| | Mínimo | Máximo | Moda |
| Acute Inflammation | 100,00 | 100,00 | 100,00 |
| Banknote Autentication | 98,54 | 100,00 | 100,00 |
| Cancer Breast Wisconsin Diagnostic | 94,64 | 100,00 | 98,21 |
| Iris | 86,67 | 100,00 | 100,00 |
| Parkinsons | 73,68 | 100,00 | 100,00 |
| Seeds | 80,95 | 100,00 | 100,00 |

Fonte: Produção do Autor (2017).

A Tabela 25 apresenta a estatística descritiva (valores de mínimo, máximo e moda) de resultados de 04 tipos de "Back-Propagation" com a HeCI, em um ambiente de aprendizado de máquina com "10-Fold Cross-Validation" repetidos 30 Vezes. Esses valores descrevem:

1. A acurácia mínima alcançada é de 73,68%.
2. A acurácia máxima alcançada é de 100%.
3. O alcance de acurácia máxima ocorre em todos os "dataset's".
4. A moda de resultado está concentrada entre 98,21% $\vdash\dashv$ 100%.

Na Tabela 25, em 05 dos 06 "dataset's", é registrado 100% de acurácia na moda dos resultados, apesar da "10-Fold Cross-Validation". Essa frequente ocorrência de máxima acurácia é analisada na visualização do histograma na Tabela 26:

Tabela 26: Histograma de Acurácia de RNPM treinada por "Back-Propagation" + HeCI

| Acurácia de RNPM | Frequência Absoluta | Percentual de Frequência (*) |
|---|---|---|
| 73,68% ⊢⊣ 78,95% | 180 | 2,5% |
| 80,95% ⊢⊣ 84,21% | 210 | 2,92% |
| 85,71% ⊢⊣ 86,67% | 180 | 2,5% |
| 89,47% ⊢⊣ 90,48% | 240 | 3,33% |
| 93,33% ⊢⊣ 94,74% | 450 | 6,25% |
| 95,24% ⊢⊣ 96,43% | 390 | 5,42% |
| 98,21% ⊢⊣ 99,27% | 600 | 8,33% |
| 100,00% | 4950 | 68,75% |
| Total de Amostras → | 7200 | 100,00% |

Fonte: Produção do Autor (2017).

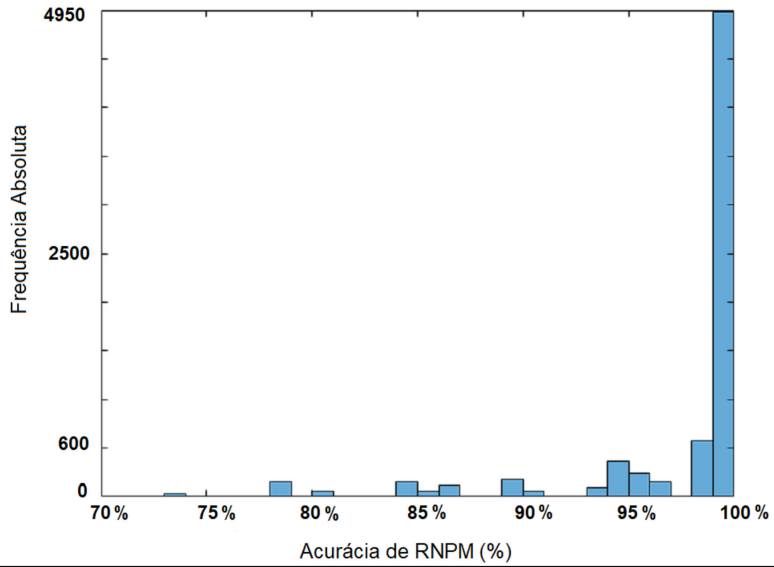
Legenda (*):
A coluna "Percentual de Frequência"é calculada conforme Walpole et al. (2011) e Weiers (2010) para o percentual de probabilidade de repetição do evento (acurácia) pela equação:

$$\lim_{Amostras \to \infty} \frac{\text{Frequência Absoluta}}{\text{Total de Amostras}} * 100 \quad (5.1)$$

A Equação 5.1 calcula a frequência de ocorrências de acurácia no histograma da Tabela 26. Apesar da rigidez da "10-Fold Cross-Validation" conforme Haykin (2001) pelo Modelo $\frac{1}{10}\sum_{i=1}^{10}(100 - erro_i)$, a HeCI alcança notoriedade de acurácia conforme demonstrado no histograma da Tabela 26.

A Tabela 26 registra a acurácia, sendo observada a frequente ocorrência de 100% de acurácia em 68,75% das vezes de uso da HeCI. A Figura 34 visualiza o gráfico do histograma:

Figura 34: Histograma de Acurácia de RNPM treinada por "Back-Propagation" + HeCI

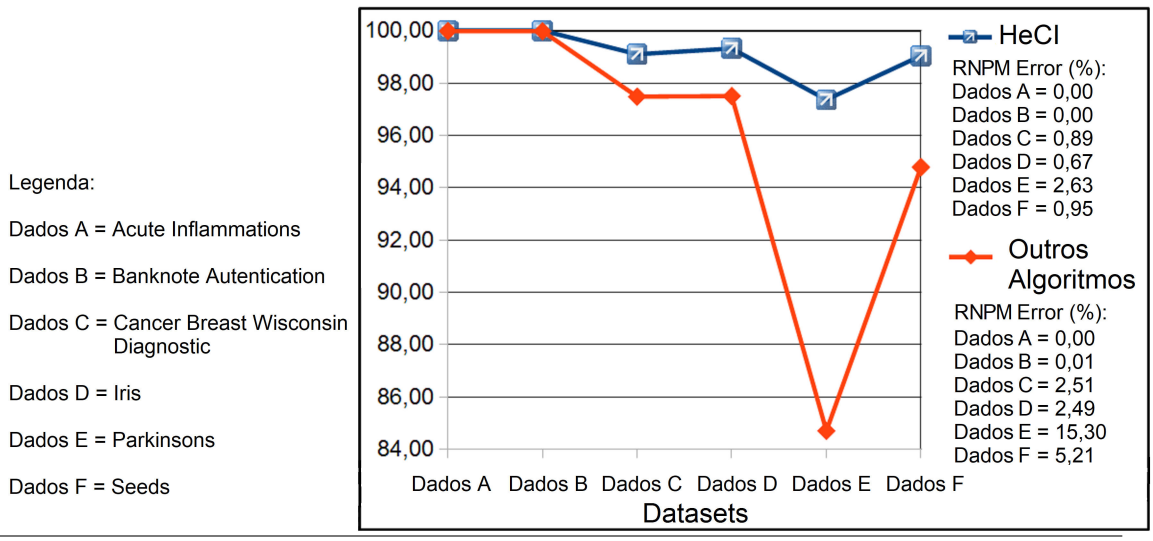



Fonte: Produção do Autor (2017).

O gráfico da Figura 34 é plotado com os dados (Acurácia de RNPM e Frequência Absoluta) da Tabela 26, e demonstra claramente a grande concentração de 100% de acurácia motivada pelo uso da HeCI.

Também é monitorado para análise, a capacidade da HeCI permanecer em 100% de acurácia ou próximo, comparando com os outros algoritmos usados nos experimentos para os mesmos "dataset's". O gráfico da Figura 35 apresenta o monitoramento:

Figura 35: Monitor de Acurácia, Erro e Comparação entre a HeCI e Outros Algoritmos




Fonte: Produção do Autor (2017).

O gráfico da Figura 35 demonstra que os resultados motivados pela HeCI são melhores que os outros algoritmos, independente de "dataset's".

Essa discussão analisou os principais pontos referente aos resultados dos experimentos:

1. A HeCI atinge máxima acurácia ou alcança a melhor acurácia.
2. A HeCI foi aprovada por três testes de hipóteses para o uso na aprendizagem de RNPM treinada por "Back-Propagation".
3. O histograma da HeCI demonstrou 68,75% de concentração dos resultados em 100% de acurácia validada com "10-Fold Cross-Validation".
4. O monitoramento de acurácia e erro demonstra os melhores resultados para a HeCI.

Diante da abordagem quantitativa na análise de resultados e dados numéricos apurados pelo uso da Estatística Descritiva e Inferencial, a HeCI demonstrou ser um algoritmo heurístico capaz de calcular os pesos e bias iniciais para o ponto de partida do "Back-Propagation" treinar RNPM, de maneira que provoca a alta acurácia, com Confiança $\geq 99\%$.

### 5.5.1 Complexidade Assintótica da HeCI

A complexidade Assintótica do algoritmo da HeCI é medida conforme Cormen et al. (2012). Símbolos usados para a medição:

"$\Psi$" Matriz de dados para treinamento de RNPM.

"$n$" Matriz de Componentes Principais.

"$\zeta$" Quantidade de Classes rotuladas para aprendizagem Supervisionada.

"$\Delta$" Matriz de Pesos e Bias.

É observado os algoritmos dos Métodos HeCI_Start e HeCI_Control descritos na Seção 4.8, procede-se a verificação do custo computacional para a execução da HeCI:

1. O algoritmo do Método HeCI_Start:
   a. A Linha (1-2) é $\zeta\ O(\Psi^2)$.
   b. As Linhas (1,3-12) são $\zeta\ O(n^3)$ porque a Análise de Componentes Principais é $O(n^3)$ de acordo com Johnstone e Lu (2009).
   c. A Linha (13,15) é $\zeta^2\ O(n^2)$. d. As Linhas (13,14,16-26) é $\zeta\ O(n^3)$ porque o Método dos Mínimos Quadrados é $O(n^3)$ de acordo com Markovsky e Huffel (2007).
   e. As Linhas (27-42) é $O(\Delta^2)$.
2. O algoritmo do Método HeCI_Control:
   a. A Linha (1) é $O(\Delta^2)$.
   b. As Linhas (9-27) é 10 $O(\Delta^2)$.
   c. A Linha (29) é 10 $O(\Delta^2)$.

Conclui-se que a HeCI é $O(max(\Psi^2, n^2, n^3, \Delta^2))$ ou $O(n^3)$.

### 5.5.2 Comparação de Resultados da HeCI com Outros Trabalhos de Pesquisa

O HeCI foi combinada com 04 tipos "Back-Propagation" para análise de resultados de experimentos computacionais apresentados na Tabela 2. De acordo com os experimentos, o "Back-Propagation Bayesian Regularization (TrainBR)" apresentou melhores resultados.

Os resultados de acurácia são comparados com outros trabalhos de pesquisas. A Tabela 27 apresenta a comparação:

Tabela 27: "Back-Propagation" + HeCI Comparando com Outros Trabalhos de Pesquisa

| "Dataset's" | Acurácia (%) | | |
|---|---|---|---|
| | **HeCI** | Outras Pesquisa | Referências |
| Acute Inflammation | **100,00%** | 90,00% | Liu et al. (2016) |
| Banknote Authentication | **100,00%** | 98,50% | Sani et al. (2016) |
| Cancer Breast Wisconsin Diagnostic | **99,11%** | 97,65% | Chen et al. (2016) |
| Iris | **99,33%** | 99,00% | Qiao et al. (2016) |
| Parkinsons | **97,37%** | 90,03% | Tiwari (2016) |
| Seeds | **99,05%** | 91,19% | Gong et al. (2016) |

Fonte: Produção do Autor (2017).
Nota: Todas as comparações considera o mesmo "dataset" e a acurácia de resolução do problema de Classificação de Dados.

A Tabela 27 apresenta a comparação dos resultados do "Back-Propagation Bayesian Regularization (TrainBR)" + HeCI com outros trabalhos de pesquisa para resolver o problema de Classificação de Dados. A HeCI alcançou melhor acurácia em todos os "dataset's".

# 6 Considerações Finais

Essa pesquisa desenvolveu uma Heurística para Calcular Pesos e Bias Iniciais (HeCI) para o ponto de partida do "Back-Propagation" treinar RNPM. Em seguida, aplicou-se a Classificação de Dados de seis "datasets", realizando seis estudos de caso.

Dessa forma, foi investigada a resolução do Problema:

> Dada a aplicação de RNPM em aprender a reconhecer padrões para classificar dados em grupos específicos, pergunta-se:
> Quais valores inicializar pesos e bias para o "Back-Propagation" convergir para a máxima acurácia?

Para resolver o Problema levantou-se duas hipóteses:

> $\mathcal{H}_0$: Inicializar pesos e bias com números Aleatórios ou método Heurístico é equivalente em acurácia e tempo de treinamento.
>
> $\mathcal{H}_1$: Inicializar pesos e bias com números Aleatórios ou método Heurístico é diferente em acurácia e tempo de treinamento.

Com foco nas hipóteses, foram combinados e comparados 06 (seis) algoritmos de inicialização de pesos e bias junto a 04 (quatro) "Back-Propagation", aplicados aos estudos de caso para a classificação de dados nas áreas da Saúde, Negócio e Botânica.

Foram usados a HeCI, 05 (cinco) tipos de algoritmos de inicialização de pesos e bias, 04 (quatro) tipos de "Back-Propagation" e seis "dataset's", em experimentos computacionais para investigação de resultados de acurácia e tempo de treinamento de RNPM.

Os resultados dos experimentos foram medidos, avaliados, comparados e testados por três testes de hipóteses para verificar se houve melhoria significativa (maior valor médio e menor desvio padrão, sem intersecção com outros valores ao aplicar o desvio padrão), respectivamente, de diferença de procedimento, média e "Rank", dos resultados de acurácia com tempo de treinamento.

Em virtude do que foi realizado, é certo que foi alcançado o Objetivo Geral:

> Desenvolver e aplicar um algoritmo Heurístico para calcular pesos e bias iniciais para o "Back-Propagation" treinar Rede Neural Perceptron Multicamadas (RNPM).

Os resultados dos experimentos apresentados na Tabela 2 e destacados em negrito com sombreamento em cinza, demonstram que a HeCI empatou com os outros algoritmos em 100% de acurácia para um "dataset", e para os outros "dataset's" alcançou a melhor acurácia, caracterizando ótimo resultado dentro do domínio dos "dataset's" usados para a solução do problema de classificação de dados. Procedeu-se então a avaliação dos resultados por testes de hipóteses.

Do ponto de vista do Teste de hipóteses de Friedman com Pós-Teste de Tukey HSD Post-hoc e o Teste de Wilcoxon-M-W, a hipótese alternativa $\mathcal{H}_1$ foi aceita para o uso da HeCI com Confiança $\geq 99\%$, sobre o conjunto universal de amostras de aprendizagem da RNPM, com incremento de melhoria significativa, dentro das áreas de estudos de casos dos "dataset's".

A HeCI alcançou significativa melhoria de acurácia e posição de $1^o$ lugar em "Rank" de resultados, atestados pelos testes de hipóteses, comparando-a aos outros cinco algoritmos que foram atestados como equivalentes em resultados.

Pela observação dos resultados quantitativos, apresentados conforme Experimentos Seção 5, destaca-se o uso da HeCI junto ao "Back-Propagation" em termos numéricos:

1. Acurácia de 100% para o "dataset Acute Inflammations" (Inflamação Aguda).
2. Acurácia de 100% para o "dataset Banknote Authentication" (Autenticação Bancária).
3. Acurácia de 99,11% para o "dataset Breast Cancer W. Diagnostic" (Câncer de Mama).
4. Acurácia de 99,33% para o "dataset Iris" (Plantas).
5. Acurácia de 97,37% para o "dataset Parkinson's" (Doença de Parkinson).
6. Acurácia de 99,05% para o "dataset Seeds" (Variedades de Trigo).
7. A Moda Estatística de resultado da HeCI ficou entre 98,21% e 100% de acurácia.
8. O Histograma da HeCI demonstra que 68,75% dos resultados são 100% de acurácia.
9. Em todos os resultados foi usado "10-Fold Cross-Validation" repetidos 30 vezes.
10. Para o conjunto universal de amostras de aprendizagem, os testes de hipótese atestaram melhoria significativa com Confiança $\geq 99\%$:
    a) O Teste de Friedman aceitou $\mathcal{H}_1$, afirmando diferença significativa de procedimento para o uso da HeCI.
    b) O Teste de Tukey Post-hoc aceitou $\mathcal{H}_1$, afirmando diferença significativa de melhoria em média de acurácia para o uso da HeCI.
    c) O Teste Wilcoxon-M-W aceitou $\mathcal{H}_1$, afirmando diferença significativa de melhoria em "Rank" de acurácia, ou seja, $1^o$ lugar para o uso HeCI.

Usa-se da argumentação quantitativa da inferência estatística, para cientificamente afirmar que por dedução, o Problema dessa Pesquisa foi resolvido - dentro do domínio de estudos de casos realizados - e por indução, para outros estudos de casos. Por isso, conclui-se:

- "Inicializar pesos e bias com números calculados por Método Heurístico é diferente em Acurácia e Tempo de Treinamento para o "Back- Propagation", produzindo uma melhoria significativa de Acurácia".

Portanto, o uso da HeCI com "Back-Propagation" produz melhoria significativa de acurácia para a RNPM, quando comparada com os outros algoritmos usados nessa pesquisa, para a classificação de dados dos "dataset's" usados.

## 6.1 Contribuições dessa Pesquisa

As seguintes contribuições foram realizadas:

1. Desenvolvimento de uma nova Heurística para Calcular Pesos e Bias Iniciais (HeCI) para o ponto de partida do "Back-Propagation" treinar RNPM:
   a) Foi feita uma nova combinação e aplicação das seguintes teorias: Análise de Componentes Principais, Métodos do Mínimos Quadrados e Função de Probabilidade de Densidade da Distribuição Normal Gaussiana;
   b) Foi criada duas Estratégias para lidar com pesos e bias, dependendo do "dataset";
   c) Foi criado um controle parcial para a quantidade de épocas de treinamento de RNPM pelo "Back-Propagation" e avaliação de acurácia;
   d) Foi aplicada uma nova maneira para o "Back-Propagation" sair de mínimos locais: o uso da Função de Probabilidade de Densidade da Distribuição Normal Gaussiana sobre os pesos de todas as camadas da RNPM.
2. Foi alcança melhoria significativa de acurácia sobre outros trabalhos de pesquisa, conforme Tabela 27, na classificação de dados, referente os mesmos "dataset's":
   a) "Acute Inflammation" (Inflamação Aguda).
   b) "Banknote Authentication" (Autenticação Bancária).
   c) "Breast Cancer Wisconsin Diagnostic" (Diagnóstico de Câncer de Mama).
   d) "Iris" (Plantas).
   e) "Parkinson's" (Doença de Parkinson).
   f) "Seeds" (Variedades de Trigo).

## 6.2 Trabalhos Futuros

Futuras investigações podem ser realizadas decorrentes dessa pesquisa:

1. Aplicação em outros estudos de casos para Classificação de Dados;
2. Modificação da HeCI para uma arquitetura de RNPM de duas ou mais camadas ocultas;
3. Uso da HeCI junto a outros tipos de "Back-Propagation";
4. Uso da HeCI junto ao treinamento de RNPM para Regressão de Dados, Classificação de Dados Multi-Rótulo e Aprendizagem Semi-Supervisionada;
5. Adaptação da HeCI para o uso junto ao treinamento de Redes Neurais Convolucionais, Redes Neurais Profundas e Máquina de Vetor de Suporte (SVM);
6. Entre outros.

# Referências

AHACHAD, A.; PéREZ, L. Álvarez; FIGUEIRAS-VIDAL, A. R. Boosting ensembles with controlled emphasis intensity. *Pattern Recognition Letters*, v. 88, p. 1 – 5, 2017. ISSN 0167-8655. Citado na página 21.

ARABASADI, Z.; ALIZADEHSANI, R.; ROSHANZAMIR, M.; MOOSAEI, H.; YARIFARD, A. A. Computer aided decision making for heart disease detection using hybrid neural network-genetic algorithm. *Computer Methods and Programs in Biomedicine*, v. 141, p. 19 – 26, 2017. ISSN 0169-2607. Citado na página 21.

BEALE, M. H.; HAGAN, M. T.; DEMUTH, H. B. *Neural Network Toolbox Reference*. Natick, MA, USA: The MathWorks, Inc, 2016. Disponível em: <http://www.mathworks.com>. Citado 6 vezes nas páginas 55, 57, 86, 87, 90 e 91.

BELFIORE, P.; FáVERO, L. P. *Pesquisa Operacional*: Para cursos de engenharia. Rio de Janeiro: Elsevier, 2013. Citado 4 vezes nas páginas 25, 27, 28 e 29.

BIAGI, M. C. *Pesquisa Científica*: Roteiro prático para desenvolver projetos e teses. Curitiba: Juruá, 2012. Citado na página 22.

BOTTOU, L. Stochastic gradient descent tricks. In: *Neural Networks: Tricks of the Trade*. [S.l.]: Springer, 2012. p. 421–436. Citado 2 vezes nas páginas 55 e 86.

BRAGA, A. d. P.; CARVALHO, A. P. d. L. F.; LUDERMIR, T. B. *Redes Neurais Artificiais*: Teoria e aplicações. 2. ed. Rio de Janeiro: LTC, 2007. Citado 6 vezes nas páginas 16, 20, 32, 35, 36 e 50.

BROOKSHEAR, J. G. *Ciência da Computação*: uma visão abrangente. 11. ed. Porto Alegre: Bookman, 2013. Citado na página 14.

BUNTINE, W. L.; WEIGEND, A. S. Bayesian back-propagation. *Complex systems*, v. 5, n. 6, p. 603–643, 1991. Citado 2 vezes nas páginas 55 e 86.

CHAPRA, S. C.; CANALE, R. P. *Numeral Methods for Engineers*. 6. ed. [S.l.]: McGraw-Hill, 2009. Citado na página 72.

CHEN, Y.; ZENG, Y.; LUO, F.; YUAN, Z. A new algorithm to optimize maximal information coeÿcient. *PLoS ONE*, Public Library of Science, v. 11, n. 6, p. 1–13, 06 2016. Citado na página 103.

CHESNOKOV, Y. V. Complexity and spectral analysis of the heart rate variability dynamics for distant prediction of paroxysmal atrial fib illation with artificia intelligence methods. *Artificial intelligence in medicine*, Elsevier, v. 43, n. 2, p. 151–165, 2008. Disponível em: <http://www.intl.elsevierhealth.com/journals/aiim>. Citado na página 21.

CIRESAN, D. C.; MEIER, U.; GAMBARDELLA, L. M.; SCHMIDHUBER, J. Deep big multilayer perceptrons for digit recognition. In: *Neural Networks: Tricks of the Trade*. Springer, 2012. v. 7700, p. 581–598. Disponível em: <http://link.springer.com/chapter/10.1007%2F978-3-642-35289-8_31>. Citado na página 21.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Algoritmos Teoria e Prática*. 3. ed. Rio de Janeiro: Elsevier, 2012. Citado 3 vezes nas páginas 14, 28 e 102.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, [S.l.], v. 2, n. 4, p. 303–314, 1989. Citado na página 33.

CZERNIAK, J.; ZARZYCKI, H. Application of rough sets in the presumptive diagnosis of urinary system diseases. In: *Artificial intelligence and security in computing systems*. [S.l.]: Springer, 2003. p. 41–51. Citado 2 vezes nas páginas 84 e 85.

DAILEY, M. N.; COTTRELL, G. W.; PADGETT, C. A mixture of experts model exhibiting prosopagnosia. In: *In Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*. [S.l.]: Erlbaum, 1997. p. 155–160. Citado na página 42.

DAO, V. N.; VEMURI, V. A performance comparison of di˙erent back propagation neural networks methods in computer network intrusion detection. *Differential equations and dynamical systems*, v. 10, n. 1&2, p. 201–214, 2002. Citado 2 vezes nas páginas 55 e 86.

DEMŠAR, J. Statistical comparisons of classifier over multiple data sets. *Journal of Machine Learning Research*, v. 7, n. Jan, p. 1–30, 2006. Citado na página 91.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. d. L. F. d. *Inteligência Artificial*: Uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, 2011. Citado 22 vezes nas páginas 14, 15, 16, 18, 32, 34, 36, 40, 41, 44, 45, 46, 48, 56, 58, 63, 64, 65, 67, 76, 83 e 84.

FILHO, W. d. P. P. *Engenharia de Software*: fundamentos, métodos e pradões. 3. ed. Rio de Janeiro: LTC, 2009. Citado na página 23.

GEBSER, M.; KAUFMANN, B.; OTERO, R.; ROMERO, J.; SCHAUB, T.; WANKO, P. Domain-specifi heuristics in answer set programming. In: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, 2013. (AAAI'13), p. 350–356. Disponível em: <http://dl.acm.org/citation.cfm?id=2891460.2891509>. Citado na página 73.

GLOROT, X.; BENGIO, Y. Understanding the diÿculty of training deep feedforward neural networks. In: *Aistats*. [S.l.: s.n.], 2010. v. 9, p. 249–256. Citado na página 21.

GONG, A.; GAO, Y.; MA, X.; GONG, W.; LI, H.; GAO, Z. An optimized artificia bee colony algorithm for clustering. *International Journal of Control and Automation*, v. 9, n. 4, p. 107–116, 2016. Citado na página 103.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 42.

HAMMER, Ø. *Paleontological Statistics Reference Manual*. [S.l.]: Natural History Museum. University of Oslo, Oslo, 2013. Citado 5 vezes nas páginas 19, 23, 87, 90 e 91.

HAYKIN, S. *Redes Neurais*: princípios e práticas. 2. ed. Porto Alegre: Bookman, 2001. Citado 28 vezes nas páginas 14, 15, 16, 18, 32, 34, 35, 36, 37, 39, 40, 41, 43, 44, 47, 49, 50, 51, 52, 53, 56, 68, 75, 76, 83, 86, 87 e 101.

HAZRA, A.; GOGTAY, N. Biostatistics series module 3: Comparing groups: Numerical variables. *Indian journal of dermatology*, Medknow Publications, v. 61, n. 3, p. 251, 2016. Citado na página 91.

HILLIER, F. S.; LIEBERMAN, G. J. *Introdução à Pesquisa Operacional*. 8. ed. Porto Alegre: AMGH, 2006. Citado 7 vezes nas páginas 14, 25, 26, 28, 29, 30 e 31.

JOHNSTONE, I. M.; LU, A. Y. On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, Taylor & Francis, v. 104, n. 486, p. 682–693, 2009. Citado 2 vezes nas páginas 68 e 103.

JOLLIFFE, I. T. *Principal Component Analysis*. 2. ed. [S.l.]: Springer, 2002. ((Springer series in statistics)). Citado na página 68.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 374, n. 2065, 2016. Citado na página 68.

KAMRAN, M.; HAIDER, S.; AKRAM, T.; NAQVI, S.; HE, S. Prediction of {IV} curves for a superconducting thin fil using artificia neural networks. *Superlattices and Microstructures*, v. 95, p. 88 – 94, 2016. ISSN 0749-6036. Citado 2 vezes nas páginas 55 e 86.

KOLMAN, B.; HILL, D. R. *Introdução à Álgebra Linear com Aplicações*. 8. ed. Rio de Janeiro: LTC, 2014. 664 p. Citado na página 69.

KOVáCS, Z. L. *Redes Neurais Artificiais*: Fundamentos e aplicações. 4. ed. São Paulo: Livraria da Física, 2006. Citado na página 16.

LARSON, R.; FARBER, B. *Estatística Aplicada*. 4. ed. São Paulo: Pearson Prentice Hall, 2010. 638 p. Citado 4 vezes nas páginas 22, 29, 46 e 48.

LAUDON, K. C.; LAUDON, J. P. *Sistemas de Informação Gerenciais*. 11. ed. São Paulo: Pearson Education do Brasil, 2014. Citado 2 vezes nas páginas 30 e 31.

LAY, D. C. *Álgebra Linear e suas Aplicações*. 4. ed. Rio de Janeiro: LTC, 2013. 445 p. Citado na página 68.

LICHMAN, M. *(UCI) Machine Learning Repository*. 2013. Disponível em: <http://archive.ics.uci.edu/ml>. Citado 2 vezes nas páginas 84 e 85.

LITTLE, M. A.; MCSHARRY, P. E.; ROBERTS, S. J.; COSTELLO, D. A.; MOROZ, I. M. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, v. 6, n. 1, p. 23, 2007. Citado 2 vezes nas páginas 84 e 85.

LIU, S.-H.; SHEN, L.-Z.; HUANG, D.-C. A three-stage framework for clustering mixed data. *WSEAS Transaction on Systems*, v. 15, n. E-ISSN: 2224-2678, 2016. Citado na página 103.

LIVNI ROI, S. S.-S.; SHAMIR, O. On the computational eÿciency of training neural networks. *Advances in Neural Information Processing Systems*, [S.l.], p. 855–863, 2014. Disponível em: <http://arxiv.org/abs/1410.1141>. Citado 4 vezes nas páginas 15, 16, 17 e 18.

LUGER, G. F. *Inteligência Artificial*. 6. ed. São Paulo: Pearson, 2013. 631 p. Citado 15 vezes nas páginas 15, 16, 17, 20, 33, 35, 37, 39, 40, 41, 55, 56, 61, 76 e 83.

MARKOVSKY, I.; HUFFEL, S. V. Overview of total least-squares methods. *Signal Processing*, v. 87, n. 10, p. 2283 – 2302, 2007. ISSN 0165-1684. Special Section: Total Least Squares and Errors-in-Variables Modeling. Citado 2 vezes nas páginas 72 e 103.

MATHWORK, I. *MATLAB Function Reference*. Natick, MA, USA: The MathWorks, Inc, 2016. Disponível em: <http://www.mathworks.com>. Citado 3 vezes nas páginas 57, 86 e 87.

MATHWORKS. *MathWorks is the Leading Developer of Mathematical Computing Software for Engineers and Scientists*. 2016b. ed. Natick, MA, USA: The MathWorks, Inc, 2016. <http://www.mathworks.com>. Citado 4 vezes nas páginas 19, 23, 74 e 84.

MCGEOCH, C. C. *A Guide to Experimental Algorithmics*. New York: Cambride University Press, 2012. Citado 2 vezes nas páginas 23 e 84.

MITCHELL, T. M. *Machine Learning*. Porland, OR: McGraw-Hill, 1997. Citado na página 43.

MURRU, N.; ROSSINI, R. A bayesian approach for initialization of weights in backpropagation neural net with application to character recognition. *Neurocomputing*, v. 193, p. 92 – 105, 2016. ISSN 0925-2312. Citado na página 21.

NAHM, F. S. Nonparametric statistical tests for the continuous data: the basic concept and the practical use. *Korean journal of anesthesiology*, v. 69, n. 1, p. 8–14, 2016. Citado 5 vezes nas páginas 19, 23, 87, 90 e 91.

NGUGEN, D.; WIDRAW, B. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In: *Procceeding of the International Joint Conference on Neural Networks*. [S.l.: s.n.], 1990. v. 3, p. 21–26. Citado 2 vezes nas páginas 57 e 86.

NIVIO, Z. *Projeto de Algoritmos*: Com implementação em java e c++. São Paulo: Cengage Learning, 2011. 621 p. Citado na página 28.

PANDEY, P.; GOVIND, R. Analysis of randomized performance of bias parameters and activation function of extreme learning machine. *International Journal of Computer Applications*, v. 135, n. 1, p. 23–28, 2016. Disponível em: <http://www.ijcaonline.org>. Citado na página 21.

PAVELKA, A.; PROCHAZKA, A. Algorithms for initialization of neural network weights. In: *In Proceedings of the 12th Annual Conference, MATLAB*. [S.l.: s.n.], 2004. p. 453–459. Citado 2 vezes nas páginas 57 e 86.

PIZZOLATO, N. D.; GANDOLPHO, A. A. *Técnicas de Otimização*. Rio de Janeiro: LTC, 2013. Citado 4 vezes nas páginas 14, 25, 26 e 27.

PLAGIANAKOS, V.; MAGOULAS, G.; VRAHATIS, M. Learning rate adaptation in stochastic gradient descent. In: *Advances in convex analysis and global optimization*. [S.l.]: Springer, 2001. p. 433–444. Citado 2 vezes nas páginas 55 e 86.

POLYA, G. *A Arte de Resolver Problemas*. Rio de Janeiro: Interciência, 2006. 203 p. Citado na página 26.

POPPER, K. R. *A Lógica da Pesquisa Científica*. 18. ed. São Paulo: Cultrix, 2007. 565 p. Citado na página 22.

QIAO, J.; LI, S.; LI, W. Mutual information based weight initialization method for sigmoidal feedforward neural networks. *Neurocomputing*, Elsevier, 2016. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2016.05.054>. Citado 2 vezes nas páginas 21 e 103.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In: *Neural Networks, 1993., IEEE International Conference on*. [s.n.], 1993. p. 586–591 vol.1. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=298623>. Citado 2 vezes nas páginas 55 e 86.

RIEDMILLER, M.; RPROP, I. *Rprop - Description and Implementation Details*. 1994. Disponível em: <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.3428&rank=13>. Citado 2 vezes nas páginas 55 e 86.

RODAN, A.; FARIS, H.; ALQATAWNA, J. Optimizing feedforward neural networks using biogeography based optimization for e-mail spam identification *International Journal of Communications, Network and System Sciences*, Scientifi Research Publishing, v. 9, n. 1, p. 19–28, 2016. Citado na página 21.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning Internal Representations by Error Propagation. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Foundations, d. e. rumelhart and j. l. mcclelland, eds. MIT Press, Cambridge, MA: Cengage Learning, 1986. v. 1. (Mit Press Computational Models Of Cognition And Perception Series, v. 1). P. 318-362. Citado 4 vezes nas páginas 15, 18, 53 e 67.

RUSSEL, S.; NORVIG, P. *Inteligência Artificial*. 3. ed. Rio de Janeiro: Elsevier, 2013. Citado 21 vezes nas páginas 14, 16, 17, 32, 33, 34, 35, 44, 47, 49, 50, 51, 53, 54, 55, 56, 58, 59, 73, 76 e 83.

SAMARASINGHE, S. *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. [S.l.]: CRC Press, 2006. Citado 3 vezes nas páginas 15, 18 e 87.

SANI, K.; WINARNO, W. W.; FAUZIATI, S. Analisis perbandingan algoritma classificat on untuk authentication uang kertas (studi kasus: Banknote authentication). *Jurnal Informatika*, v. 10, n. 1, 2016. Citado na página 103.

SAREMI, S.; MIRJALILI, S.; LEWIS, A. Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software*, v. 105, p. 30 – 47, 2017. Citado na página 91.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistema de Banco de Dados*. 5. ed. Rio de Janeiro: Elsevier, 2006. Citado na página 45.

SILVA, D. A.; SILVA, J. P.; NETO, A. R. R. Novel approaches using evolutionary computation for sparse least square support vector machines. *Neurocomputing*, v. 168, p. 908 – 916, 2015. ISSN 0925-2312. Citado na página 72.

SILVA, I. N. d.; SPATI, D. H.; FALUZINO, R. A. *Redes Neurais Artificiais para Engenheiros e Ciência Aplicada*: Curso prático. São Paulo: Artliber, 2010. Citado 13 vezes nas páginas 16, 20, 34, 35, 36, 49, 50, 52, 53, 56, 58, 59 e 60.

SINGH, A.; SAXENA, P.; LALWANI, S. A study of various training algorithms on neural network for angle based triangular problem. *International Journal of Computer Applications*, Foundation of Computer Science, v. 71, n. 13, 2013. Citado 2 vezes nas páginas 57 e 86.

SODHI, S. S.; CHANDRA, P. Interval based weight initialization method for sigmoidal feedforward artificia neural networks. *AASRI Procedia*, v. 6, p. 19 – 25, 2014. ISSN 2212-6716. Citado na página 21.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011. Citado 3 vezes nas páginas 30, 31 e 58.

STEWART, J. *Cálculo*. 7. ed. [S.l.]: Cengage Learning, 2013. v. 1. Citado 5 vezes nas páginas 27, 33, 37, 38 e 39.

SUNDARAM, N. M.; RAMESH, P. N. Optimization of training phase of elman neural networks by suitable adjustments on the network parameters. In: *Proceedings of the International Conference on Systems, Science, Control, Communication, Engineering and Technology, ICSSCCET*. [S.l.: s.n.], 2015. p. 229–235 Print. Citado na página 21.

TAN, P.-N. et al. *Introduction to data mining*. [S.l.]: Pearson Education India, 2006. Citado na página 21.

TIWARI, A. K. Machine learning based approaches for prediction of parkinson's disease. *An International Journal (MLAIJ)*, v. 3, n. 2, p. 33 – 39, 2016. Citado na página 103.

VALIANT, L. G. A theory of the learnable. *Commun. ACM*, ACM, New York, NY, USA, v. 27, n. 11, p. 1134–1142, nov. 1984. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1968.1972>. Citado na página 47.

WALPOLE, R. E.; MYERS, R. H.; MYERS, S. L.; YE, K. *Probability and statistics for engineers and scientists*. 9. ed. [S.l.]: Pearson, 2011. Citado 4 vezes nas páginas 74, 90, 91 e 101.

WAZLAWICK, R. S. *Metodologia de Pesquisa para Ciência da Computação*. Rio de Janeiro: Elsevier, 2008. Citado 2 vezes nas páginas 16 e 22.

WEIERS, R. M. *Introduction to business statistics*. [S.l.]: Cengage Learning, 2010. Citado na página 101.

WITTEN, I. H.; FRANK, E.; HALL, M. A. *Data Mining*: Practical machine learning tools and techniques. 3. ed. Burlington, MA: Elsevier, 2011. Citado 3 vezes nas páginas 58, 61 e 64.

ZAKI, M. J.; JR, W. M. *Data mining and analysis: fundamental concepts and algorithms*. [S.l.]: Cambridge University Press, 2014. Citado na página 84.

ZHANG, Y.; LEE, J. D.; JORDAN, M. I. L1-regularized neural networks are improperly learnable in polynomial time. *arXiv preprint arXiv:1510.03528*, 2015. Disponível em: <http://arxiv.org/pdf/1510.03528v1.pdf>. Citado 4 vezes nas páginas 15, 16, 17 e 18.

ZHAO, Z.; AL. et. Investigation and improvement of multi-layer perceptron neural networks for credit scoring. *Expert Systems with Applications*, v. 42, n. 7, p. 3508 – 3516, 2015. ISSN 0957-4174. Citado na página 21.