UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Programa de Pós-Graduação em Engenharia Elétrica

Identificação de Adulterantes no Leite Bovino Por Meio do Uso de Redes Neurais Artificiais e Propagação de Ondas Mecânicas

Rodrigo Hans Mazer

Bauru – SP

2021

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Programa de Pós-Graduação em Engenharia Elétrica

Identificação de Adulterantes no Leite Bovino Por Meio do Uso de Redes Neurais Artificiais e Propagação de Ondas Mecânicas

Rodrigo Hans Mazer

Dissertação apresentada à Faculdade de Engenharia de Bauru, como requisito para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. José Alfredo Covolan Ulson

Coorientadora: Dra. Maria Izabel Merino de Medeiros

Bauru – SP

2021

Mazer, Rodrigo Hans.
Identificação de Adulterantes no Leite Bovino por Meio do Uso de Redes Neurais Artificiais e Propagação de Ondas Mecânicas / Rodrigo Hans Mazer, 2021
87 f. : il.

Orientador: Prof. Dr. José Alfredo Covolan Ulson

Dissertação (Mestrado) – Universidade Estadual Paulista. Faculdade de Engenharia, Bauru, 2021.

1. Adulteração de leite bovino. 2. Técnica cromática. 3. Redes neurais artificiais. 4. Reconhecimento de padrões. I. Universidade Estadual Paulista. Faculdade de Engenharia. II. Título.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Bauru

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE RODRIGO HANS MAZER, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, DA FACULDADE DE ENGENHARIA - CÂMPUS DE BAURU.

Aos 11 dias do mês de agosto do ano de 2021, às 09:30 horas, por meio de Videoconferência, realizou-se a defesa de DISSERTAÇÃO DE MESTRADO de RODRIGO HANS MAZER, intitulada **IDENTIFICAÇÃO DE ADULTERANTES NO LEITE BOVINO POR MEIO DO USO DE REDES NEURAIS ARTIFICIAIS E PROPAGAÇÃO DE ONDAS MECÂNICAS**. A Comissão Examinadora foi constituida pelos seguintes membros: Prof. Dr. JOSÉ ALFREDO COVOLAN ULSON (Orientador(a) - Participação Virtual) do(a) Departamento de Engenharia Elétrica / Faculdade de Engenharia de Bauru - UNESP, Prof. Dr. MARCELO NICOLETTI FRANCHIN (Participação Virtual) do(a) Departamento de Engenharia Elétrica / Faculdade de Engenharia de Bauru - UNESP, Prof. Dr. ADRIANO DE SOUZA MARQUES (Participação Virtual) do(a) Engenharia da Computação / Instituto Federal de São Paulo - IFSP - Câmpus Birigui. Após a exposição pelo mestrando e arguição pelos membros da Comissão Examinadora que participaram do ato, de forma presencial e/ou virtual, o discente recebeu o conceito final: aprovado _ _ _ _ _ _ _ . Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelo(a) Presidente(a) da Comissão Examinadora.

Prof. Dr. JOSÉ ALFREDO COVOLAN ULSON

Prof. Dr. José Alfredo Covolan Ulson
Vice-Diretor - FE - Campus de Bauru

Agradecimentos

A Deus.

À minha família e à minha namorada por todo o suporte.

Ao meu orientador, professor Dr. José Alfredo Covolan Ulson, pela oportunidade concedida, orientação técnica, ensinamentos em sala de aula, paciência e cobrança durante todo o trabalho.

À minha coorientadora Dra. Maria Izabel Merino de Medeiros pela atenção e orientação técnica referente ao leite.

Ao meu grande amigo Me. Marcos Messias dos Santos Júnior, por toda sua ajuda e prestatividade.

A todos os professores e amigos que tive a honra de conhecer durante o curso.

Aos meus grandes amigos, Cíntia e Niltom, pelas dicas sobre o Mestrado.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio financeiro recebido nesta pesquisa.

RESUMO

As adulterações alimentares causam prejuízos econômicos e de saúde, em âmbito mundial. O leite é um dos alimentos mais consumidos da humanidade por possuir grande poder nutricional e é alvo de inúmeras fraudes, difíceis de serem reconhecidas, especialmente quando são utilizados baixos percentuais de contaminações. Atualmente, as tecnologias capazes de inspeção de adulterantes no leite bovino são onerosas e lentas, questões motivacionais para a busca de novas alternativas para solucionar o problema. Nesse contexto, foi proposto um trabalho que desenvolveu um recipiente construído com diafragmas piezelétricos de baixo custo, propagando ondas mecânicas em amostras de leite contaminadas por bicarbonato de sódio, ureia e peróxido de hidrogênio, em proporções de 0,5% e 1%, cujos sinais elétricos foram adquiridos para processamentos computacionais. Por meio da utilização da técnica cromática, foi feita a decomposição dos sinais em três parâmetros que apresentaram características em que se foi possível diferenciar adulterações perante amostras de leite puro. Partindo-se desses dados, o presente trabalho utilizou oito ferramentas de Redes Neurais Artificiais (RNAs) de arquiteturas distintas para identificar cada amostra: Rede Neural Perceprom Multicamadas (RNPMC), Rede Neural de Retropropagaçãom em Cascata (RNRC), Rede Neural Elman de Retropropagação (RNER), Rede Neural de Reconhecimento de Padrões (RNRP), Rede Neural Probabilística (RNP), Rede Neural de Regressão Generalizada (RNRG), Rede Neural de Base Radial (RNBR) e Rede Neural de Base Radial Exata (RNBRE). Os dados, dispostos em 2 matrizes de 700 linhas por 4 colunas, foram divididos em 50% para o treinamento das redes e 50% o para teste, para todas as redes. Para efeitos comparativos individuais, também foram feitas execuções extras para as redes RNPMC, RNRC, RNER e RNRP, dividindo-se os dados em 75% para treinamento, 25% para validação e 25% para teste, pois as arquiteturas das mesmas permitem esta possibilidade. Os ensaios consistiram em 10 parametrizações diferentes para cada RNA, em que buscou-se os melhores desempenhos, evidenciados pelas quantidades e percentuais de acertos, erros quadráticos médios e regressões, para as devidas comparações individuais e entre as redes neurais. Os resultados das melhores parametrizações obtidas mostraram que as redes RNRP, RNER, RNPMC e RNRC foram capazes de identificar mais de 98% das amostras. As redes RNRG, RNP e RNBR obtiveram acertos de 86,71%, 86,57% e 85%, respectivamente, e a RNBRE, apenas 46,57%. Desta forma, a proposta desenvolvida demonstrou-se promissora e benéfica no apoio ao trabalho de identificação de adulterantes no leite, apresentando quatro opções com percentuais de acertos acima de 98%.

Palavras-chave: adulteração de leite bovino, técnica cromática, redes neurais artificiais, reconhecimento de padrões.

ABSTRACT

Food adulterations cause economic and health damage worldwide. Milk is one of the most consumed foods in humanity because it has great nutritional power and is the target of numerous frauds, difficult to be recognized, especially when low percentages of contamination are used. Currently, technologies capable of inspecting adulterants in bovine milk are costly and slow, motivational issues for the search for new alternatives to solve the problem. In this context, a study was proposed that developed a container built with low-cost piezoelectric diaphragms, propagating mechanical waves in milk samples contaminated by sodium bicarbonate, urea and hydrogen peroxide, in proportions of 0.5% and 1%, whose electrical signals were acquired for computational processing. Through the use of the chromatic technique, the signals were decomposed into three parameters that presented characteristics in which it was possible to differentiate adulterations toward samples of pure milk. Based on these data, the present work used eight tools of Artificial Neural Networks (ANNs) of different architectures to identify each sample: Perceprom Multilayer Neural Network (PMLNN), Cascade Backpropagation Neural Network (CBNN), Elman Backpropagation Neural Network (EBNN), Pattern Recognition Neural Network (PRNN), Probabilistic Neural Network (PNN), Generalized Regression Neural Network (GRNN), Radial Base Neural Network (RBNN) and Exact Radial Base Neural Network (ERBNN). The data, arranged in 2 matrices of 700 rows by 4 columns, were divided into 50% for training the networks and 50% for testing, for all networks. For individual comparative purposes, extra runs were also made for the PMLNN, CBNN, EBNN and PRNN networks, dividing the data into 75% for training, 25% for validation and 25% for testing, as their architectures allow this possibility . The experiments consisted of 10 different parameterizations for each ANN, in which the best performances were sought, evidenced by the amounts and percentages of correct answers, mean square errors and regressions, for the appropriate comparisons between individual and neural networks. The results of the best parameterizations obtained showed that the PRNN, EBNN, PMLNN and CBNN networks were able to identify more than 98% of the samples. The networks GRNN, PNN and RBNN obtained correct answers of 86.71%, 86.57% and 85%, respectively, and the ERBNN, only 46.57%. Thus, the developed proposal proved to be promising and beneficial in supporting the work of identifying adulterants in milk, presenting four options with percentages of correct answers above 98%.

Keywords: adulteration of bovine milk, chromatic technique, artificial neural networks, pattern recognition.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E DEFINIÇÕES

Sigla	Definição
3D	Representação gráfica tridimensional
EQM	Erro Quadrático Médio
DAQ	Dispositivo de Aquisição de Dados (*Data Acquisition*)
DE	Evolução Diferencial (*Diferencial Evolution*)
IA	Inteligência Artificial
RNBR	Rede Neural de Base Radial
RNBRE	Rede Neural de Base Radial Exata
RNEN	Rede Neural Elman de Retropropagação
RNP	Rede Neural Probabilística
RNPMC	Rede Neural Perceptron Multicamadas
RNRC	Rede Neural de Retropropagação em Cascata
RNRG	Rede Neural de Regressão Generalizada
RNRP	Rede Neural de Reconhecimento de Padrões
RGB	Vermelho, Verde, Azul (*Red, Green, Blue)* - método combinativo de cores que compõem um espectro cromático
ReLU	Unidade Linear Retificada (*Rectified Linear Unit*) - espécie de função de ativação
RNA	Rede Neural Artificial
RNAs	Redes Neurais Artificiais
TC	Técnica Cromática
USB	Barramento Serial Universal (*Universal Serial Bus*)

LISTA DE SÍMBOLOS

Símbolo	Descrição	Unidade
A	Vetor com j-ésimo elemento na função de ativação *softmax*	
A^i/B^i	Incrementos para cada vez que uma observação de treinamento Y^j para o *cluster i* é encontrada	
b_k	*Bias* aplicado ao neurônio *k* (aumenta ou diminui função de ativação)	
B	Largura de banda equivalente	Rad/s
c_{jn}	Vetor que define o centro da função de uma base radial	
D_i^2	Função escalar de ponderação exponencial de acordo com sua distância euclidiana para os padrões de entradas *X* da rede RNRG	
$e(n)$	Erros de uma RNA com algoritmo de Levenberg-Marquardt, em relação aos pesos com *n* linhas	
E_b	Energia	J
f_i	Equação de uma rede RNP	
F_i	Transformada de *Fourier* dos sinais gerais de *f(t)*	$J \cdot Hz$
J	Matriz jacobiana dos vetores de erros *e(n)* de uma RNA com algoritmo de Levenberg-Marquardt	
M_i	Quantidade de vetores no treinamento na classe *i* de uma rede RNP	
N	Tamanho total de uma categoria na camada da função de ativação *softmax* que possui quantidade de neurônios igual a N	
p	Dimensão do vetor *x* de uma rede RNP	
u_k	Saída do combinador linear	
v_k	Potencial de ativação do neurônio *k*	
w_{kj}	Peso sináptico da sinapse *j* pertencente ao neurônio *k*	
x_j	Sinal de entrada da sinapse *j*	
y_i	Valor obtido na saída de uma RNA para cálculo de EQM	
y_p	Valor desejado para saída de uma RNA para cálculo de EQM	
y_k	Sinal de saída do neurônio *k*	
Z_i	Vetor resultante da multiplicação do vetor de entrada pelo vetor de pesos na unidade padrão de uma rede RNP	
α	Termo de momento (varia de 0 a 1)	
$\varphi(.)$	Função de ativação (ou transferência)	
μ	Variável inversamente proporcinal à função custo em interações sucessivas na implementação computacional (para uma RNA com algoritmo de Levenberg-Marquardt)	

η	Taxa de aprendizado	
δ	Gradiente local	
σ	Taxa de propagação (ou fator de suavidade/alisamento) de uma função	
σ_{jn}	Vetor responsável pela taxa de decrescimento da gaussiana junto a cada coordenada do espaço de entrada, em uma RNA de base radial	
ω_c	Banda média	Rad/s
ω_i	Frequência angular do sinal *f[n]* no tempo discreto	Rad/s

PUBLICAÇÃO

Trabalho publicado em revista científica

Mazer, R.H.; Medeiros, M.I.M.; Ulson, J.A.C., Identificação de adulterantes no leite bovino por meio de redes neurais artificiais e técnica cromática. **Revista Científica Multidisciplinar Núcleo do Conhecimento**, 19 de Fevereiro de 2021. 53-68. Disponível em https://www.nucleodoconhecimento.com.br/engenharia-eletrica/adulterantes-no-leite. DOI: 10.32749.

SUMÁRIO

Capítulo 1 – Introdução

O leite está presente na alimentação de cerca de 80% da população mundial, possuindo diversos elementos em sua composição, como água, lipídios, carboidratos, sais, no qual contribui com 5% da energia, 10% da proteína e 9% da gordura consumida no planeta (JENNESS, 1988; EMBRAPA, 2020).

Devido à sua grande importância econômica, o leite é alvo de ações fraudulentas que podem ser praticadas durante a etapa de produção, transporte até o laticínio, dentro da própria indústria láctea, e/ou nos produtos já expostos ao consumidor. Fraude é qualquer ação que adicione ou subtraia substâncias do leite. As principais fraudes encontadas são por adição de água, reconstituintes, conservantes ou neutralizantes, que consequentemente, podem prejudicar muito a saúde dos seres humanos, sendo capazes de causar intoxicação aguda, dermatite, dores de cabeça, tontura, asma, gastrite, náuseas e convulsões (HANDFORD, *et al.*, 2016). Os adulterantes podem ser produtos adicionados ao leite em sua própria composição natural, mas que excedem proporções permitidas pelos órgãos reguladores governamentais, como a ureia (TRIVEDI, *et al.*, 2009; GREGORIO, 2019) ou outros produtos, como o bicarbonato de sódio, utilizado na neutralização do pH do leite no processo de controle de qualidade (STAPLES e LOUGH, 1989). O peróxido de hidrogênio é utilizado em fraudes para prolongar o prazo de validade do leite, mas sabe-se que a indústria faz uso controlado deste produto na higienização de embalagens e equipamentos (EU, 2003; DE ARAUJO, *et al.*, 2021). Este trabalho teve como foco as adulterações com bicarbonato de sódio, ureia e peróxido de hidrogênio, portanto, por serem mais utilizados na cadeia do leite. Todavia, existem outros adulterantes em potencial: água, detergentes, leite em pó, sal, açúcar, formalina, hipoclorito e gorduras (HARDING, *et al.*, 1995; ABRANTES, *et al.*, 2014).

Com o aprimoramento das fraudes, as indústrias e órgãos de fiscalização desenvolveram diversas técnicas para identificação de adulterações no leite. A grande maioria das inspeções são caras e lentas, por se tratarem de análises físico-químicas laboratoriais (GREGORIO, 2019; SANTOS JUNIOR, 2019). As pesquisas sobre alternativas para avaliações da qualidade do leite, entretanto, são vastas, e incluem: narizes eletrônicos capazes de identificar água misturada ao leite, bem como o envelhecimento do mesmo (TOKO, 1998); línguas eletrônicas que podem diferenciar marcas de leite (HRU KAR, *et al.*, 2009); detectores ultrassônicos que identificam bicarbonato de sódio e formalina

(MOHANAN, *et al.*, 2002); biosensores potenciométricos, algoritmos combinados com espectrômetro Raman (para detectar informações químicas e de estrutura de materiais) para detectar ureia (TRIVEDI, *et al.*, 2009; KHAN, *et al.*, 2015); sensores amperométricos para identificar conteúdo de lactose (CONZUELO, *et al.*, 2010); curva de crescimento de capacitância que verifica o desenvolvimento de micro-organismos (FELICE, *et al.*, 1999); cromatografia líquida de alta eficiência para identificação de peróxido de hidrogênio (IVANOVA, *et al.*, 2019); sonda de impedância para detectar bactérias (VAN DER WESTHUYZEN, 2006), além de sensores piezelétricos para contagem das mesmas (CHEN e REN, 2011).

1.1 Justificativa e Motivação

A proposta de Santos Junior (2019), empregando a montagem de um dispositvo com dois diafragmas piezelétricos de baixo custo, que utiliza o princípio de propagação de ondas acústicas em meio material e processamento de sinal pela técnica cromática, foi capaz de obter resultados rápidos na identificação de pequenas concentrações de três adulterantes utilizados nas fraudes em leite: bicarbonato de sódio, ureia agrícola e peróxido de hidrogênio. Com 0,5 e 1% de contaminação, quando comparadas a amostras de leite puro, apresentaram características específicas, tornando eficiente a avaliação qualitativa.

Além disso, a manipulação é simples e dispensa o envio de amostras a laboratórios. O trabalho em si se revela promissor, podendo posteriormente, abranger outras aplicações. O presente trabalho faz uso de todos os dados obtidos provenientes dos ensaios da técnica desenvolvida por Santos Junior (2019), para fazer uma complementação, identificando as amostras de maneira automática e simplificada.

1.2 Objetivos

Este trabalho pretende identificar cada amostra de leite adulterado e puro, utilizando sistemas inteligentes baseados em redes neurais artificiais (RNAs), comparando oito ferramentas disponíveis no *software* para definir qual(is) é(são) mais indicada(s) para esta aplicação.

1.3 Organização do trabalho

A organização do trabalho é feita por meio de 5 capítulos e 1 apêndice:

Capítulo 1: Introdução sobre a importância do leite, principais adulterações e prejuízos causados, apresentação de algumas técnicas de identificação em desenvolvimento no cenário atual, bem como a motivação e objetivos almejados no presente trabalho;

Capítulo 2: São feitas breves explicações sobre diafragmas piezelétricos, emissão acústica, técnica cromática, e uma apresentação do trabalho de Santos Junior (2019), que utilizou estes estudos em que se originarm os dados que foram usados para identificação das RNAs;

Capítulo 3: São apresentados os conceitos de RNAs, neurônios, funções de ativação, taxa de propagação, estruturas de redes, algoritmos de treinamento e detalhes sobre oito ferramentas utilizadas para identificação de amostras de leite puro e adulterado: redes RNPMC, RNRC, RNER, RNRP, RNP, RNRG, RNBRN e RNBRE;

Capítulo 4: Descrição da preparação dos dados, as estruturas e parametrizações das RNAs e metodologias usadas nos ensaios realizados;

Capítulo 5: Apresentação das parametrizações das RNAs e respectivos resultados obtidos individualmente, além de um resumo comparativo;

Ao final, apresenta-se a conclusão, sugestões para trabalhos futuros e referências bibliográficas.

Apêndice A: Os *scripts* utilizados nas execuções das RNAs são demonstrados.

Capítulo 2 – Revisão Bibliográfica

Os conceitos de diafragmas transdutores piezelétricos, emissão acústica e técnica cromática são apresentados neste capítulo, para entendimento do estudo decorrente na validação das amostras contaminadas em relação ao leite puro. No item final, o estudo que utilizou essas técnicas é demonstrado de forma resumida.

2.1 Transdutores piezelétricos

Os transdutores piezelétricos são os dispositivos de materiais inteligentes mais populares, que possuem, em sua construção, elementos como o titanato de zirconato de chumbo, empregado diretamente, em estruturas compostas ou empilhadas com eletrodos, para formar piezocerâmicas – as composições mais utilizadas (CHOPRA, 2002).

Quando submetidos a contrações ou expansões mecânicas, os transdutores piezelétricos convertem vibrações em eletricidade, podendo ser empregados em importantes aplicações, como dispositivo de alarme sísmico, em caso de terremoto (KIMURA, 1998). Outra forma de aproveitar os transdutores nesta configuração, é para geração de energia elétrica, em conjunto com pequenos equipamentos que requerem bateria ou como fonte de energia portátil permanente (KIM, H. *et al.*, 2011). Desta forma, os transdutores piezelétricos são utilizados como sensores.

Por outro lado, os transdutores piezelétricos podem funcionar como atuadores, quando são submetidos a tensões elétricas. Neste caso, produzem vibrações mecânicas, com precisão de deslocamento, grande velocidade de resposta e alta eficiência de energia. São aplicados, por exemplo, em estabilização de câmeras de vídeo (YOICHI, 2006).

2.2 Emissão acústica

O fenômeno físico da emissão acústica acontece intrinsicamente nos materiais durante processos dinâmicos, como quando uma tensão elétrica é aplicada a um elemento. O resultado depende basicamente da intensidade do estresse provocado e do material. Metais, rochas, gelo, madeira, solos, cerâmica e concreto são materiais que podem sofrer ação da

propagação de emissões acústicas e, consequentemente, terem suas integridades abaladas (LORD, 1975; NAZARCHUK, 2017).

Assim como em materiais sólidos, gases e líquidos também servem de meios transmissores para emissões acústicas. Em uma técnica de transmissão ultrassônica, uma amostra de leite é submetida à propagação de ondas acústicas por um transdutor de emissão e detectado por um transdutor de recepção, ambos conectados a um osciloscópio. Um computador pode ser utilizado para processar, monitorar, calcular parâmetros e executar algoritmos para fazer avaliação de qualidade microbiológica, caracterização de partículas coloidais e medições reológicas (MOHAMMADI, *et al.*, 2014).

2.3 Técnica Cromática

Um dos objetivos do processamento de sinais é a separação de parâmetros para extração de informações (WEISS, 1994). No caso de um sistema de monitoramento, os dados amostrados muitas vezes necessitam de identificação, classificação e caracterização, pois seus sinais originais são complexos (WEISS, 1994; ZHANG, *et al.*, 2005).

A técnica cromática (TC) é um método de abordagem capaz de simplificar um sinal de estrutura complexa em três parâmetros, a partir de filtros de cor RGB em paralelo, resultando-se nas Equações (1-3), em que os componentes de frequência de um sinal variam com o tempo através desses filtros (JONES, *et al.*, 2000; PAPOULIS e PILLAI, 2002; ZHANG, *et al.*, 2005).

$$E_b = \frac{1}{2\pi} \sum_{i=1}^{k} F_i^2 \tag{1}$$

em que E_b é a energia, e F_i a transformada *Fourier* da série temporal dos sinais gerais de *f(t)*;

$$\omega_c = \frac{\sum_{i=1}^{k} \omega_i F_i^2}{2\pi E_b} \tag{2}$$

sendo ω_c a banda média e ω_i a frequência angular do sinal *f[n]* no tempo discreto;

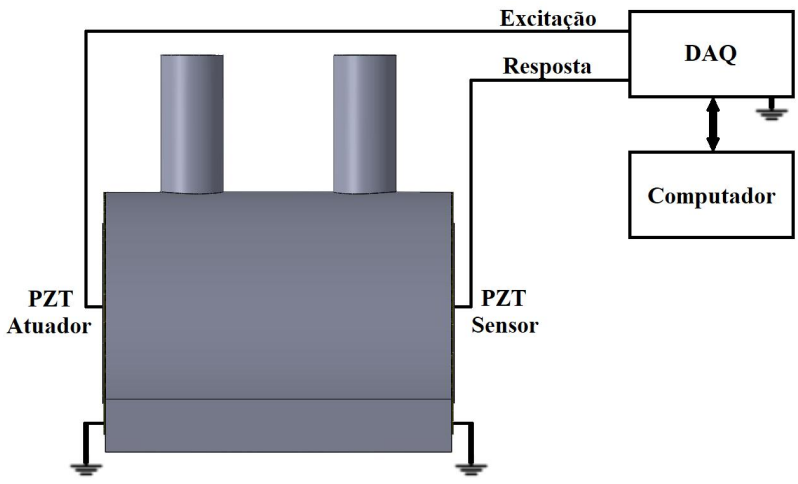
$$B = \sqrt{\frac{1}{E_b} \sum_{i=1}^{k} (\omega_i - \omega_c)^2 F_i^2} \tag{3}$$

e sendo que *B* é a largura de banda equivalente.

2.4 Identificação de adulteração de leite bovino pelo método de propagação de onda em meio material utilizando diafragmas piezelétricos

No ascendente contexto das adulterações do leite em sua forma natural, o trabalho de Santos Junior (2019) se dispôs da utilização de diafragmas piezelétricos de baixo custo na montagem de um recipiente produzido em impressora 3D em que se foram inseridas amostras de leite puro e adulterado para identificação das mesmas. Foram usados dois diafragmas circulares com as mesmas características: discos cerâmicos de 25,0 mm x 0,023 mm com placa de latão medindo 35,0 mm x 0,30 mm (MURATA, 2021). Estes, por sua vez, foram adaptados às extremidades de um tubo de mesmo diâmetro de forma a ficarem um de fronte ao outro. Perpendicularmente, no corpo desse tubo existem ainda dois orifícios menores com canudos por onde foram inseridas as amostras de leite, servindo também para estabilizar bolhas. Um dos diafragmas piezelétricos foi configurado como atuador/emissor, sendo o outro, o sensor/receptor. Os mesmos foram ligados a um dispositivo aquisitor de dados (DAQ) que excita o diafragma emissor com um sinal *chirp* de 1 V de amplitude, frequência na faixa de 0 a 65 kHz, com passo de 2 Hz e 1,2 segundos de duração. Essa excitação provocou emissões acústicas que se propagaram em meio ao leite, chegando no diafragma receptor, do outro lado. O DAQ, conectado a um computador via porta USB, envia estes sinais para serem modelados, por conta de suas complexidades, pelas equações da TC, via *software* MATLAB (MATLAB, 2010). Na Figura 1, tem-se a configuração utilizada (SANTOS JUNIOR, 2019).

Figura 1: Recipiente montado com diafragmas conectado ao DAQ, ligado a um computador.



Fonte: (SANTOS JUNIOR, 2019).

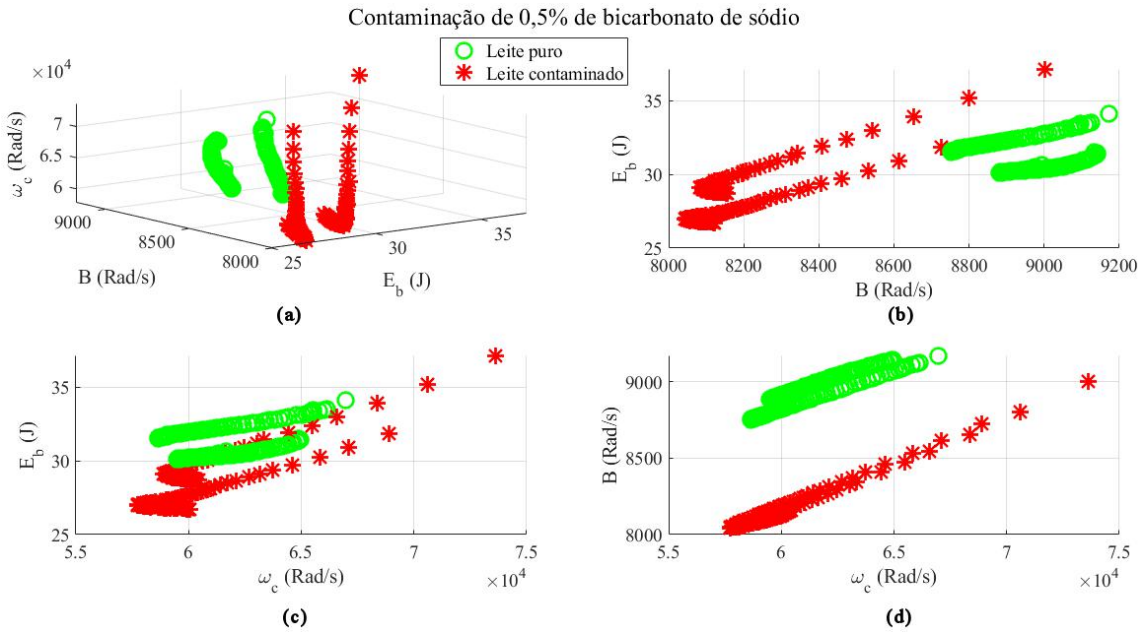
A Técnica Cromática extraiu três parâmetros nesta experimentação: energia (E_b), banda média (ω_c) e largura de banda equivalente (B). Cada ensaio realizado e processado através da TC gerou esses parâmetros, que foram apresentados em mapas tridimensionais utilizando cada parâmetro como eixo (SANTOS JUNIOR, 2019).

O trabalho de Santos Junior (2019) focou nos três principais contaminantes utilizados para adulterações, em percentuais de 0,5% e 1%, para validar a sensibilidade dos transdutores piezelétricos: bicarbonato de sódio, ureia fertilizante e peróxido de hidrogênio (água oxigenada). Os adulterantes foram misturados ao leite integral do tipo “A” em sua forma mais pura, sem conservantes ou neutralizantes. Foram realizadas aquisições de dois blocos de sinais com 100 pontos de respostas, processados pela TC, para cada amostra, resultando em 14 grupos de 100 sinais ensaiados. Essa duplicidade foi feita para se buscar repetibilidade com execuções diferentes para cada amostra. Todos os ensaios foram realizados em ambiente com temperatura devidamente controlada e estável de 25 ºC.

Para demonstrar as diferenças entre os sinais das amostras, foram efetuadas comparações dos ensaios de cada contaminante (com seu respectivo percentual) em relação ao leite puro, considerando seus dois blocos de ensaios correspondentes (SANTOS JUNIOR, 2019).

Na Figura 2 está retratado o mapa de agrupamento dos ensaios de leite puro e leite adulterado com 0,5% de bicarbonato de sódio, destacados em cores e em quatro situações: (a), em três dimensões, com a disposição dos sinais para todos os parâmetros da TC; (b) estão relacionados os parâmetros de energia (E_b) com largura de banda equivalente (B); (c) energia (E_b) com relação à banda média (ω_C); e finalmente em (d), a correlação da banda equivalente (B) com a banda média (ω_C). Para todos os contaminantes foram obtidos mapas correspondentes (SANTOS JUNIOR, 2019). Nos inícios dos ensaios, os pontos foram agrupados de forma mais concentrada em relação aos finais.

Figura 2: Mapas comparativos de amostra de leite contaminado por 0,5% de bicarbonato de sódio com amostra de leite puro, processados pela Técnica Cromática.



Fonte: Adaptado de (SANTOS JUNIOR, 2019).

Por estas classificações se pôde concluir que cada amostra de leite puro e com suas devidas contaminações possui uma característica própria, como uma “impressão digital” do tipo do líquido.

Capítulo 3 – Redes Neurais Artificiais

Este capítulo apresenta conceitos sobre RNAs, funções de ativação, categorização das principais redes, comparação de algoritmos de treinamento e descrição das oito ferramentas empregadas no desenvolvimento deste trabalho.

3.1 Introdução

As buscas por soluções de problemas complexos levam pesquisadores a se inspirarem em fontes da própria natureza para criar novos algoritmos metaheurísticos, baseados em inteligência de enxame, sistemas biológicos, físicos e químicos. Sistemas baseados em situações bem sucedidas da natureza inspiram otimizações verificadas no comportamento coletivo de formigas, abelhas e pássaros, derivando a metodologias para obtenção de resultados melhores pelos menores caminhos possíveis (FISTER, *et al.*, 2013). Nesse contexto, os sistemas de Inteligência Artificial (IA) relacionam-se às capacidades de discernimento demonstradas pelas máquinas para realizar funções que envolvam aprendizagem e solucionamento de problemas, inclusive com criatividade – parte do comportamento exclusivo "cognitivo" dos seres humanos (SAGE, 1990).

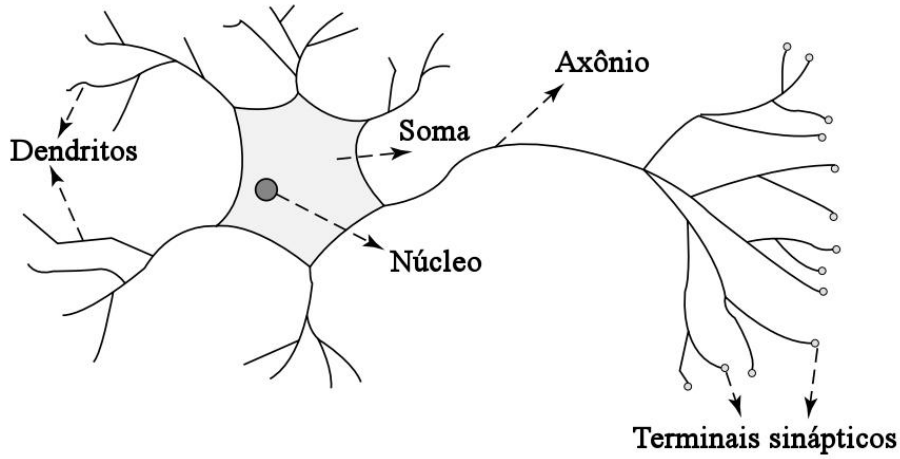
Historicamente, as redes neurais artificiais tiveram início com o desenvolvimento da neurocomputação, correlacionando os neurônios biológicos humanos com cálculos matemáticos. Foi entendido que havia uma lógica entre as atividades nervosas e os eventos neurais, e que existe uma rede estruturada trabalhando os impulsos nervosos (MCCULLOCH e PITTS, 1943).

As RNAs possuem conjuntos de entradas que produzem direta ou indiretamente, por meio de relaxamentos, um conjunto de saídas desejadas, conforme as arquiteturas de suas conexões internas, manipuladas através de unidades básicas com pesos ajustados para se efetuar os processamentos dos dados (KRÖSE, *et al.*, 1993). Semelhantes ao cérebro humano, as RNAs adquirem conhecimento por meio de processo de aprendizagem, possuindo unidades básicas conhecidas como neurônios. Os neurônios são conectados por pesos sinápticos, forças utilizadas para armazenar o conhecimento (HAYKIN, 2001).

3.2 O neurônio

Os mais de 10 milhões de neurônios interconectados no cérebro humano permitem receber, processar e transmitir informações através de reações bioquímicas, resultando em tarefas bem-sucedidas, como reconhecimento de rostos, fala e movimentação corporal das pessoas. Na Figura 3 ilustra-se um neurônio humano, que no interior de seu corpo celular (ou soma) possui um núcleo, responsável por funções lógicas, e externamente extende-se ao axônio, fibra fina e comprida com terminais sinápticos, bem como os dendritos, mais curtos, parecidos com galhos de árvores, formando uma rede. Os axônios de um neurônio se conectam aos terminais sinápticos de outro neurônio, transmitindo sinais complexos quimicamente liberados pelo efeito de aumentar ou diminuir o potencial elétrico dentro do corpo de uma célula. Um pulso é finalmente enviado por um axônio quando o potencial atinge um limiar (ABRAHAM, 2005).

Figura 3: Representação de um neurônio biológico humano.



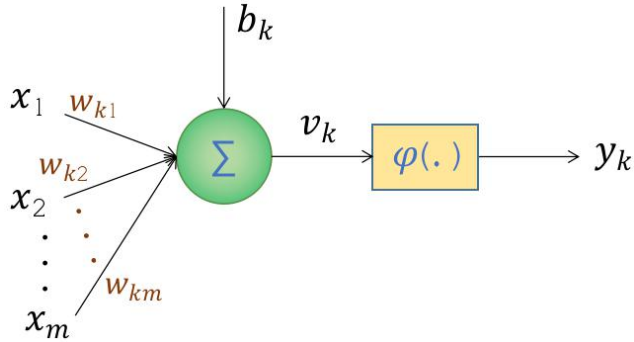
Fonte: Traduzido de (ABRAHAM, 2005).

Os neurônios das RNAs baseiam-se nos neurônios humanos mediante generalizações de sistemas matemáticos, em que os efeitos das sinapses são estabelecidos como pesos que modulam os sinais de entradas. Uma função de ativação transforma a soma ponderada dos sinais de entrada e resultam no impulso do neurônio. De acordo com o algoritmo utilizado na RNA se estabelece a capacidade de aprendizado de um neurônio artificial (ABRAHAM, 2005).

Na Figura 4 está apresentado o modelo generalizado de um neurônio artificial, base

para as RNAs. As entradas são representadas por x_j; w_{kj} são os pesos sinápticos; o viés (*bias)* b_k é encarregado de aumentar ou diminuir a entrada da função de ativação; $\varphi(.)$ é a função de ativação (transferência); u_k é a saída do combinador linear, v_k é o potencial de ativação e y_k é o sinal de saída do neurônio *k* (HAYKIN, 2001).

Figura 4: Aspecto típico de um neurônio artificial.



Fonte: O autor.

Em termos algébricos, essas variáveis são apresentadas nas Equações (4), (5) e (6):

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{4}$$

$$y_k = \varphi(u_k + b_k) \tag{5}$$

$$v_k = u_k + b_k \tag{6}$$

3.3 Funções de ativação

A função de ativação (ou função de transferência) é usada na RNA para transformar um sinal de entrada em um sinal de saída, que, por sua vez, pode alimentar outro sinal de entrada. Se uma RNA não utiliza função de ativação, o sinal de saída se torna uma função linear simples, sendo incapaz de aprender e tampouco executar tarefas mais complexas (SHARMA, 2020).

O aperfeiçoamento das funções de ativação tornou possível que elas passassem de uma simples função binária, em que os resultados eram apenas sinais baixos ou altos, para funções como as descritas nas Equações (7-10). A função de limiar (ou degrau, ou passo) binário é mais simples, utilizada para classificadores binários, dada pela Equação (7). A função de

ativação linear por partes é caracterizada em forma de rampa, com suas condições na Equação (8). A função sigmóide, representada na Equação (9), muito utilizada, torna a curva entre 0 e 1 mais suave (HAYKIN, 2001).

$$\varphi(v) = \begin{cases} 1, se\ v \geq 0 \\ 0, se\ v < 0 \end{cases} \tag{7}$$

$$\varphi(v) = \begin{cases} 1, se\ v \geq +\frac{1}{2} \\ v, +\frac{1}{2} > v > -\frac{1}{2} \\ 0, se\ v \leq -\frac{1}{2} \end{cases} \tag{8}$$

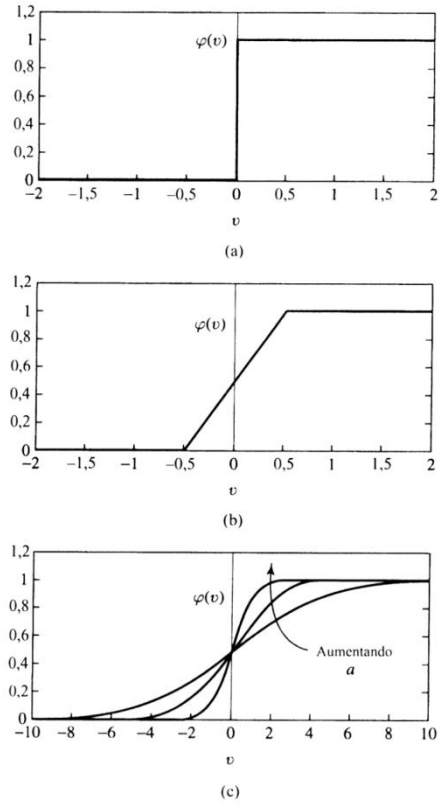
$$\varphi(v) = \frac{1}{1 + e^{-av}} \tag{9}$$

A função tangente hiperbólica é similar à sigmóide, com a diferença de que pode assumir valores entre -1 e 1 (ao invés de 0 a 1), resultando em diferentes sinais de saídas de camadas anteriores que podem alimentar as próximas camadas da rede. Está representada na Expressão (10).

$$\varphi(v) = tanh(v) \tag{10}$$

Na Figura 5 estão representadas as curvas das funções de ativação para as Equações (7-9).

Figura 5: Função de limiar (a), função linear por partes (b) e função sigmóide (c).



Fonte: (HAYKIN, 2001).

Na função de base radial, normalmente utiliza-se a curva de Gauss, em que um valor central de largura σ, responsável pelo controle da suavidade da interpolação, é tomado como referência, e sendo $|x_j - v_j|$ a distância euclidiana (entre dois pontos), onde x_j é o vetor de entrada e v_j, o centro da curva. A equação dessa função Gaussiana é apresentada na Equação (11) (HUSH e HORN, 1993).

A função de ativação competitiva (*compet*) transforma a saída de uma camada anterior em matriz binária que associa o valor 1 ao vetor correspondente e 0 no restante da linha, calculando as probabilidades de cada classe (PAN, *et al.*, 2011).

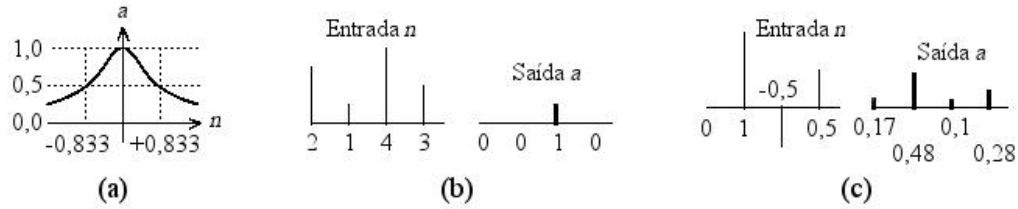
$$\varphi(x_j) = exp\left(-\frac{//x_j - v_j//^2}{2\sigma_j^2}\right) \tag{11}$$

$$Pr(m - \acute{e}sima\ categ.) = \frac{e^{W_m.A}}{\sum_{L=1}^{N} e^{W_L.A}} \tag{12}$$

A função *softmax* é uma combinação de múltiplas sigmóides, capaz de lidar com mais de dois tipos de classes, transformando a saída delas para valores entre 0 e 1 e dividindo pela soma das saídas. A saída do m-ésimo neurônio na camada da função *softmax* da teoria probabilística está demonstrada na Equação (12), sendo que: *N é* a quantidade de classes, *W* e *A* são vetores com j-ésimo elementos (YUAN, 2016).

Na Figura 6 podem ser verificados exemplos de respostas das funções de ativação de base radial, competitiva e *softmax*.

Figura 6: Função de base radial (a), função competitiva (b) e função *softmax* (c).



Fonte: O autor.

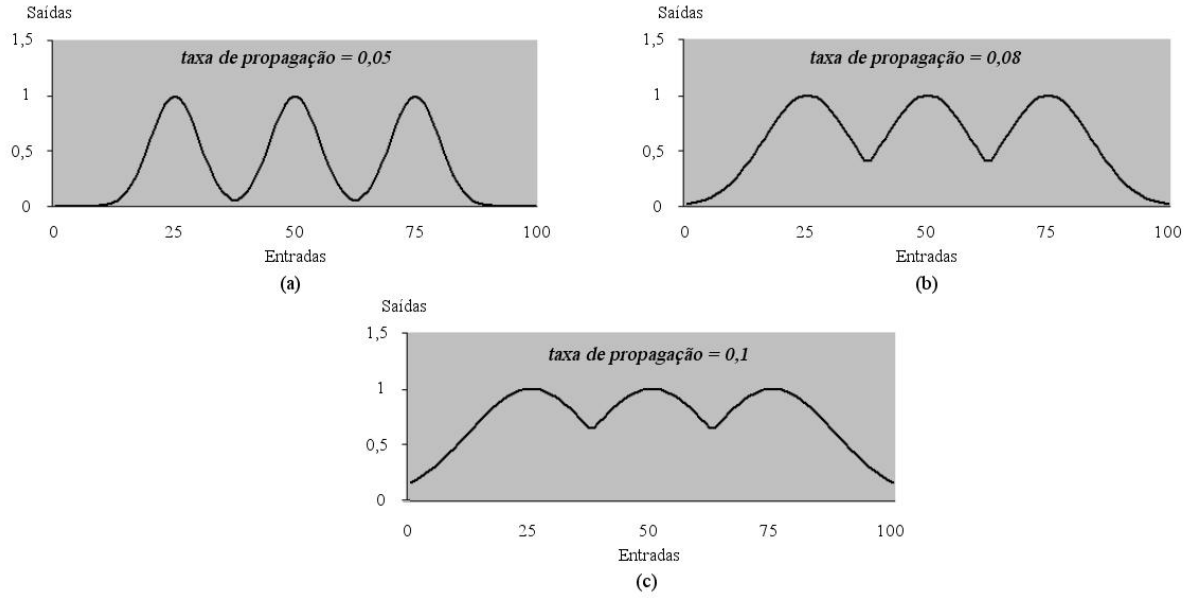
Outras funções de ativação foram criadas a partir de combinações das anteriores: ReLU (Unidade Linear Retificada), ReLU *leaky* e ReLU parametrizada, derivadas da função linear; unidade linear exponencial, variante da ReLU com exponencial e *swish*, uma função sigmóide avançada (SHARMA, 2020).

3.4 Taxa de propagação

Redes neurais com função de base radial, como a Probabilística, a de Regressão Generalizada, e as de Base Radial e Base Radial Exata, possuem como parâmetro principal alterável, a taxa de propagação (*spread*, fator de alisamento, ou fator de suavidade), que pode aumentar ou diminuir a diferença das saídas dessas redes, pois define a área de vizinhança, variando o número de padrões a serem levados em consideração. Na Figura 7 está apresentada uma exemplificação em que é possível se observar entradas de valores numéricos quaisquer (25, 50 e 75), e faixa de saída desejada de 0 a 1. Neste caso específico, utilizando a taxa de propagação de 0,05, os valores de entrada atingem saídas iguais a 1 e as outras entradas têm saídas próximas a 0 (7-a). Aumentando a taxa de propagação para 0,08 (7-b), nota-se que as saídas intermediárias aos valores de entrada estão entre 0,5 e 1. Dobrando-se o

valor inicial da taxa de propagação para 0,1, as saídas resultam em valores mais próximos de 1 (7-c) (FERREIRA, 2004).

Figura 7: Diferenças nas saídas para taxa de propagação 0,05 (a); 0,08 (b) e 0,1 (c).



Fonte: Adaptado de (FERREIRA, 2004).

3.5 Estruturação de redes neurais

O algoritmo perceptron, capaz de fazer a classificação linear de padrões por meio de aprendizagem, é a RNA em sua forma mais simples – tal qual representado na Figura 4 –, possuindo entradas ponderadas por pesos sinápticos, somadas no combinador linear e transferidas à(s) saída(s) (ROSENBLATT, 1958). Com as associações paralelas de neurônios interconectados, foi possível se efetuar processamentos mais complexos, formando arquiteturas de redes progressivas (*feedforward*) ou recorrentes (*feedback*). Para retratar essas RNAs, são usados gráficos direcionados ponderados, nos quais os neurônios são interligados por setas direcionadas, com seus pesos sinápticos correspondentes, às entradas e saídas (JAIN, *et al.*, 1996).

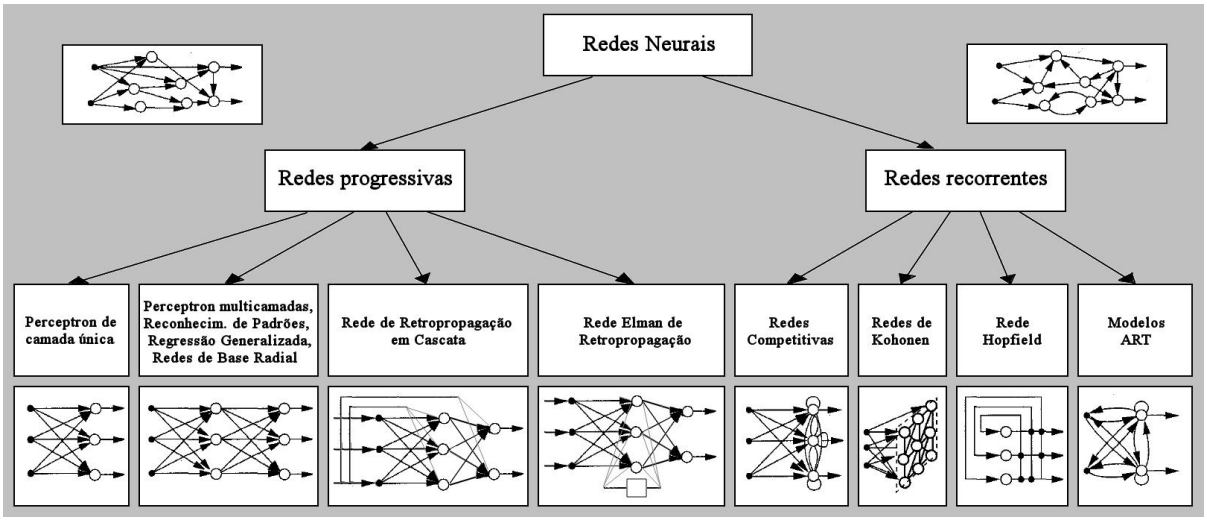
As redes progressivas têm fluxo de sinal em apenas uma direção, são estáticas e podem possuir uma ou múltiplas camadas de neurônios (também conhecidas como camadas ocultas, escondidas ou intermediárias). As redes com múltiplas camadas são as mais comuns, e também conhecidas como perceptron multicamada. Há ainda as redes de função de base radial, possuindo em sua camada principal a função de ativação radial (JAIN, *et al.*, 1996).

Já as redes recorrentes são sistemas dinâmicos e possuem ao menos um laço de realimentação, dependendo de sua configuração. A realimentação consiste em conectar a saída de um neurônio em sua própria entrada. Essas redes podem conter uma ou múltiplas camadas de neurônios (HAYKIN, 2001).

Por meio dessas estruturações avançadas, as aplicações das RNAs progrediram da classificação de padrões para atingir resultados mais complexos, como aproximação de funções, memórias associativas, otimização e estimação.

Na Figura 8 são apresentadas derivações típicas das categorias de redes neurais artificiais.

Figura 8: Arquiteturas de RNAs e respectivas categorias.



Fonte: Adaptado de (JAIN, *et al.*, 1996).

3.6 Algoritmos de treinamento

Uma característica importante das RNAs é sua capacidade de aprendizagem. Basicamente, os dados são apresentados na forma de entradas, é feito o treinamento pelo algoritmo que busca ajustes ideais para os pesos sinápticos e *bias*, em que posteriormente são feitas comparações com novos exemplos e, finalmente apresentados nas formas de saídas.

Para se fazer a otimização não linear dos pesos das redes progressivas, são aplicados métodos de retropropagação modificada, retropropagação com aproximação de gradiente conjugado, retropropagação com base em aprendizagem de conceito, gradiente conjugado escalado e sua contraparte estocástica (de variável aleatória), e o algoritmo de Marquardt, que

é extremamente rápido para redes pequenas, com algumas centenas de parâmetros (ILONEN, *et al.*, 2003).

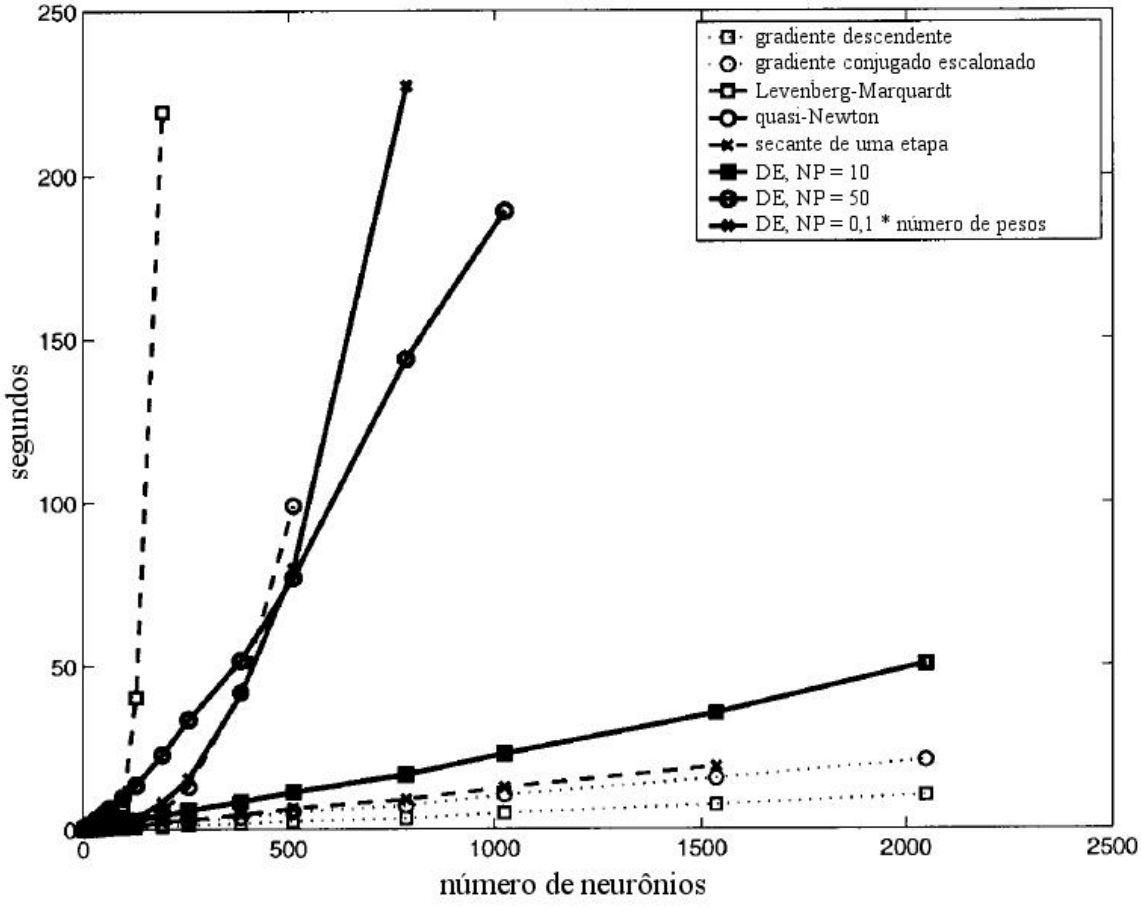
Desenvolvido por Kenneth Levenberg e Donald Marquardt, o algoritmo Levenberg-Marquardt se destaca por sua flexibilidade para implementações e possui convergência estável. É fortemente recomendável para treinamento de RNAs de forma geral, exceto para redes muito complexas, como as de reconhecimento de imagem. Em comparação com o algoritmo de retropropagação de erro original, o algoritmo de Levenberg-Marquardt se mostra mais rápido e com eficiência de 10 a 1000 vezes superior. Em relação ao algoritmo de Gauss-Newton, o algoritmo de Levenberg-Marquardt é mais estável, além de eficiente (YU e WILAMOWSKI, 2011). Os ajustes dos pesos sinápticos do algoritmo de Levenberg-Marquardt podem ser verificados na Equação (13), sendo *J* a matriz jacobiana dos vetores de erros *e(n)* da RNA em relação aos pesos com *n* linhas. A variável μ é inversamente proporcinal à função custo em interações sucessivas na implementação computacional.

$$w(n+1) = w(n) - (J^T(n)J(n) + \mu I)^{-1} J^T(n)e(n) \tag{13}$$

Fazendo uma comparação de complexidade computacional e tempo, uma rede estruturada com 1000 pontos de amostra da função *sen(x)* foi submetida a seis algoritmos de treinamento em um ciclo de execução.

Na Figura 9 estão exibidos os resultados desta experimentação, em que se é possível verificar que os métodos gradientes, conhecidos como gradiente descendente, gradiente conjugado escalonado, Levenberg-Marquardt e secante de uma etapa, são mais rápidos em aplicações que exigem menos neurônios; em contrapartida, os métodos quasi-Newton e de Evolução Diferencial (DE – *Diferencial Evolution*) com três variações nos pesos sinápticos, necessitam de menos neurônios, porém, levam mais tempo para chegarem aos resultados (ILONEN, *et al.*, 2003).

Figura 9: Comparação de algoritmos de treinamento de RNAs por complexidade computacional.

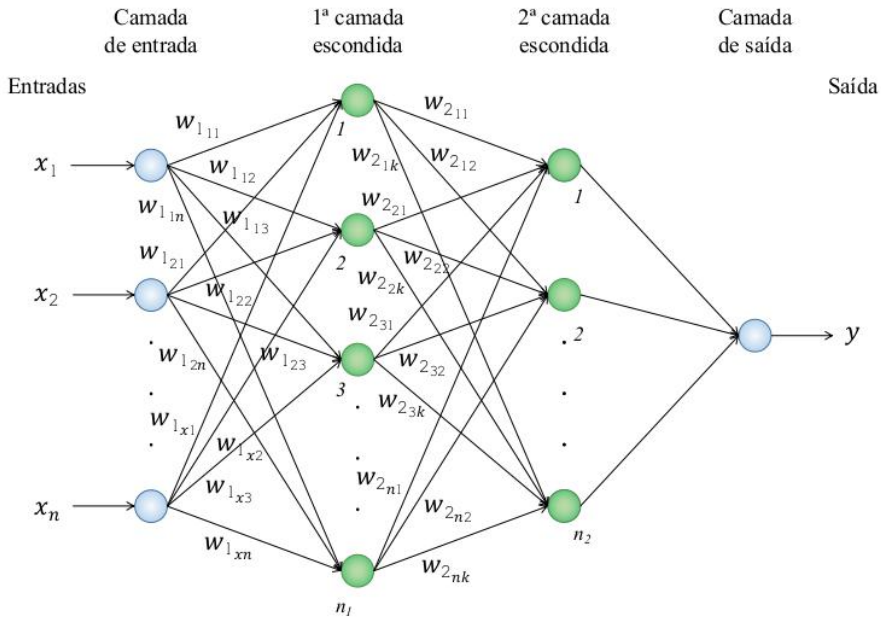


Fonte: Traduzido de (ILONEN, *et al.*, 2003).

3.7 Rede Neural Perceptron Multicamadas

As Redes Neurais Perceptron Multicamadas (RNPMC) são as mais utilizadas e podem possuir mais de uma camada de neurônios, treinadas com retropropagação, que internamente são conhecidas como camadas escondidas ou intermediárias (SVOZIL, *et al.*, 1997). O processamento consiste na apresentação de padrões de entrada à rede, que retorna uma saída. Esta saída é comparada a uma saída desejada e, dependendo da diferença entre elas, o algoritmo de retropropagação efetua ajustes nos pesos sinápticos da saída até a entrada para, novamente haver uma comparação. O ciclo para quando alguma condição, como número de tentativas ou erro mínimo previamente estabelecido é auferido (WERBOS, 1990). Na Figura 10 apresenta-se um exemplo de rede RNPMC com duas camadas intermediárias.

Figura 10: Diagrama de uma rede RNPMC com duas camadas escondidas.



Fonte: O autor.

A regra delta generalizada é o algoritmo padrão de treinamento de uma rede RNPMC, apresentado na Equação (14), em que o gradiente local δ é o responsável pelo apontamento das modificações necessárias nos pesos sinápticos w_{ji}, regulado pela taxa de aprendizado η, que determina o tamanho dessas modificações. O termo de momento α de valores entre 0 e 1 tem a função de garantir estabilidade para as variações de η.

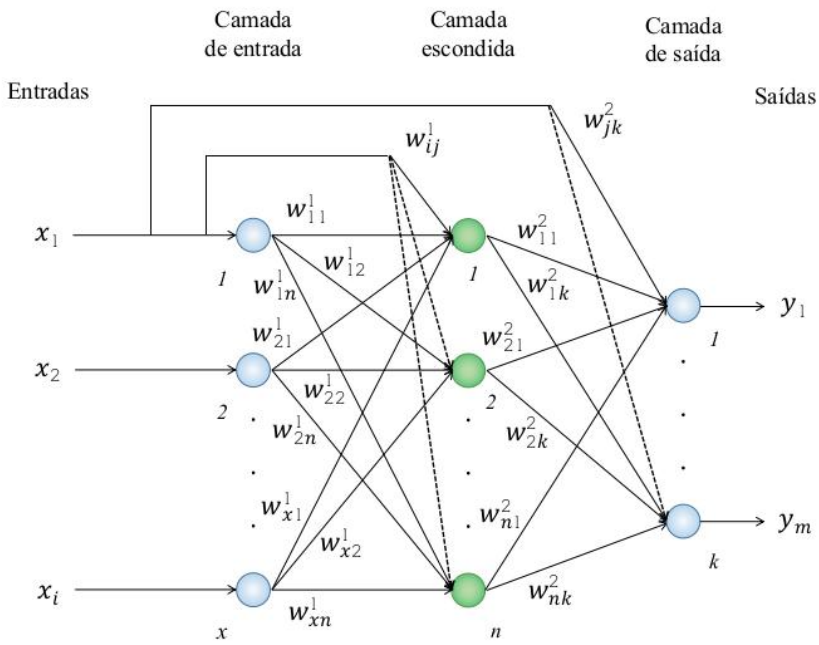
Cada camada da rede pode possuir uma função de ativação diferente, do tipo limiar, linear por partes, sigmóide, tangente hiperbólica ou outra, dependendo de seu projeto (HAYKIN, 2001).

$$\Delta w_{ji}(n+1) = \alpha \Delta w_{ji}(n) + \eta \delta_j y_j \tag{14}$$

3.8 Rede Neural de Retropropagação em Cascata

A RNA de Retropropagação em Cascata (RNRC) é similar à rede RNPMC, utiliza o mesmo algoritmo de aprendizagem, e também pode conter camadas escondidas. O que difere as duas é uma conexão adicional de processamento da entrada às demais camadas, para extrair estatísticas de ordem elevada, sendo assim, capaz de ter uma perspectiva global além da conectividade local de uma camada para a outra. Desta forma, há uma melhoria da velocidade de treinamento. Na Figura 11 está representada uma rede RNRC de três camadas, sendo uma de entrada – interligada às demais pela unidade de conexão adicional –, uma escondida e uma de saída (ALKHASAWNEH, *et al.*, 2014; DE AGUIAR FILHO, 2018).

Figura 11: Arquitetura de uma RNA de Retropropagação em Cascata.



Fonte: O autor.

Na Equação (15) é descrito um modelo simplificado de Rede de Retropropagação em Cascata, sendo que φ^i é a função de ativação da camada de entrada para a camada de saída, e w_i^i representa o peso sináptico da mesma (WARSITO, *et al.*, 2018).

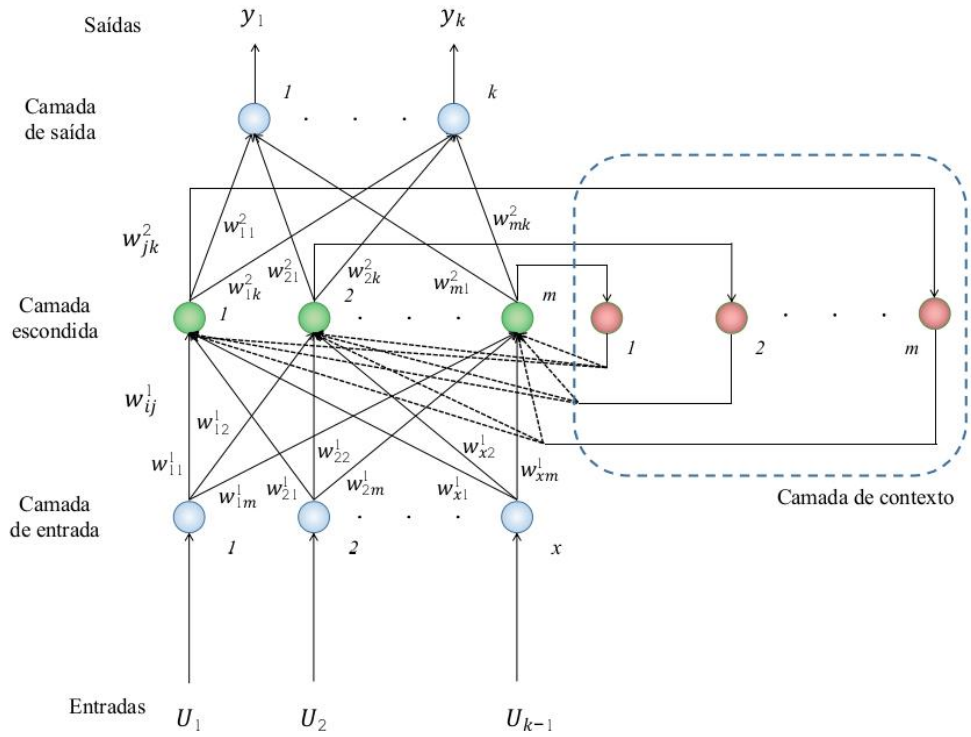
$$y = \sum_{i=1}^{n} \varphi^{i} w_i^i x_i + \varphi^{0} \left(\sum_{j=1}^{k} w_j^{0} \varphi_j^h \left(\sum_{i=1}^{n} w_{ji}^h x_i \right) \right) \quad (15)$$

3.9 Rede Neural Elman de Retropropagação

O princípio de operação da Rede Neural Elman de Retropropagação (RNER) se assemelha às RNPMC e de Retropropagação em Cascata, mas com a adição de uma camada especial de contexto, que realimenta os neurônios das camadas escondidas e de saída, com seus valores salvos previamente, capacitando a RNA a aprender, reconhecer e gerar padrões temporais e espaciais. Esta rede também pode conter mais de uma camada escondida. O algoritmo utilizado para aprendizagem melhora a velocidade de treinamento e evita que a RNA fique presa em um mínimo local, modificando os pesos e limites para suprimir o erro da camada de saída, bem como a diferença para a saída real (RAKHSHANDEHROO, *et al.*, 2012; ALKHASAWNEH, *et al.*, 2014).

O padrão esquemático de uma rede Elman de Retropropagação é exibido na Figura 12.

Figura 12: Modelo de Rede Neural Elman de Retropropagação.



Fonte: O autor.

Nas Equações (16-18) estão apresentados os modelos matemáticos para a RNA, tendo-se que: φ_k e φ_1 representam as funções de ativação das camadas de saída e escondida, respectivamente; k é o momento; w^2_{jk} é o peso da camada escondida para a camada de saída; w_{ij} é o peso da camada escondida para a camada de entrada; w_c é o peso da camada escondida para a camada de contexto; $x(k)$ é a camada escondida de saída; b_2 e b_1 são limiares da camada de saída e da camada escondida, respectivamente; $x_c(k)$ representa a saída da camada de contexto (ALKHASAWNEH, *et al.*, 2014).

$$y_k(k) = \varphi_k(w^2_{jk}x(k) + b_2) \tag{16}$$

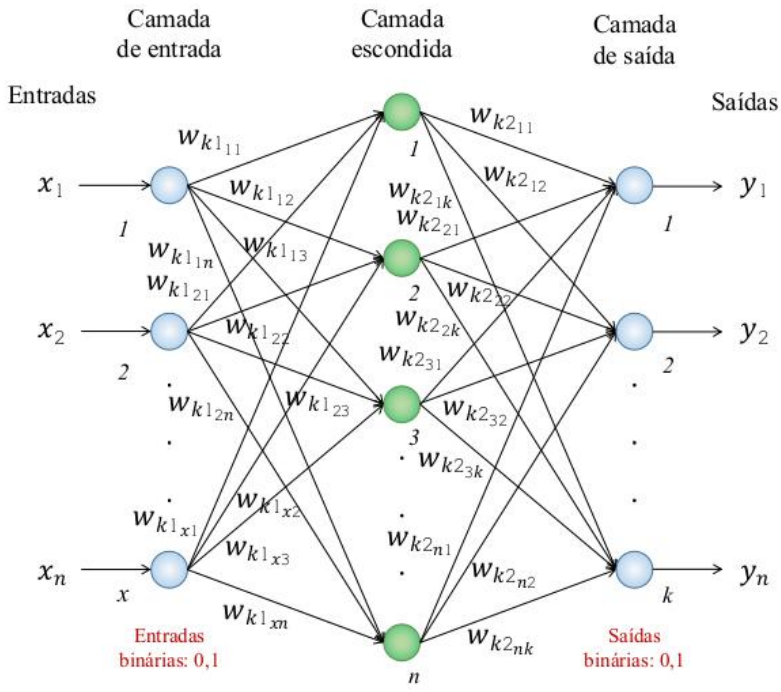
$$x(k) = \varphi_1(w_c x_c(k) + w_{ij}(u(k-1)) + b_1) \tag{17}$$

$$x_c(k) = x(k-1) \tag{18}$$

3.10 Rede Neural de Reconhecimento de Padrões

As Redes Neurais de Reconhecimento de Padrões (RNRP) são baseadas em redes RNPMC, treinadas para classificar entradas de acordo com classes de destino. Estas redes possuem uma camada escondida com função de ativação tangente hiperbólica e em sua saída, uma camada com função de ativação *softmax* (KIM, B. e PARK, 2018). Os dados alvo devem consistir em matrizes de zeros, exceto para a classe a ser representada, com valor 1 (Neural Network Toolbox User's Guide, 2010). Para atingir esta condição binária no MATLAB, é necessário criar uma matriz de zeros, converter as entradas não-binárias para binárias e no final executar processo reverso. A estrutura de uma rede de reconhecimento de padrões está retratada na Figura 13.

Figura 13: Estrutura de uma RNA de reconhecimento de padrões.



Fonte: O autor.

3.11 Rede Neural Probabilística

As RNAs Probabilísticas (RNP) têm capacidade de separar os dados em um número específico de categorias de saída, e classificam os vetores de entrada com base na estratégia de Bayes, método semelhante ao algoritmo de k-vizinhos mais próximos, em que identifica-

se um padrão de teste tomando-se o principal atributo de classe de seus padrões *k* nas amostras mais próximas. Esta rede possui quatro unidades: a primeira unidade aceita os padrões de entrada; a segunda unidade, a de padrões, multiplica o vetor de entrada pelo vetor de pesos, resultando em Z_i, que posteriormente é submetida a uma operação não linear $exp[(Z_i - 1)/\sigma^2]$; a terceira unidade é a de soma, que adiciona as entradas originadas nas correspondentes unidades da camada de padrão de uma determinada classe; as unidades de saída constituem-se de neurônios de duas entradas que produzem saídas binárias com apenas um único peso variável, ou seja, é igual a 1 em apenas uma das unidades e 0 nas demais (SPECHT, 1990).

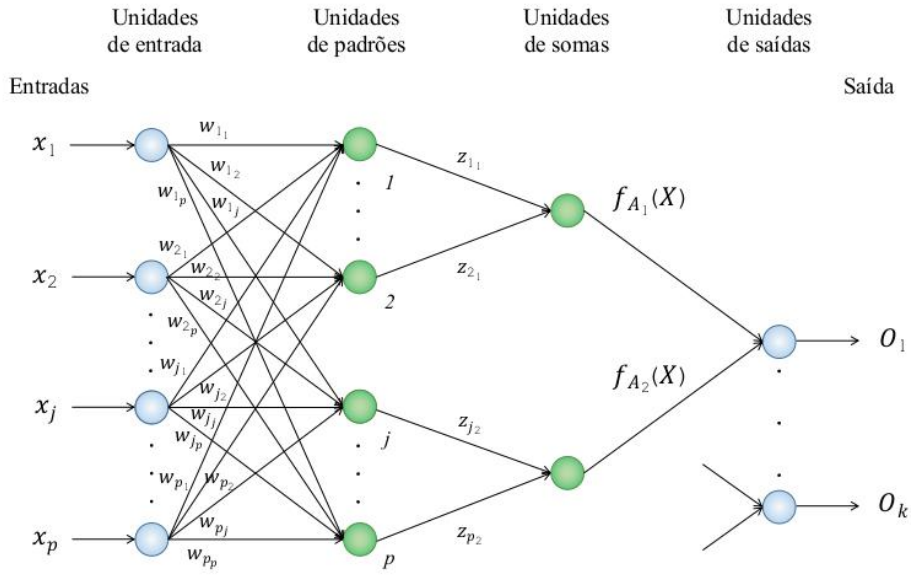
A base matemática de uma rede RNP é apresentada na Equação (19), em que se é utilizada a soma de várias funções Gaussianas esféricas $f_i(x)$ centradas em cada vetor de seu treinamento, dados por x_i – que consiste basicamente em ajustar um único parâmetro, o fator de suavidade σ, ou taxa de propagação (*spread*). Nesta equação, *i* representa o número da classe, *j* é o número do teste padrão, x_{ij} é o j-ésimo vetor treinamento da classe *i*, *x* são os vetores do teste, M_i é o número de vetores no treinamento na classe *i*, e *p* é uma dimensão do vetor *x* (VON ZUBEN e DE CASTRO, 2003). As funções Gaussianas são as funções de ativação que agrupam os padrões de entradas em classes, fazendo com que a RNA calcule a função de distribuição da probabilidade depois para uma classe simples, avaliando o ponto definido pelo padrão de entrada (FERREIRA, 2004).

$$f_i(x) = \frac{1}{(\sqrt{2\pi})^p \sigma^p M_i} \sum_{j=1}^{M_i} exp \frac{-(x - x_{ij})^T (x - x_{ij})}{2\sigma^2} \qquad (19)$$

As redes RNP são de fácil uso e extremamente rápidas para bancos de dados de tamanhos moderados (SPECHT, 1990).

A configuração de uma RNA Probabilística está mostrada na Figura 14.

Figura 14: Esquema típico de uma Rede Neural Probabilística.



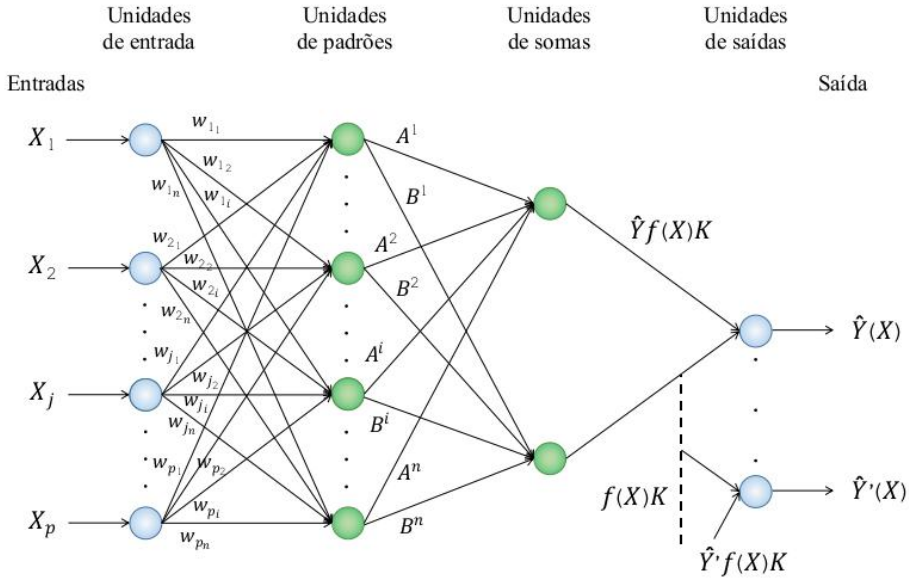
Fonte: O autor.

3.12 Rede Neural de Regressão Generalizada

A rede de Regressão Generalizada (RNRG) prevê o valor de uma variável dependente a partir do valor de variáveis independentes. Essa RNA compara um novo padrão de entrada com um padrão armazenado, gerando uma saída real média com base nas saídas associadas armazenadas (HEIMES e VAN HEUVELN, 1998). O diagrama de uma rede RNRG é mostrado na Figura 15, em que se é possível notar as interconexões dos neurônios operando em sistema paralelo. Padrões de treinamentos são atribuídos para os centros de agrupamentos (*cluster*), atualizando os coeficientes A' e B' para as camadas de soma, sendo estes os únicos pesos que precisam ser adaptáveis e prontamente alterados (SPECHT, 1991). No MATLAB (MATLAB, 2010), as primeiras camadas escondidas de unidades de padrões têm função de ativação de base radial, possuem as mesmas quantidades de neurônios que os vetores de entrada, sendo a taxa de propagação (*spread*) o parâmetro que calcula a distância do vetor de entrada ao vetor de peso de um neurônio, embora a descrição original sugeria função de ativação exponencial, dentre outras (SPECHT, 1991).

As unidades de somas são constituídas de funções lineares especiais que ponderam suas camadas anteriores com a aplicação de pesos (WASSERMAN, 1993).

Figura 15: Diagrama de uma Rede Neural de Regressão Generalizada.



Fonte: O autor.

A estrutura de uma rede RNRG está apresentada nas Equações (20-23), em que $A(k)^i$ e $B(k)^i$ representam o incremento cada vez que uma observação de treinamento Y^j para o *cluster i* é encontrada, sendo $A(k)^i$ a soma dos valores Y e $B(k)^i$ o número das amostras associadas ao *cluster i*; D_i^2 é uma função escalar de ponderação exponencial de acordo com sua distância euclidiana para os padrões de entradas X, e σ é a taxa de propagação que definie a abertura da função de ativação (SPECHT, 1991).

$$D_i^2 = (\hat{X} - X^i)^T(\hat{X} - X^i) \tag{20}$$

$$A^i(k) = A^i(k-1) + Y^j \tag{21}$$

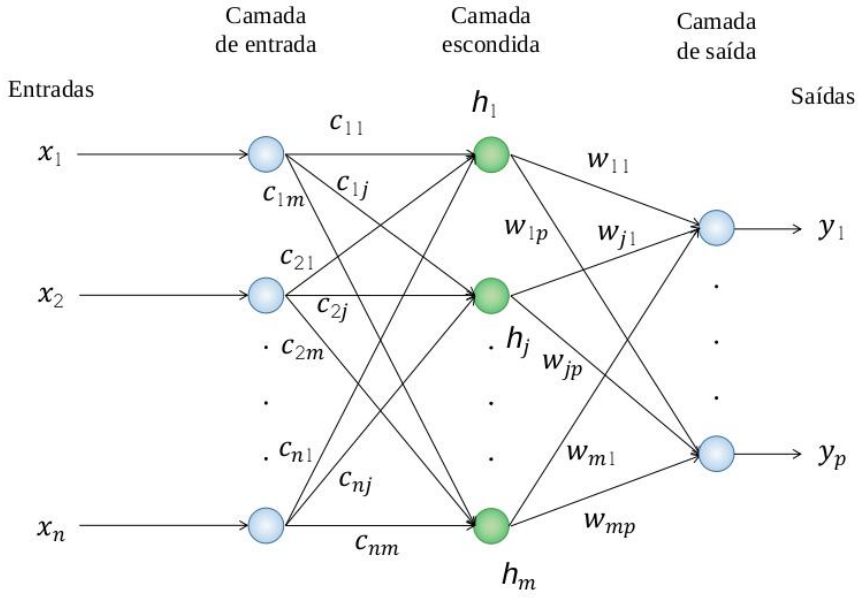
$$B^i(k) = B^i(k-1) + 1 \tag{22}$$

$$\hat{Y}(X) = \frac{\sum_{i=1}^{m} A^i exp\left(\frac{-D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^{m} B^i exp\left(\frac{-D_i^2}{2\sigma^2}\right)} \tag{23}$$

3.13 Rede Neural de Base Radial

As RNAs de Base Radial (RNBR) utilizam hiperelipsóides para agrupar os dados de entrada por aproximação de funções, em espaços de alta dimensionalidade. Estas redes possuem três camadas: a camada de entrada, que não tem pesos sinápticos, é conectada diretamente à única camada escondida, tipicamente de base radial de função Gaussiana com um conjunto de nós para cada centro de função de base radial; e a camada de saída, com função de ativação linear, classifica os padrões recebidos da camada anterior (BROOMHEAD e LOWE, 1988; MOODY e DARKEN, 1989). O diagrama de uma RNBR é apresentado na Figura 16.

Figura 16: Diagrama de uma rede de função de base radial.



Fonte: O autor.

O treinamento das redes de função de base radial se resume em se buscar os centros e raios das unidades radiais e o vetor de pesos entre a camada escondida e a camada de saída. Na Equação (24) está descrita a saída *y* de uma rede RNBR, em que w_j é a matriz de pesos sinápticos da camada de saída, x_n é o vetor de entradas, c_{jn} é o vetor que define o centro da função de base radial e σ_{jn} é o vetor responsável pela taxa de decrescimento da gaussiana junto a cada coordenada do espaço de entrada.

Já a função exponencial é uma norma ponderada da diferença entre o vetor de entrada e o centro de função de base radial (MOODY e DARKEN, 1989).

$$y = \sum_{i=1}^{m} w_j \, exp\left(- \frac{(x_1 - c_{j1})^2}{\sigma_{j1}} - \frac{(x_2 - c_{j2})^2}{\sigma_{j2}} - \ldots - \frac{(x_n - c_{jn})^2}{\sigma_{jn}} \right) \quad (24)$$

No MATLAB, existem duas ferramentas semelhantes para a RNA de função de base radial: RNBR e RNBRE (RNBR Exata). Ambas têm como principal parâmetro alterável, a taxa de propagação (*spread*), que, quanto maior, mais suave torna a aproximação da função. A rede RNBR ainda acrescenta neurônios na camada escondida para que se atinja uma meta de Erro Quadrático Médio, bem como a quantidade dos mesmos para se adicionar entre as exibições – todos estes três parâmetros são ajustáveis, junto à taxa de propagação. Em contrapartida, neste caso, uma taxa de propagação muito pequena exige mais neurônios para se adequar a uma função suave, fazendo com que a generalização não seja suficiente (Neural Network Toolbox User's Guide, 2010).

3.14 Métricas de avaliação de desempenho

Após a execução de uma RNA, é importante efetuar a medição de seu desempenho. Desta forma, é possível identificar a necessidade de alteração de parâmetros e se fazer comparações, inclusive entre outras ferramentas de redes. Dentre os diversos meios de avaliação de RNAs, a taxa de acertos, o Erro Quadrático Médio (EQM) e a regressão são amplamente utilizados (BARROS, 2018).

A taxa de acertos consiste na subtração do valor obtido na saída da simulação da rede, de seu valor desejado, tendo seu percentual calculado, conforme mostrado na Expressão (25), em que se divide a quantidade de acertos pela quantidade total de vetores, e depois se multiplica por 100. O EQM, apresentado na Equação (26), é uma medida de qualidade em que se buscam, idealmente, resultantes próximos a 0, indicando que o erro é mínimo. Assim, n é o número de elementos do conjunto de dados; y_p é o valor desejado, e y_i é o valor obtido na saída da rede. No caso da regressão, os valores variam de 0 (pior) a 1 (melhor), representando a capacidade de convergência da rede, e podendo ser apresentada também de forma gráfica (BARROS, 2018).

$$Percentual_{Acertos} = \left(\frac{n^{\circ}_{Acertos}}{n^{\circ}_{Total}}\right) 100 \tag{25}$$

$$EQM = \frac{1}{n}\sum_{i=1}^{n} (y_p - y_i)^2 \tag{26}$$

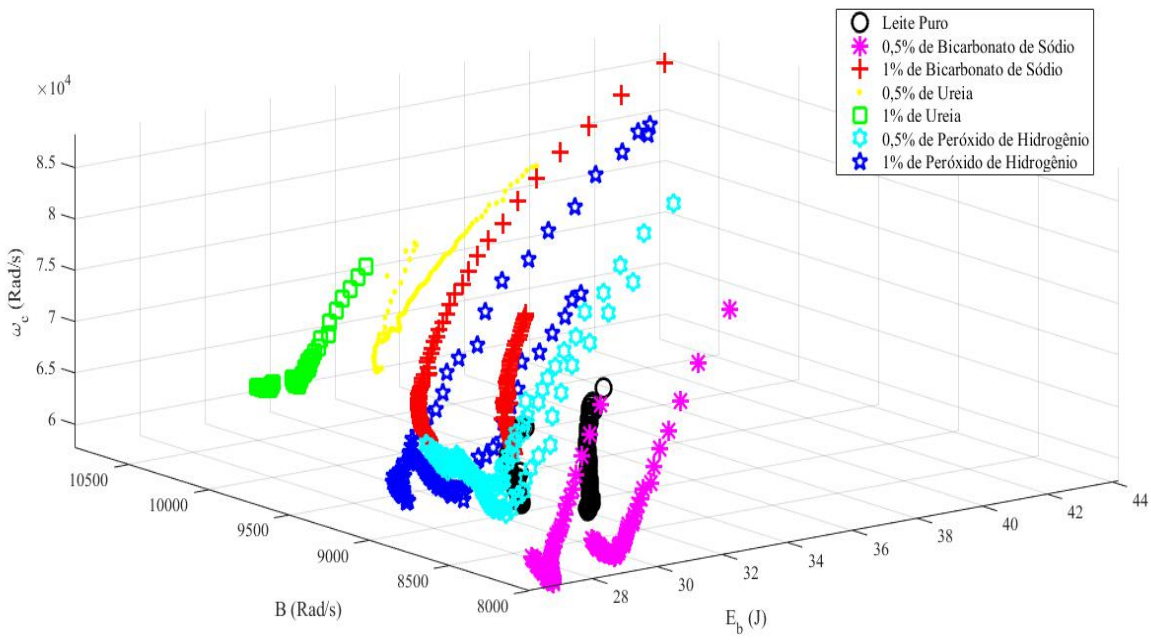
Capítulo 4 – Condicionamento dos Dados e Implementação no MATLAB

Este capítulo faz a descrição de como foram feitas as preparações dos dados utilizados, quais as estruturações e parametrizações das RNAs e especifica como foram realizados os ensaios.

4.1 Introdução

Os conjuntos de amostras de leite puro e adulterado processados com a TC no trabalho de Santos Junior (2019) foram devidamente rotulados durante os ensaios, para a representação em mapas em duas e três dimensões (como mostrado na Figura 2) e validação, através de comparações. Na Figura 17 está apresentado um mapa completo em 3D, cujos eixos correspondem aos parâmetros da TC e todos os pontos dos ensaios estão classificados.

Figura 17: Mapa 3D de classificação das amostras de leite puro e adulterado.



Fonte: Adaptado de (SANTOS JUNIOR, 2019).

4.2 Preparação dos dados

Os dados disponibilizados à partir do trabalho de Santos Junior (2019) fazem parte de em um *script* de MATLAB, programa que possui poderosas ferramentas para realização de complexos cálculos numéricos, matriciais e gráficos (MATLAB, 2010). Todas as amostras foram apresentadas na área de trabalho como variáveis em forma de matrizes de vetores de 100 linhas por 3 colunas, cada qual identificada com suas devidas nomenclaturas correspondentes, da seguinte forma: leite puro; leite adulterado com 0,5% de bicarbonato de sódio; leite adulterado com 1% de bicarbonato de sódio; leite adulterado com 0,5% de ureia; leite adulterado com 1% de ureia; leite adulterado com 0,5% de peróxido de hidrogênio e leite adulterado com 1% de peróxido de hidrogênio, totalizando 7 possibilidades. As 100 linhas se referem aos 100 blocos de ensaios ralizados para cada amostragem de leite puro e adulterado, sendo importante salientar que existem dois conjuntos desses blocos para garantir a repetibilidade da metodologia empregada (SANTOS JUNIOR, 2019).

As colunas das matrizes se referem aos parâmetros processados pela TC, dispostas por energia (E_b) em [J], banda média (ω_c) em [rad/s] e largura de banda equivalente (B) em [rad/s], respectivamente, possuindo classes numéricas diferentes umas das outras. As linhas são os três pontos no espaço de tempo do ensaio.

Os blocos de amostras foram considerados como entradas das RNAs. Para compor as saídas desejadas (“alvos”) e permitir os cálculos algébricos das redes, foram atribuídos valores inteiros de 1 a 7, na sequência:

- leite puro = 1;
- leite adulterado com 0,5% de bicarbonato de sódio = 2;
- leite adulterado com 1% de bicarbonato de sódio = 3;
- leite adulterado com 0,5% de ureia = 4;
- leite adulterado com 1% de ureia = 5;
- leite adulterado com 0,5% de peróxido de hidrogênio = 6;
- leite adulterado com 1% de peróxido de hidrogênio = 7.

Nessa circunstância, foi criada uma quarta coluna para as matrizes dos conjuntos de amostras com esses números repetidos da primeira à ultima linha. Esta quarta coluna foi repetida para os dois conjuntos de amostras redundantes. Assim, tem-se o conjunto de dados, constituído de uma matriz de 1400 linhas por 4 colunas. Uma pequena parte da matriz de dados referente ao leite adulterado com 1% de ureia (amostra 1) é reproduzido na Tabela 1,

em que as três primeiras colunas são os parâmetros de energia, banda média e largura de banda equivalente, respectivamente, utilizados como dados de entrada, e a quarta, o número atribuído à saída desejada.

Tabela 1: Parte da matriz de dados da amostra de leite adulterado com 1% de ureia.

Entradas			**Saída desejada**
Energia [J]	**Banda média [rad/s]**	**Largura de banda equivalente (*B*)**	
25,9845	9728,0756	69747,1722	5
25,9607	9727,2096	69712,6873	5
25,9547	9729,0754	69719,6259	5
25,9534	9727,0542	69692,7329	5
25,9435	9725,4332	69649,9323	5
25,9354	9723,9044	69613,9884	5
25,9419	9720,4632	69562,0362	5
25,9406	9717,5840	69511,6817	5
⋮	⋮	⋮	⋮
26,4330	9687,9792	69442,6744	5
26,4442	9688,8568	69458,8034	5
26,4468	9685,1290	69429,1206	5

Fonte: O autor.

4.3 RNAs utilizadas

A implementação das redes neurais neste trabalho contemplou oito ferramentas de diferentes arquiteturas disponíveis no MATLAB, em que se buscou a verificação de seus desempenhos individuais com relação às identificações dos adulterantes no leite, e se pôde comparar quais foram as melhores opções de forma simples e objetiva.

Foram empregadas as seguintes ferramentas: *feedforwardnet* (Rede Perceptron Multicamadas *Feed-Forward*), uma das RNAs mais utilizadas, tanto para aproximação de funções, como para classificação de padrões, bem como as redes derivadas dela com algumas modificações, mais detalhadas no Capítulo 3 – *newcf* (Rede de Retropropagação em Cascata), *newelm* (Rede Elman de Retropropagação) e *patternnet* (Rede de Reconhecimento de Padrões), esta última, mais focada no reconhecimento de padrões (Neural Network Toolbox User's Guide, 2010; DE AGUIAR FILHO, 2018). Além disso, também foram utilizadas as redes com funções de ativação essencialmente de base radial – aplicadas principalmente para modelagem de sistemas de predição e classificações de dados –, *newrb* (Rede Neural de Base

Radial) *e newrbe* (Rede Neural de Base Radial Exata), e suas variantes: *newpnn* (Rede Neural Probabilística), que tem habilidade de treinar sob conjuntos de dados escassos, e sua generalização, a rede *newgrnn* (Rede Neural de Regressão Generalizada), ambas baseadas em estimação e probabilidade, aprendendo à partir de padrões apresentados à elas (JAIN, *et al.*, 1996; PANCHAL, F. S. e PANCHAL, M., 2014; FURTADO, 2019).

As redes perceptron multicamadas diferem das de função de base radial requerendo um número menor de parâmetros para se obter um mesmo grau de precisão, mas são mais lentas (HAYKIN, 2001).

Foram criados *scripts* individuais para as RNAs, de acordo com suas particularidades referentes às arquiteturas internas, porém, mantendo o mesmo banco de dados construído anteriormente.

4.3.1 Épocas, erro, tempo de execução e algoritmo de treinamento

As seguintes constituições foram adotadas nas redes neurais RNPMC, RNRC, RNER e de Reconhecimento de Padrões: o número de épocas – a quantidade máxima de ciclos em que os pesos são ajustados pelo algoritmo de treinamento – foi estabelecido em 2000, para aumentar as chances das RNAs buscarem o menor Erro Quadrático Médio, mesmo motivo para que o tempo de treinamento fosse definido como “infinito”; EQM de $1e^{-7}$, mantendo o padrão do MATLAB; e algoritmo de treinamento Levenberg-Marquardt, devido sua flexibilidade, estabilidade e rapidez para esta aplicação (YU e WILAMOWSKI, 2011). Para as redes neurais com funções de bases radiais, estas configurações específicas não são utilizadas, exceto para a RNBR, em que o EQM é determinado também como $1e^{-7}$, para comparação equivalente entre RNAs.

4.3.2 Funções de ativação

As funções de ativação foram mantidas no padrão do MATLAB, sendo dispostas da seguinte forma:

- Redes Perceptron Multicamadas, de Retropagação em Cascata e Elman de Retropropagação – sigmóide tangente hiperbólica para camadas de neurônio de entrada/escondida; linear para camada de saída;
- Rede de Reconhecimento de Padrões – base radial para camadas de neurônio de entrada/escondida; *softmax* para camada de saída;

- Rede Neural Probabilística – base radial para camadas de neurônio de entrada; *compet* para camada de saída;
- Rede de Regressão Generalizada – base radial para camadas de neurônio de entrada; linear para camada de saída;
- Redes de Base Radial e Base Radial Exata – base radial para camada escondida; linear para camada de saída.

4.3.3 Quantidades de neurônios, camadas escondidas e taxa de propagação

Os parâmetros alterados de maneira geral nas RNAs foram as quantidades de neurônios, camadas escondidas e a taxa de propagação (*spread*), determinados de maneira heurística, por tentativa e erro, uma vez que não há uma fórmula exata para se encontrar o melhor resultado com o mínimo erro (STATHAKIS, 2009).

Para as redes RNPMC, RNRC, RNER e RNRP, partiu-se, inicialmente, de uma quantidade de 3 neurônios – mesma quantidade de entradas –, que foram elevados de forma gradual, observando-se os aumentos dos desempenhos, tanto nos casos de camadas de entrada única, como em conjunto com camada escondida, aproximando-se dos métodos de tentativa e erro e do modo básico descritos por Panchal F. S. e Panchal M. (2014).

As redes RNP e RNRG projetam diretamente as quantidades de linhas da matriz de vetores do banco de dados como quantidades de neurônios na camada de entrada com função de ativação de base radial, e tiveram seu parâmetro principal, a taxa de propagação, variada progressivamente. Com baixa taxa de propagação, os dados de entrada são aproximados; elevando-se a taxa de propagação, os dados de entrada são suavizados e uniformizados (Neural Network Toolbox User's Guide, 2010). A taxa de propagação também foi o parâmetro fundamental modificado nas RNAs RNBRE e RNBR. A RNBR possui três parâmetros adicionais em sua estrutura, em relação às três anteriores: erro quadrático médio (fixado em $1e^{-7}$), quantidade de neurônios na camada escondida e quantidade de neurônios para se adicionar entre suas exibições (mantido em 1). Da mesma forma que as quantidades de neurônios das outras quatro redes neurais, esses parâmetros foram modificados com tentativas e erros, baseando-se nos desempenhos das saídas das redes (PANCHAL, F. S. e PANCHAL, M., 2014). Para essas RNAs, o valor inicial de taxa de propagação foi de 0,01.

Para cada rede neural, foram realizados 10 ensaios com parâmetros diferentes, apresentados na Tabela 2. Os parâmetros entre colchetes separados por espaços são referentes aos números de neurônios entre camadas, para o caso de multicamadas.

Tabela 2: Parâmetros utilizados nas RNAs.

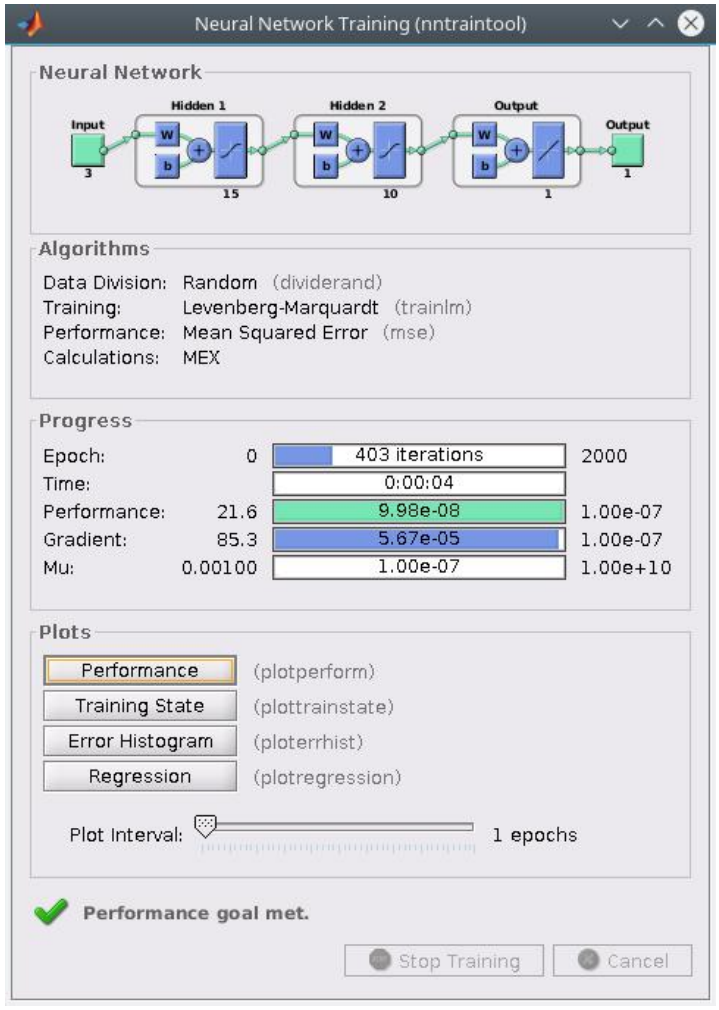
Ensaio	**RNPMC, RNRC, RNER e RNRP**	**RNP**	**RNRG**	**RNBR**		**RNBRE**
	Neurônios	**Taxa prop.**	**Taxa prop.**	**Taxa prop.**	**Neurônios**	**Taxa prop.**
1	3	0,01	0,01	0,01	100	0,01
2	5	0,1	0,1	0,1	100	0,1
3	10	0,2	0,2	1	100	1
4	15	0,4	0,4	10	100	10
5	30	1	1	1500	100	13
6	[10 5]	1,2	1,2	1650	100	18
7	[15 10]	2	2	1800	80	20
8	[20 15]	10	10	1850	90	22
9	[15 10 5]	20	20	1850	100	25
10	[20 15 10]	30	100	2000	30	100

Fonte: O autor.

4.3.4 Particularidades

Na Figura 18, apresenta-se o exemplo da tela de treinamento de um ensaio da RNA RNRC com 2 camadas escondidas contendo 15 neurônios na primeira delas e 10 na segunda. Nela é possível notar as 3 entradas referentes aos parâmetros da TC e a única saída, bem como as configurações de época, erro e algoritmo de treinamento. As redes RNPMC, RNER e RNRP possuem telas similares, diferentemente das RNAs RNP, RNRG, RNBR e RNBRE, com aspectos em que se contemplam apenas os esquemas de redes do campo *"Neural Network"*.

Figura 18: Tela de treinamento da Rede de Retropropagação em Cascata.



Fonte: O autor.

4.3.5 Divisões dos dados

No MATLAB, as arquiteturas das redes RNPMC, RNRC, RNER e RNRP permitem separar os dados de forma automática em percentuais de treinamento, validação e teste. Foram efetuados ensaios com 70% dos dados destinados ao treinamento, 15% à validação e 15% ao teste, utilizando os mesmos conjuntos para a comparação justa das quatro RNAs.

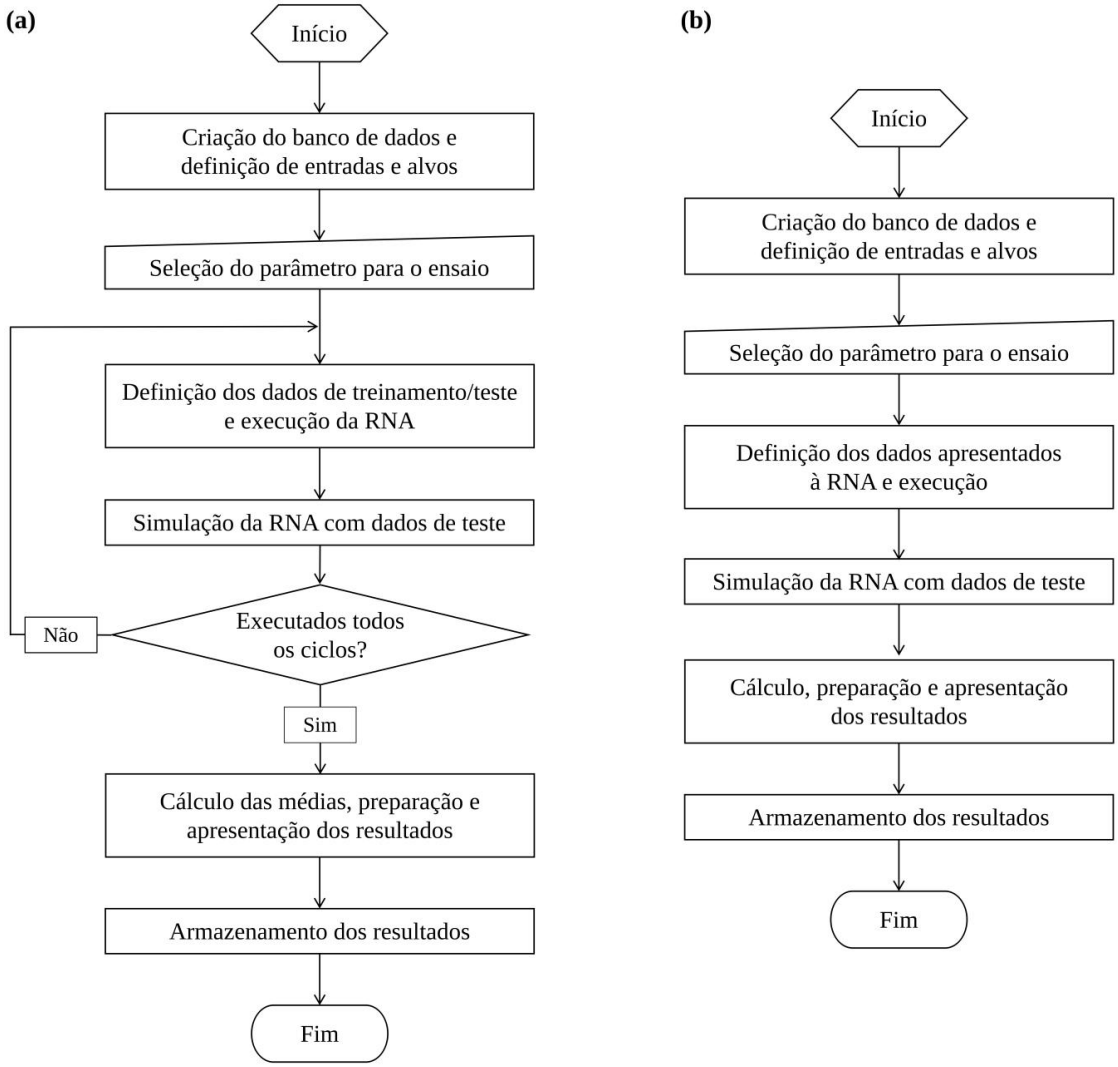
Entretanto, as redes RNP, RNRG, RNBR e RNBRE não possuem esta possibilidade em suas estruturas, o que foi resolvido utilizando-se uma segmentação de 50% dos dados para o treinamento das redes e 50% para os testes delas. A divisão consistiu na intercalação das linhas da matriz de dados, ocupando-se as ímpares nos treinamentos e as pares para os testes. Nesse cenário, a mesma divisão foi efetuada para as redes RNPMC, RNRC, RNER e RNRP, e outra sequência de ensaios foi realizada para elas, buscando-se uma comparação mais adequada entre as oito RNAs. Com isso, também foi possível fazer a equiparação dos desempenhos para essas quatro redes perante dois modelos de divisão de dados.

4.3.6 Etapas de execução das RNAs e metodologias dos ensaios

As etapas das execuções das RNAs foram convenientemente efetuadas de acordo com as particularidades estruturais em cada *script*. Conforme está apresentado na Figura 19, dois fluxogramas mostram os passos desenvolvidos para que todas as redes neurais pudessem ser, ao final, comparadas de forma apropriada. O que mais se diferencia é o fato de que existe um *loop* no fluxograma da Figura 19 (a), relativo às RNAs RNPMC, RNRC, RNER e RNRP: neste caso, as redes foram treinadas e simuladas por dez vezes, para se assegurar um resultado médio, garantindo-se que mesmo que se execute uma rede iniciando com pesos sinápticos diferentes e aleatórios, existe repetibilidade para a parametrização principal realizada.

As redes RNP, RNRG, RNBR e RNBRE, cujas etapas são mostradas no fluxograma da Figura 19 (b), dispensam a execução de vários ciclos por produzirem os mesmos resultados em diferentes execuções quando não existe alterações em parâmetros.

Figura 19: Fluxogramas das etapas de execução das redes neurais: (a) para RNPMC, RNRC, RNER e RNRP e (b) referente às redes RNP, RNRG, RNBR e RNBRE.



Fonte: O autor.

Foram realizados dez ensaios para cada RNA, nos quais foram alterados os parâmetros mostrados na Tabela 2 e armazenados os resultados correspondentes para comparações de desempenhos. Para tal, foram calculadas as quantidades de acertos das redes, seus percentuais, os erros quadráticos médios e as regressões, mostradas graficamente para os melhores resultados de cada RNA no Capítulo 5. Também são apresentados gráficos dos melhores resultados relacionados aos alvos de cada rede neural.

Capítulo 5 – Resultados e Discussão

Neste capítulo estão apresentados os resultados obtidos nos ensaios para cada uma das redes neurais, conforme as parametrizações utilizadas. Desta forma, é possível se observar seus desempenhos para a mínima parametrização que resulta no máximo número de acertos, buscando-se a utilização do processamento computacional de forma racional.

As apresentações consistem em tabelas mostrando a quantidade e o percentual de acertos obtidos pelas simulações das redes, os erros quadráticos médios e as regressões. Graficamente, são exibidas as regressões e a relação entre todos os pontos alvos, referentes às sete amostras de leite puro e adulterado, equiparados aos resultados obtidos nas melhores parametrizações entre os ensaios realizados. Nestes gráficos, os números de 1 a 7 são referências aos números atribuídos inicialmente às sete amostras de leite (especificados no Item 4.2). No gráfico de regressão, a linha tracejada representa o objetivo que relaciona alvos e acertos; as circunferências na cor preta são os pontos dos resultados do teste; e a linha azul, a relação entre esses pontos resultantes e a linha tracejada. Os pontos mencionados se referem aos conjuntos de três parâmetros obtidos pela TC.

Para as devidas comparações, foi adotado, por convenção, a apresentação de gráficos das divisões de dados “50/50”, ou seja, para os experimentos que utilizaram 50% dos dados para treinamento e 50% para teste (700 linhas da matriz de dados para cada). Entretanto, nas tabelas das redes RNPMC, RNRC, RNER e RNRP estão apresentados também os acertos e os valores de erro quadrático médio e regressão, obtidos em seus treinamentos e validações (cujos dados foram divididos em 70% para treinamento, referentes a 980 linhas da matriz de dados; 15% para validação e 15% para teste, 210 linhas da matriz de dados para cada: “70/15/15”). As quantidades de acertos dessas quatro redes são mostradas por números decimais porque são as médias dos resultados dos dez ciclos realizados pelas RNAs. Como pode ser observado nas tabelas de cada rede neural, as diferenças entre os resultados obtidos para as RNAs com as duas formas de divisão de dados é pequena; entretanto, para aplicação prática, é preferível a utilização da divisão “70/15/15” (uma vez que esta configuração já é disponível em suas arquiteturas), para que a validação sempre garanta uma melhor otimização dos parâmetros.

Por fim, é mostrada uma tabela para sumarizar os melhores desempenhos de cada RNA.

5.1 Rede Neural Perceptron Multicamadas

A rede RNPMC, parametrizada com 3 neurônios no primeiro ensaio, já foi capaz de acertar mais da metade das amostras apresentadas. Das dez experimentações realizadas, obteve melhor desempenho no ensaio 9, com 15 neurônios na camada de entrada, 10 na primeira camada escondia e 5 na segunda camada escondida, como pode ser observado na Tabela 3.

No geral, pode-se concluir que, à partir da inclusão de uma camada escondida, no 6º ensaio, o desempenho da rede RNPMC apresenta quandidades de acertos satisfatórios, com erros quadráticos médios próximos de 0 e regressões perto de 1 – condições praticamente ideais.

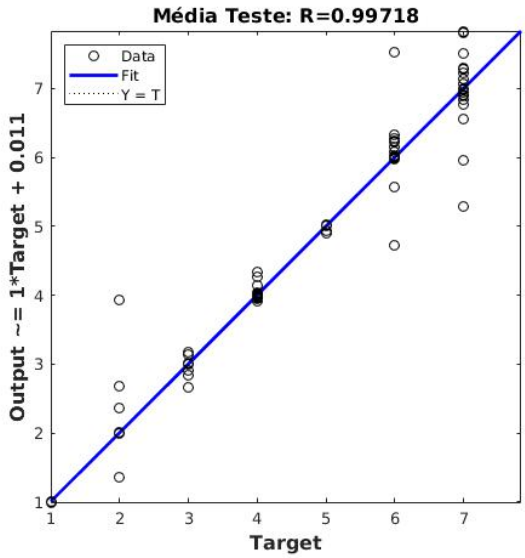
Tabela 3: Resultados registrados para a RNPMC.

Ens.	Parâmet.	Divisão dados	Total dados	Quant. acertos	Percent. ace. (%)	EQM			Regressão		
						Trein.	Valid.	Teste	Trein.	Valid.	Teste
1	3	70/15/15	210	124	59,05	0,67	0,71	0,71	0,91	0,91	0,91
		50/50	700	444,4	63,49	0,68		0,67	0,91		0,91
2	5	70/15/15	210	154,3	73,48	0,46	0,48	0,48	0,94	0,94	0,94
		50/50	700	554,2	79,17	0,33		0,33	0,96		0,96
3	10	70/15/15	210	177,5	84,52	0,24	0,27	0,27	0,97	0,97	0,97
		50/50	700	633,6	90,51	0,13		0,16	0,98		0,98
4	15	70/15/15	210	189	90,00	0,16	0,16	0,21	0,98	0,98	0,97
		50/50	700	681,6	97,37	0,02		2,22	1,00		0,93
5	30	70/15/15	210	199,6	95,05	0,08	0,09	0,13	0,99	0,99	0,98
		50/50	700	683,3	97,61	0,00		1,95	1,00		0,86
6	[10 5]	70/15/15	210	200,5	95,48	0,07	0,08	0,13	0,99	0,99	0,98
		50/50	700	688,1	98,30	0,00		0,35	1,00		0,96
7	[15 10]	70/15/15	210	206,7	98,43	0,01	0,03	0,04	1,00	1,00	1,00
		50/50	700	684,1	97,73	0,00		0,26	1,00		0,97
8	[20 15]	70/15/15	210	206,1	98,14	0,01	0,03	0,05	1,00	1,00	0,99
		50/50	700	683,5	97,64	0,00		0,12	1,00		0,99
9	[15 10 5]	70/15/15	210	208	99,05	0,01	0,01	0,03	1,00	1,00	1,00
		50/50	700	689,5	98,50	0,00		0,14	1,00		0,98
10	[20 15 10]	70/15/15	210	207	98,57	0,00	0,01	0,03	1,00	1,00	1,00
		50/50	700	689,3	98,47	0,00		0,05	1,00		0,99

Fonte: O autor.

Na Figura 20 está apresentada a regressão da média do teste do melhor dos ensaios realizados para a rede RNPMC.

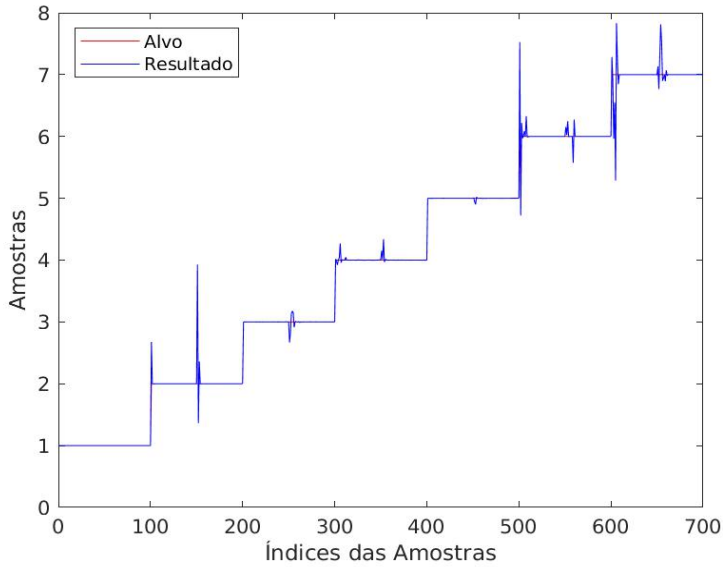
Figura 20: Regressão do teste do 9º ensaio da rede RNPMC.



Fonte: O autor.

O gráfico da Figura 21 mostra os resultados obtidos para todos os pontos de teste (na cor azul) em relação aos alvos (na cor vermelha) que se desejou encontrar com a rede no ensaio 9. Na média, foram 10,5 pontos desviados, para o total de 700.

Figura 21: Relação de respostas pelos alvos da rede RNPMC, referentes ao 9º ensaio.



Fonte: O autor.

5.2 Rede Neural de Retropropagação em Cascata

A rede RNRC apresentou resultados semelhantes à RNPMC, de forma geral, e identificou 90% das amostras no 3º ensaio com 10 neurônios na camada de entrada. Neste ensaio, o EQM foi de 0,25 e a regressão, de 0,97, no teste. Conforme mostrado na Tabela 4, os ensaios 9 e 10 com duas camadas escondidas, resultaram em percentuais de acertos acima de 98%, com erros quadráticos médios muito próximos de 0 e regressão aproximada a 1. O melhor desempenho foi obtido no 10º ensaio, ligeiramente superior ao 9º.

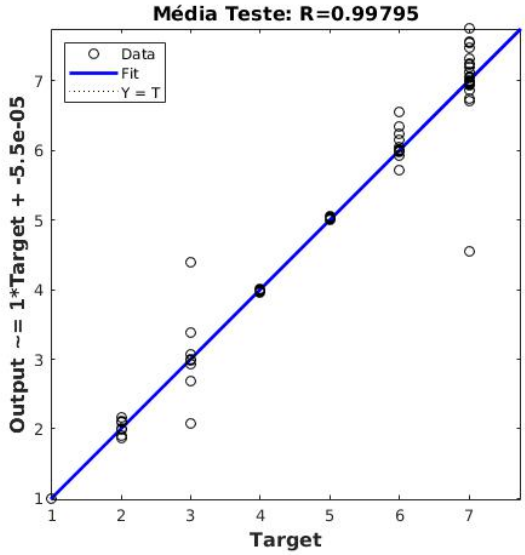
Tabela 4: Resultados registrados para a RNRC.

Ens.	Parâmet.	Divisão dados	Total dados	Quant. acertos	Percent. ace. (%)	EQM			Regressão		
						Trein.	Valid.	Teste	Trein.	Valid.	Teste
1	3	70/15/15	210	124,9	59,48	0,73	0,76	0,73	0,90	0,90	0,90
		50/50	700	437,5	62,50	0,71		0,70	0,91		0,91
2	5	70/15/15	210	161,7	77,00	0,40	0,38	0,40	0,95	0,95	0,95
		50/50	700	542,3	77,47	0,41		0,41	0,95		0,95
3	10	70/15/15	210	189	90,00	0,15	0,16	0,18	0,98	0,98	0,98
		50/50	700	636,8	90,97	0,11		0,25	0,99		0,97
4	15	70/15/15	210	191,6	91,24	0,13	0,13	0,18	0,98	0,98	0,98
		50/50	700	668,1	95,44	0,04		1,81	0,99		0,93
5	30	70/15/15	210	198,5	94,52	0,07	0,10	0,13	0,99	0,99	0,98
		50/50	700	684,2	97,74	0,00		0,87	1,00		0,92
6	[10 5]	70/15/15	210	204,5	97,38	0,04	0,05	0,07	0,99	0,99	0,99
		50/50	700	685,3	97,90	0,00		0,14	1,00		0,98
7	[15 10]	70/15/15	210	203,4	96,86	0,04	0,05	0,10	0,99	0,99	0,99
		50/50	700	685,8	97,97	0,00		0,15	1,00		0,98
8	[20 15]	70/15/15	210	203	96,67	0,03	0,05	0,08	1,00	0,99	0,99
		50/50	700	685,5	97,93	0,00		0,08	1,00		0,99
9	[15 10 5]	70/15/15	210	206,3	98,24	0,01	0,03	0,03	1,00	1,00	1,00
		50/50	700	688,1	98,30	0,00		0,10	1,00		0,99
10	[20 15 10]	70/15/15	210	205	97,62	0,01	0,03	0,06	1,00	1,00	0,99
		50/50	700	688,5	98,36	0,00		0,04	1,00		1,00

Fonte: O autor.

A regressão da média do teste do 10º ensaio da rede RNRC pode ser vista na Figura 22.

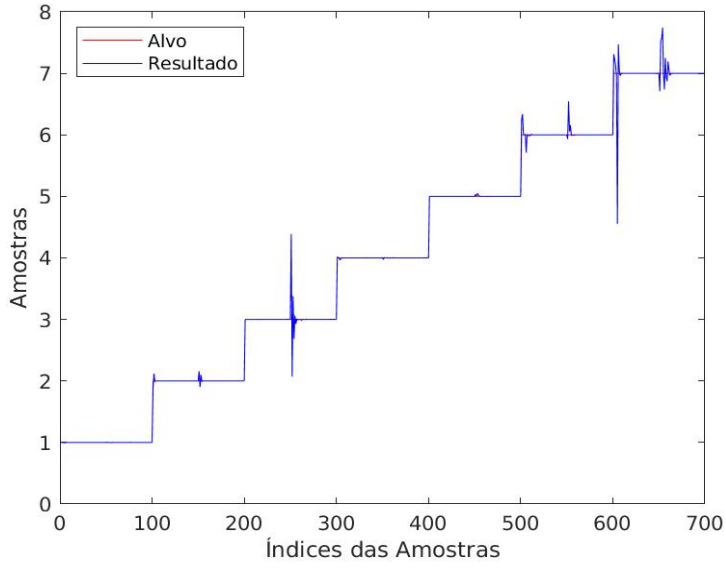
Figura 22: Regressão do teste do 10º ensaio da rede RNRC.



Fonte: O autor.

Na Figura 24 é apresentado o gráfico para o ensaio 10, cujos resultados obtidos para todos os pontos de teste estão na cor azul e seus respectivos alvos, na cor vermelha. Para o total de 700 vetores, 11,5, em média, foram identificados de forma incorreta.

Figura 23: Relação de respostas pelos alvos da rede RNRC, referentes ao 10º ensaio.



Fonte: O autor.

5.3 Rede Neural Elman de Retropropagação

A rede RNER obteve resultados similares às RNAs RNPMC e RNRC. No 5º ensaio, com 30 neurônios na camada de entrada, conseguiu 97,74% das identificações dos conjuntos de amostras de leite (puro e adulterados). O melhor resultado entre as experimentações realizadas foi obtido no ensaio 9, com a parametrização que teve 15 neurônios na camada de entrada, 10 na primeira camada escondida e 5 na segunda camada escondida: média de 693,4 acertos, do total de 700 pontos de testes, EQM de 0,12 e regressão de 0,99. Os resultados de todos os ensaios da rede RNER são mostrados na Tabela 5.

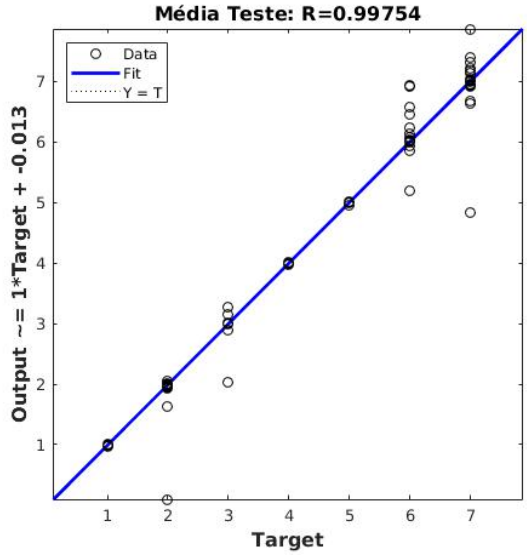
Tabela 5: Resultados registrados para a RNER.

Ens.	Parâmet.	Divisão dados	Total dados	Quant. acertos	Percent. ace. (%)	EQM			Regressão		
						Trein.	Valid.	Teste	Trein.	Valid.	Teste
1	3	70/15/15	210	122,8	58,48	0,80	0,83	0,81	0,89	0,89	0,89
		50/50	700	445,3	63,61	0,55		0,55	0,93		0,93
2	5	70/15/15	210	149,1	71,00	0,51	0,52	0,54	0,93	0,93	0,93
		50/50	700	561,3	80,19	0,34		0,34	0,96		0,96
3	10	70/15/15	210	187,5	89,29	0,17	0,17	0,20	0,98	0,98	0,97
		50/50	700	648,7	92,67	0,09		0,12	0,99		0,98
4	15	70/15/15	210	192,7	91,76	0,13	0,14	0,17	0,98	0,98	0,98
		50/50	700	680,5	97,21	0,02		0,34	1,00		0,97
5	30	70/15/15	210	198,1	94,33	0,08	0,09	0,14	0,99	0,99	0,98
		50/50	700	684,2	97,74	0,00		0,88	1,00		0,93
6	[10 5]	70/15/15	210	202,8	96,57	0,03	0,04	0,09	1,00	1,00	0,99
		50/50	700	692,1	98,87	0,00		0,06	1,00		0,99
7	[15 10]	70/15/15	210	207,7	98,90	0,01	0,02	0,03	1,00	1,00	1,00
		50/50	700	688,2	98,31	0,00		0,13	1,00		0,98
8	[20 15]	70/15/15	210	208,3	99,19	0,01	0,01	0,03	1,00	1,00	1,00
		50/50	700	688,7	98,39	0,00		0,07	1,00		0,99
9	[15 10 5]	70/15/15	210	208,7	99,38	0,00	0,01	0,03	1,00	1,00	1,00
		50/50	700	693,4	99,06	0,00		0,12	1,00		0,99
10	[20 15 10]	70/15/15	210	208,5	99,29	0,00	0,00	0,02	1,00	1,00	1,00
		50/50	700	691,7	98,81	0,00		0,05	1,00		0,99

Fonte: O autor.

Na Figura 24 está apresentada a regressão da média do teste do 9º ensaio da rede RNER.

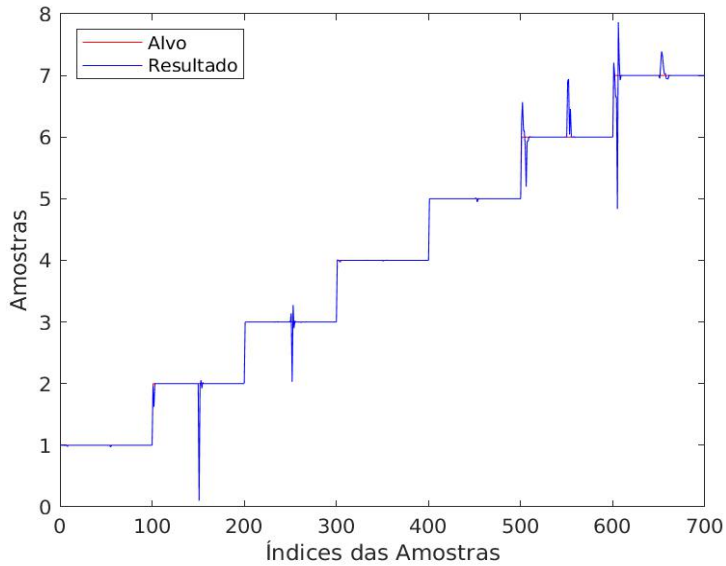
Figura 24: Regressão do teste do 9º ensaio da rede RNER.



Fonte: O autor.

A relação entre resultados obtidos pela rede RNER e seus alvos está mostrada no gráfico da Figura 25. Do total de 700 pontos, 6,6, em média, não atingiram seus alvos.

Figura 25: Relação de respostas pelos alvos da rede RNER, referentes ao 9º ensaio.



Fonte: O autor.

5.4 Rede Neural de Reconhecimento de Padrões

Os resultados dos ensaios da Rede Neural de Reconhecimento de Padrões (RNRP) são parecidos com os das RNAs RNPMC, RNRC e RNER, mas em geral, ligeiramente melhores. No 6º ensaio, a rede não conseguiu identificar apenas 5,4 (média) do total de 700 vetores, sendo muito próximo dos resultados obtidos pelo 10º ensaio – com uma camada escondida a menos e menor quantidade de vetores, o que, do ponto de vista do uso computacional, é mais vantajoso, por utilizar menos processamento algébrico. Na Tabela 6 podem ser conferidos todos os resultados dos ensaios.

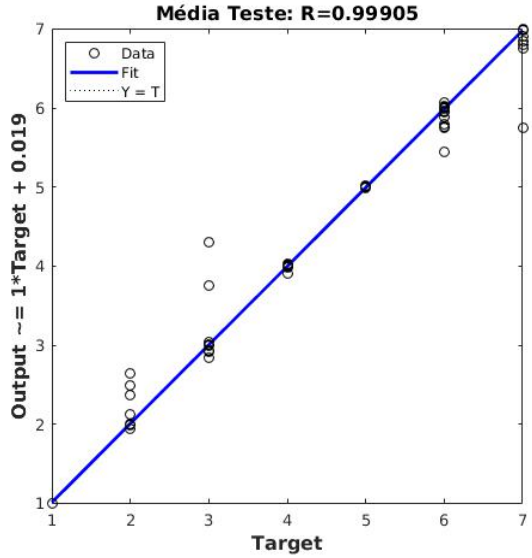
Tabela 6: Resultados registrados para a RNRP.

Ens.	Parâmet.	Divisão dados	Total dados	Quant. acertos	Percent. ace. (%)	EQM			Regressão		
						Trein.	Valid.	Teste	Trein.	Valid.	Teste
1	3	70/15/15	210	135,3	64,43	0,60	0,63	0,63	0,92	0,92	0,92
		50/50	700	493,1	70,44	0,59		0,58	0,92		0,92
2	5	70/15/15	210	166,9	79,48	0,26	0,28	0,30	0,97	0,96	0,96
		50/50	700	585,1	83,59	0,20		0,21	0,97		0,97
3	10	70/15/15	210	191,3	91,10	0,10	0,12	0,14	0,99	0,99	0,98
		50/50	700	676,9	96,70	0,03		0,07	1,00		0,99
4	15	70/15/15	210	196,2	93,43	0,08	0,07	0,12	0,99	0,99	0,98
		50/50	700	690,8	98,69	0,00		0,08	1,00		0,99
5	30	70/15/15	210	199,3	94,90	0,06	0,05	0,11	0,99	0,99	0,99
		50/50	700	690,8	98,69	0,00		0,10	1,00		0,99
6	[10 5]	70/15/15	210	195	92,86	0,10	0,13	0,13	0,99	0,98	0,98
		50/50	700	694,6	99,23	0,00		0,05	1,00		0,99
7	[15 10]	70/15/15	210	207,7	98,90	0,01	0,01	0,04	1,00	1,00	1,00
		50/50	700	692,2	98,89	0,00		0,05	1,00		0,99
8	[20 15]	70/15/15	210	207,8	98,95	0,01	0,02	0,02	1,00	1,00	1,00
		50/50	700	693,1	99,01	0,00		0,04	1,00		0,99
9	[15 10 5]	70/15/15	210	203,9	97,10	0,03	0,03	0,06	1,00	1,00	0,99
		50/50	700	693,7	99,10	0,00		0,05	1,00		0,99
10	[20 15 10]	70/15/15	210	203,3	96,81	0,03	0,07	0,07	1,00	0,99	0,99
		50/50	700	694,5	99,21	0,00		0,03	1,00		1,00

Fonte: O autor.

A regressão média do teste do 6º ensaio é apresentada na Figura 26.

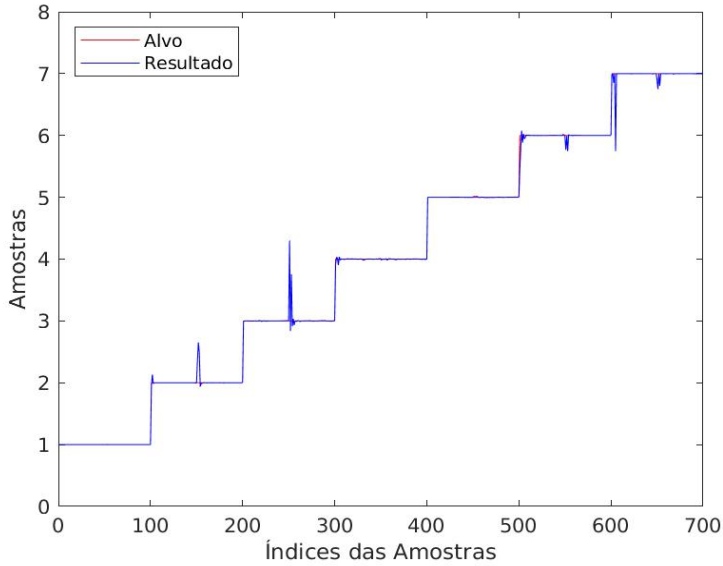
Figura 26: Regressão do teste do 6º ensaio da rede RNRP.



Fonte: O autor.

Os melhores resultados médios da saída da RNA (na cor azul) são comparados com os alvos (na cor vermelha) no gráfico da Figura 27.

Figura 27: Relação de respostas pelos alvos da rede RNRP, referentes ao 6º ensaio.



Fonte: O autor.

5.5 Rede Neural Probabilística

A Rede Neural Probabilística (RNP) apresentou o melhor resultado entre os ensaios realizados com uma taxa de propagação de 20, conforme mostrado na Tabela 7. Como pode ser observado no ensaio 9, a rede foi capaz de identificar 86,57% dos pontos das amostras de leite puro e adulterado. Entretando, quando comparada à RNA RNPMC para um resultado pouco inferior de percentual de acertos no ensaio 2, observado na Tabela 3, a rede RNP teve EQM do teste superior e regressão inferior. A comparação pontual destes dois ensaios indica que a rede Probabilística errou mais na sua previsão, embora tenha conseguido maior percentual de acertos reais do que a rede RNPMC.

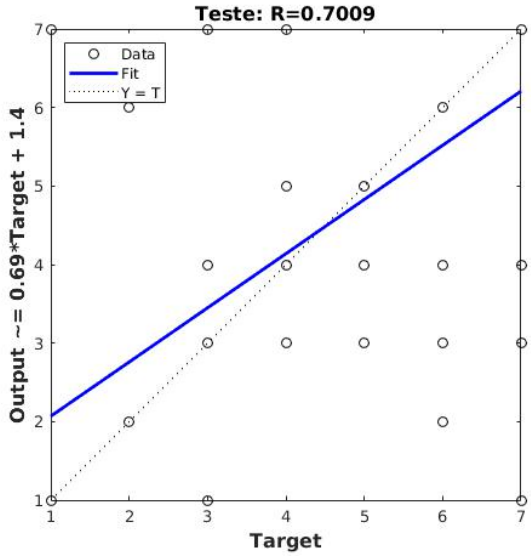
Tabela 7: Resultados registrados para a RNP.

Ensaio	**Parâmetro**	**Total de dados**	**Quantidade de acertos**	**Percentual de acertos (%)**	**Teste**	
					EQM	**Regressão**
1	0,01	700	104	14,86	12,94	0,02
2	0,1	700	171	24,43	11,83	0,09
3	0,2	700	228	32,57	10,74	0,17
4	0,4	700	312	44,57	9,06	0,28
5	1	700	443	63,29	5,55	0,53
6	1,2	700	463	66,14	5,23	0,53
7	2	700	518	74,00	4,45	0,54
8	10	700	604	86,29	2,44	0,69
9	20	700	606	86,57	2,38	0,70
10	30	700	606	86,57	2,33	0,71

Fonte: O autor.

Na Figura 28 está apresentada a regressão do teste do melhor resultado obtido com a rede RNP.

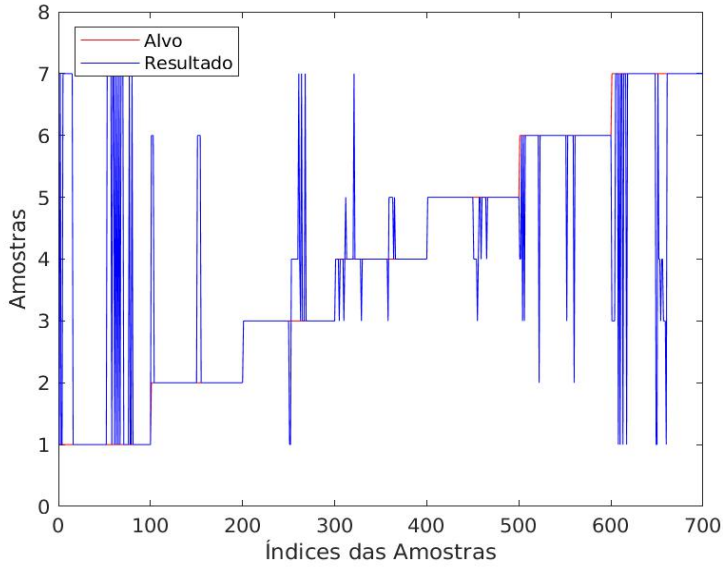
Figura 28: Regressão do teste do 9º ensaio da rede RNP.



Fonte: O autor.

No gráfico da Figura 29, está mostrada a relação entre resultados obtidos e seus alvos para o 9º ensaio da rede RNP, que foi capaz de acertar 606 pontos do total de 700 no teste.

Figura 29: Relação de respostas pelos alvos da rede RNP, referentes ao 9º ensaio.



Fonte: O autor.

5.6 Rede Neural de Regressão Generalizada

Os resultados da rede RNRG apresentaram muito poucas alterações, mesmo com a variação da ordem de até 10000 vezes da taxa de propagação nos 10 ensaios realizados, como pode ser observado na Tabela 8. O melhor desempenho foi obtido no ensaio 9, quando foram acertados 607 pontos. Todavia, assim como na rede RNP, no teste, o EQM foi alto, e a regressão, baixa.

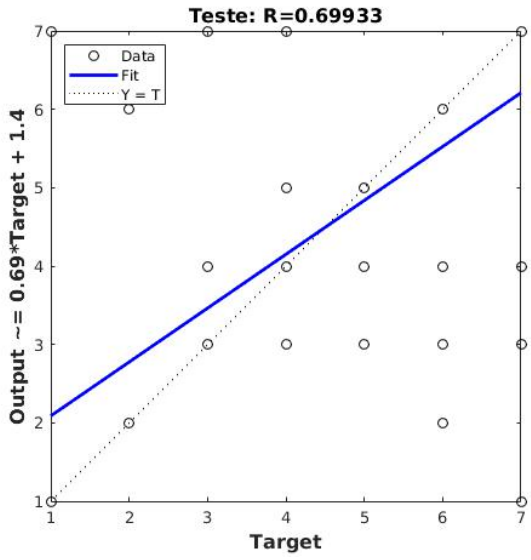
Tabela 8: Resultados registrados para a RNRG.

Ensaio	Parâmetro	Total de dados	Quantidade de acertos	Percentual de acertos (%)	Teste	
					EQM	Regressão
1	0,01	700	607	86,71	2,39	0,70
2	0,1	700	607	86,71	2,39	0,70
3	0,2	700	607	86,71	2,39	0,70
4	0,4	700	607	86,71	2,39	0,70
5	1	700	607	86,71	2,39	0,70
6	1,2	700	607	86,71	2,39	0,70
7	2	700	607	86,71	2,39	0,70
8	10	700	607	86,71	2,39	0,70
9	20	700	607	86,71	2,39	0,70
10	100	700	600	85,71	2,33	0,71

Fonte: O autor.

A regressão do teste referente ao 1º ensaio da rede RNRG é mostrada na Figura 30.

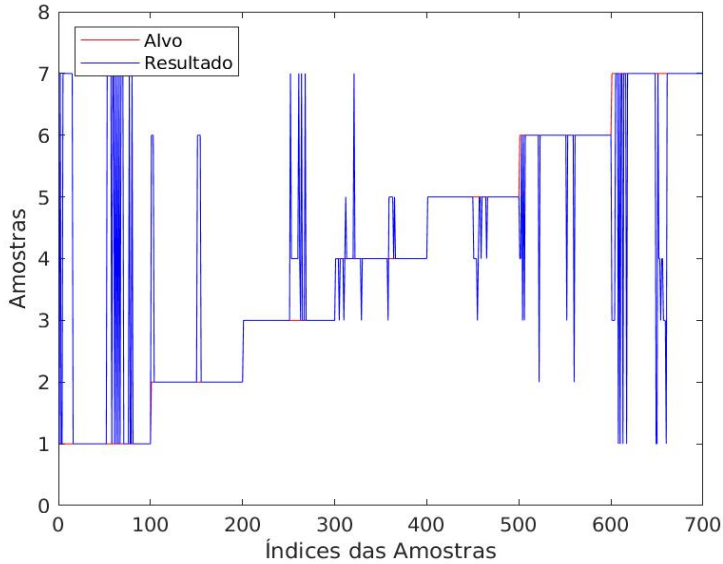
Figura 30: Regressão do teste do 1º ensaio da rede RNRG.



Fonte: O autor.

Na Figura 31, é possível se observar o gráfico com a relação entre os alvos (linhas vermelhas) e os valores obtidos (linhas azuis) na saída da rede, do ensaio 1. O percentual de acertos foi de 86,71.

Figura 31: Relação de respostas pelos alvos da rede RNRG, referentes ao 1º ensaio.



Fonte: O autor.

5.7 Rede Neural de Base Radial

Os parâmetros ajustados para a rede de base radial (RNBR) também necessitaram de grandes variações combinadas entre si, para que se fosse atingida a quantidade de acertos de 85%, referente ao ensaio 6. Na Tabela 9 estão demonstrados todos os resultados da RNA e é possível se notar que foi necessário aumentar de forma excessiva a taxa de propagação para que se fossem registrados 595 acertos, no melhor desempenho da rede, com número máximo de 100 neurônios, que teve pouco impacto quando alterado nos demais ensaios.

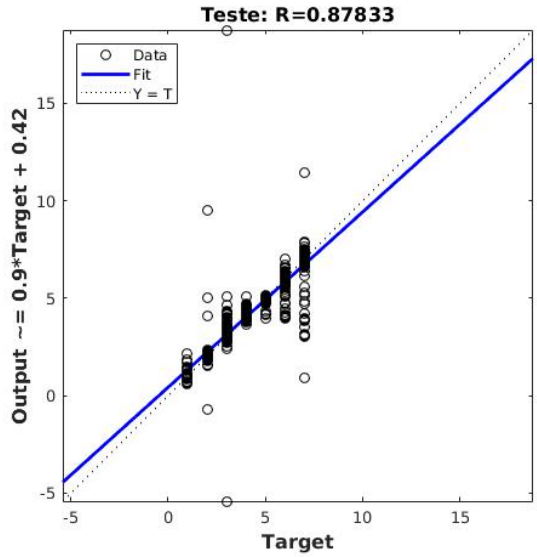
Tabela 9: Resultados registrados para a RNBR.

Ensaio	**Parâmetro**		**Total de dados**	**Quantidade de acertos**	**Percentual de acertos (%)**	**Teste**	
	Taxa propag.	**Máximo nº de neurônios (MN)**				**EQM**	**Regressão**
1	0,01	100	700	100	14,29	4,25	0,00
2	0,1	100	700	100	14,29	4,25	0,00
3	1	100	700	102	14,57	4,25	0,03
4	10	100	700	155	22,14	3,94	0,26
5	1500	100	700	545	77,86	1,01	0,88
6	1650	100	700	595	85,00	1,00	0,88
7	1800	80	700	568	81,14	0,68	0,91
8	1850	90	700	584	83,43	0,86	0,89
9	1850	100	700	586	83,71	0,81	0,90
10	2000	30	700	394	56,29	0,78	0,90

Fonte: O autor.

Os dados de teste produziram uma regressão aproximada de 0,88, mostrado na Figura 32, para o melhor resultado entre os ensaios da rede de base radial, em relação a acertos. Apesar de apresentarem melhores regressões e menores erros quadráticos médios em relação ao 6º ensaio, as experimentações 7, 8, 9 e 10 produziram menos acertos.

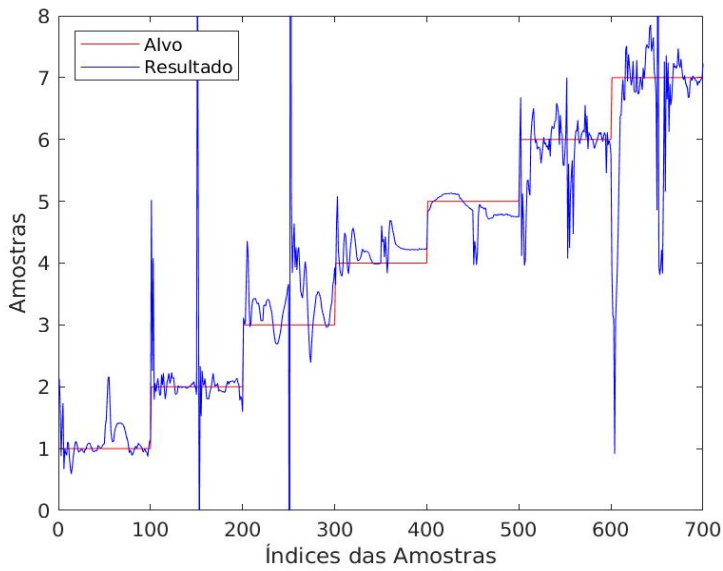
Figura 32: Regressão do teste do 6º ensaio da rede RNBR.



Fonte: O autor.

A relação entre os melhores resultados obtidos pela RNBR e seus alvos está mostrada no gráfico da Figura 33. De 700 pontos, a rede errou 105.

Figura 33: Relação de respostas pelos alvos da rede RNBR, referentes ao 6º ensaio.



Fonte: O autor.

5.8 Rede Neural de Base Radial Exata

A rede de base radial exata (RNBRE) foi capaz de identificar menos da metade das amostras apresentadas para a sua melhor parametrização entre os ensaios realizados, como pode ser verificado na Tabela 10. O melhor resultado, obtido no 9º ensaio, ainda mostrou no teste, EQM alto, de 2,35, e regressão de apenas 0,67 – esses indicadores extrapolaram no 10º ensaio, mesmo com percentual de acertos de 29,29.

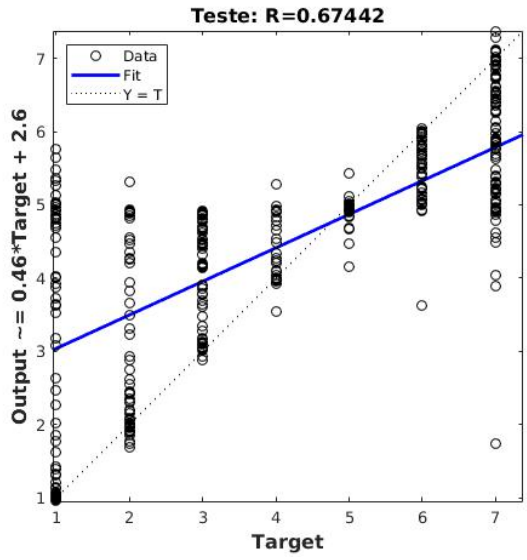
Tabela 10: Resultados registrados para a RNBRE.

Ensaio	Parâmetro	Total de dados	Quantidade de acertos	Percentual de acertos (%)	Teste	
					EQM	Regressão
1	0,01	700	100	14,29	13,00	0,00
2	0,1	700	100	14,29	5,00	0,04
3	1	700	108	15,43	4,88	0,12
4	10	700	200	28,57	3,40	0,51
5	13	700	233	33,29	3,10	0,56
6	18	700	270	38,57	2,73	0,62
7	20	700	285	40,71	2,63	0,64
8	22	700	305	43,57	2,53	0,65
9	25	700	326	46,57	2,35	0,67
10	100	700	205	29,29	148028,69	-0,07

Fonte: O autor.

A regressão da melhor parametrização da RNBRE é mostrada na Figura 34.

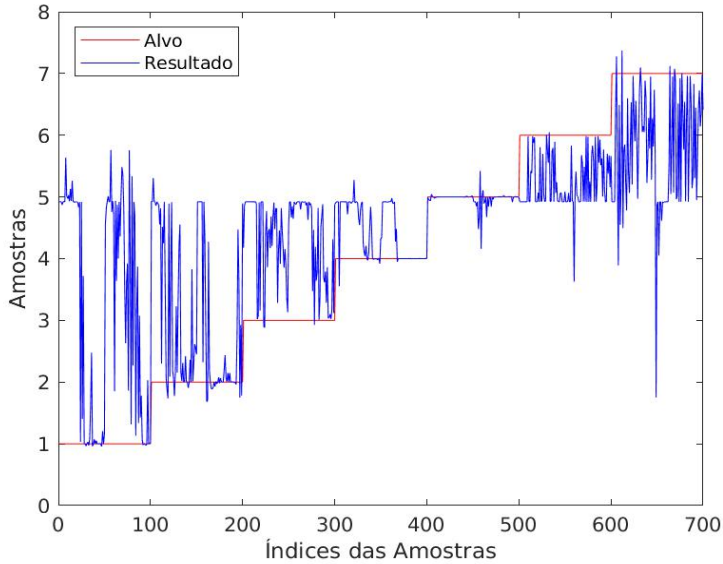
Figura 34: Regressão do teste do 9º ensaio da rede RNBRE.



Fonte: O autor.

Na Figura 35 está apresentado o gráfico dos resultados obtidos (na cor azul) no ensaio 9, em relação aos seus alvos (na cor vermelha). A rede RNBRE acertou 374 pontos do total de 700, e é possível se notar com clareza os desvios.

Figura 35: Relação de respostas pelos alvos da rede RNBRE, referentes ao 9º ensaio.



Fonte: O autor.

5.9 Resumo comparativo dos resultados

O resumo dos melhores resultados obtidos pelas RNAs está apresentado na Tabela 11, organizado do maior percentual de acertos para o menor, entre as redes. Nesse caso, foi considerada a divisão dos dados de “50/50”, utilizada em todas as redes, para efeito comparativo absoluto. As redes RNRP, RNER, RNPMC e RNRC, foram capazes de identificar mais de 98% das amostras de leite (puro e adulterado), com erros quadráticos médios muito próximos de 0 e regressões próximas de 1 o que indica consistência e qualidade nas previsões. As redes de base radial RNRG, RNP e RNBR tiveram desempenhos inferiores em todos os indicadores nesta comparação, e a pior RNA para os dados utilizados, foi a de base radial exata.

Tabela 11: Comparação dos melhores resultados das RNAs, ordenados da rede que mais acertou (Reconhecimento de Padrões) para a que menos obteve percentual de acertos (RNBRE).

Rede	Ensaio	Total dados	Quant. acertos	Percentual de acertos (%)	EQM		Regressão	
					Trein.	Teste	Trein.	Teste
RNRP	6	700	694,6	99,23	0,00	0,05	1,00	0,99
RNER	9	700	693,4	99,06	0,00	0,12	1,00	0,99
RNPMC	9	700	689,5	98,50	0,00	0,14	1,00	0,98
RNRC	10	700	688,5	98,36	0,00	0,04	1,00	1,00
RNRG	1	700	607	86,71		2,39		0,70
RNP	9	700	606	86,57		2,38		0,70
RNBR	6	700	595	85,00		1,00		0,88
RNBRE	9	700	326	46,57		2,35		0,67

Fonte: O autor.

Conclusão

Os parâmetros oriundos do tratamento de sinais da técnica cromática para amostras de leite puro e contaminado por três adulterantes em baixas proporções foram utilizados para a criação de um banco de dados com todas as variáveis disponíveis, em que buscou-se verificar suas identificações por meio do uso de sistemas inteligentes baseados em Redes Neurais Artificiais.

Foram usadas oito ferramentas de RNAs diferentes para identificar e comparar os resultados para 7 amostras de leite puro e adulterado por bicarbonato de sódio, ureia agrícola e peróxido de hidrogênio, nas titulações de 0,5 e 1%. Como pode ser visto no Capítulo 5, as

redes de Reconhecimento de Padrões, Elman de Retropropagação, Perceptron Multicamadas e de Retropropagação em Cascata, nas parametrizações utilizadas, foram capazes de identificar mais de 98% das amostras, com valores de erros quadráticos médios de 0,05, 0,12, 0,14 e 0,04, respectivamente, e regressão de 0,99, 0,99, 0,98 e 1, nessa ordem, para as 4 RNAs, indicando qualidade nas identificações. É importante ressaltar que a maioria das amostras apresenta bastante variação entre o início e o fim de suas aquisições no tempo, tendo mais concentração entre os primeiros pontos, e ainda assim, essas RNAs obtiveram elevada taxa de acertos. Por outro lado, as redes de Regressão Generalizada, Probabilística, de Base Radial e de Base Radial Exata obtiveram resultados inferiores para os dados utilizados neste trabalho, sendo que esta última, não foi capaz de identificar nem metade do percentual das amostras.

A junção de uma das opções de ferramentas de RNA que demonstraram ter o mínimo de 98% de precisão (a rede de Reconhecimento de Padrões, por exemplo, que teve pequeno destaque no desempenho em relação às redes RNER, RNPMC, e RNRC, acertando 99,23% das amostras apresentadas), com o trabalho de tratamento dos sinais pela TC, que identificou contaminações de apenas 0,5 e 1% no leite, pode apresentar resultados muito promissores no cenário alimentício. A criação de um equipamento com essas duas técnicas deve oferecer vantagens de custo e rapidez em relação às metodologias tradicionais, auxiliando na tomadas de decisões de produtores, indústrias lácteas e vendedores, que buscam, cada vez mais, melhorar seus controles de qualidade. As consequências refletem na saúde das pessoas, que buscam alimentos de qualidade, além de desejarem pagar um preço justo pelos produtos.

Sugestões para Trabalhos Futuros

- Confeccionar o recipiente de amostragem com material de aço inoxidável, verificando a compatibilidade de respostas, para atender a exigências sanitárias.
- Analisar mais amostras para criar um banco de dados maior, aumentando-se a capacidade das redes neurais para reconhecimento de padrões.
- Extender a pesquisa para outros produtos que sofrem adulterações.
- Agregar os trabalhos de identificação pela técnica cromática com o de redes neurais artificiais.
- Efetuar novos ensaios com outras RNAs e softwares diferentes.

Referências Bibliográficas

ABRAHAM, A. Artificial neural networks. **Handbook of measuring system design**, 15 de Julho de 2005. 901-908. DOI: https://doi.org/10.1002/0471497398.mm421

ABRANTES, M. R. et al. Fraude em leite: Métodos de detecção e implicações para o consumidor. **Revista do Instituto Adolfo Lutz**. 2014. 244-251. DOI: 10.18241/0073-98552014731611

ALKHASAWNEH, M. S. et al. Intelligent landslide system based on discriminant analysis and cascade-forward back-propagation network. **Arabian Journal for Science and Engineering**, Abril de 2014. 5575-5584. DOI: https://doi.org/10.1007/s13369-014-1105-8

BARROS, V. P. A., **Avaliação do desempenho de algoritmos de retropropagação com redes neurais artificiais para a resolução de problemas não-lineares**. Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Ponta Grossa, PR, 2018. Disponível em http://repositorio.utfpr.edu.br/jspui/handle/1/3340. Acesso em 18 de Setembro de 2020

BROOMHEAD, D. S. e LOWE, D. Radial basis functions, multi-variable functional interpolation and adaptive networks. **Royal Signals and Radar Establishment Malvern (United Kingdom)**, 1988. Disponível em https://apps.dtic.mil/sti/pdfs/ADA196234.pdf. Acesso em 29 de Agosto de 2020

CHEN, C. e REN, J. The effect of controlled atmosphere storage on aroma components of Hami melon. In: **2011 International Conference on New Technology of Agricultural**. IEEE, 2011. 764-768. DOI: https://doi.org/10.1109/ICAE.2011.5943905

CHOPRA, I. Review of state of art of smart structures and integrated systems. **AIAA journal**, Novembro de 2002. 2145-2187. DOI: https://doi.org/10.2514/2.1561

CONZUELO, F. et al. An integrated amperometric biosensor for the determination of lactose in milk and dairy products. **Journal of agricultural and food chemistry**, 28 de Maio de 2010. 7141-7148. DOI: https://doi.org/10.1021/jf101173e

DE AGUIAR FILHO, G. A. et al. **Rede Neural e Processamento De Imagem Aplicados Em Um Sistema De Detecção e Alarme De Incêndio**. 2018. Disponível em https://ri.unipac.br/repositorio/wp-content/uploads/2019/09/2018-02-Gilmar-Alves-de-Aguiar-Filho.pdf. Acesso em 22 de Dezembro de 2020

DE ARAUJO, J. et al. Qualitative tests for the determination of fraud in raw milk: evaluation of the influence of analytical parameters of the tests and the stability of the samples as a function of time

and preservation form. **Research, Society and Development**. 2021. 1-13. DOI: http://dx.doi.org/10.33448/rsd-v10i11.19860

EMBRAPA. Anuário Leite 2020. Disponível em: https://www.embrapa.br/busca-de-publicacoes/-/publicacao/1124722/anuario-leite-2020-leite-de-vacas-felizes. Acesso em 14 de Janeiro de 2021

EU (EUROPEAN UNION). **Risk assessment report, hydrogen peroxide**. 2003

FELICE, C. J. et al. Impedance microbiology: quantification of bacterial content in milk by means of capacitance growth curves. **Journal of microbiological methods**. 1999. 37-42. DOI: https://doi.org/10.1016/S0167-7012(98)00098-0

FERREIRA, A. A. **Comparação de arquiteturas de redes neurais para sistemas de reconheceimento de padrões em narizes artificiais**. Dissertação (Mestrado) – Universidade Federal de Pernambuco. Recife, PE, 2004. Disponível em https://repositorio.ufpe.br/bitstream/123456789/2465/1/arquivo4572_1.pdf. Acesso em 13 de Dezembro de 2020

FISTER Jr. I. et al. A brief review of nature-inspired algorithms for optimization. **Elektrotehniski Vestnik**, 2013. 1-7. Disponível em: https://arxiv.org/pdf/1307.4186.pdf. Acesso em: 19 de Julho de 2020

FURTADO, M. I. V. **Redes neurais artificiais: uma abordagem para sala de aula,** 2019. Atena Editora, 2019. Disponível em https://www.atenaeditora.com.br/wp-content/uploads/2019/05/e-book-Redes-Neurais-Artificiais-uma-Abordagem-para-Sala-de-Aula.pdf. Acesso em: 15 de Agosto de 2021

GREGORIO, N. F. C. **Desenvolvimento de sistema de sensores para identificação de fraude no leite bovino**. Dissertação (Mestrado) – Universidade de São Paulo. Faculdade de Zootecnia e Engenharia de Alimentos, Pirassununga, SP, 2019. Disponível em https://www.teses.usp.br/teses/disponiveis/74/74134/tde-17022020-093254/publico/ME6409073COR.pdf

HANDFORD, C. E. et al. Impacts of milk fraud on food safety and nutrition with special emphasis on developing countries. **Comprehensive Reviews in Food Science and Food Safety**, Janeiro de 2016. 130-142. DOI: https://doi.org/10.1111/1541-4337.12181

HARDING F. et al. Adulteration of milk. In: **Milk Quality**, 1995. 60-74. Springer, Boston, MA. DOI: https://doi.org/10.1007/978-1-4615-2195-2_5

HAYKIN, S. **Redes neurais: princípios e prática,** 2001. Bookman Editora, 2001

HEIMES, F. e VAN HEUVELN, B. The normalized radial basis function neural network. In: **SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)**. IEEE, 1998. 1609-1614. DOI: https://doi.org/10.1109/ICSMC.1998.728118

HRU KAR, M. et al. Evaluation of milk and dairy products by electronic tongue. **Mljekarstvo: časopis za unaprjeđenje proizvodnje i prerade mlijeka.** 2009. 193-200. Disponível em: https://hrcak.srce.hr/file/63540. Acesso em: 22 de Agosto de 2021

HUSH, D. R. e HORNE, B. G. Progress in supervised neural networks. **IEEE signal processing magazine**, Janeiro de 1993. 8-39. DOI: https://doi.org/10.1109/79.180705

ILONEN, J. et al. Differential evolution training algorithm for feed-forward neural networks. **Neural Processing Letters**, Fevereiro de 2003. 93-105. DOI: https://doi.org/10.1023/A:1022995128597

IVANOVA, A. S. et al. Method for determination of hydrogen peroxide in adulterated milk using high performance liquid chromatography. **Food chemistry**, 15 de Junho de 2019. 431-436. DOI: https://doi.org/10.1016/j.foodchem.2019.01.051

JAIN, A. et al. Artificial neural networks: A tutorial. **Computer**, Março de 1996. 31-44. DOI: https://doi.org/10.1109/2.485891

JENNESS, R. Composition of milk. In: **Fundamentals of dairy chemistry**, 1988. 1-38. Springer, Boston, MA. DOI: https://doi.org/10.1007/978-1-4615-7050-9_1

JONES, G. R. et al. The Gabor transform basis of chromatic monitoring. **Measurement Science and Technology**, Fevereiro de 2000. 489, 2000. DOI: https://doi.org/10.1088/0957-0233/11/5/307

KHAN, K. M. et al. Detection of urea adulteration in milk using near-infrared Raman spectroscopy. **Food analytical methods**, Janeiro de 2015. 93-102. DOI: https://doi.org/10.1007/s12161-014-9873-z

KIM, H. S. et al. A review of piezoelectric energy harvesting based on vibration. **International journal of precision engineering and manufacturing**, 04 de Dezembro de 2011. 1129-1141. DOI: https://doi.org/10.1007/s12541-011-0151-3

KIM, B. e PARK, Y. H. Beginner's guide to neural networks for the MNIST dataset using MATLAB. **The Korean Journal of Mathematics**, 26 de Março de 2018. 337-348. DOI: https://doi.org/10.11568/kjm.2018.26.2.337

KIMURA, M. **Piezo-electricity generation device**. U.S. Patent n. 5,801,475, 1º de Setembro de 1998. Disponível em https://www.freepatentsonline.com/5801475.html. Acesso em: 20 de Agosto de 2020

KRÖSE, B. et al. **An introduction to neural networks.** 1993. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.8890&rep=rep1&type=pdf. Acesso em: 23 de Agosto de 2020

LORD, A. E. Acoustic Emission. **Physical Acoustics**, 1975. 289–353. DOI: https://doi.org/10.1016/B978-0-12-477911-2.50011-9

MATLAB, Version. 7.11.0 (R2010b). **The MathWorks Inc., Natick, Massachusetts**, 2010

MCCULLOCH, W. S. e PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, 1943. 115-133. DOI: https://doi.org/10.1007/BF02478259

MOHAMMADI, V. et al. Ultrasonic techniques for the milk production industry. **Measurement**, Dezembro de 2014. 93-102. DOI: https://doi.org/10.1016/j.measurement.2014.08.022

MOHANAN, S. et al. A new ultrasonic method to detect chemical additives in branded milk. **Pramana**. Setembro de 2002. 525-529. DOI: https://doi.org/10.1007/s12043-002-0049-9

MOODY, J. e DARKEN, C. J. Fast learning in networks of locally-tuned processing units. **Neural computation**, 01 de Junho de 1989. 281-294. DOI: https://doi.org/10.1162/neco.1989.1.2.281

MURATA. Murata Manufacturing Co. Ltd. **Piezoelectric Diaphragms**. Disponível em https://www.murata.com/en-us/products/sound/diaphragm. Acesso em: 18 de Agosto de 2021

Neural Network Toolbox User's Guide, documentação do programa **Matlab R2010b**, 2010

NAZARCHUK, Z. et al. **Acoustic emission. Foundations of Engineering Mechanics**, 2017. Springer Editora, 2017. DOI: 0.1007/978-3-319-49350-3

PAN, G. et al. Bleeding detection in wireless capsule endoscopy based on probabilistic neural network. **Journal of medical systems**, 2011. 1477-1484. DOI: https://doi.org/10.1007/s10916-009-9424-0

PANCHAL, F. S. e PANCHAL, M. Review on methods of selecting number of hidden nodes in artificial neural network. **International Journal of Computer Science and Mobile Computing**, Novembro de 2014. 455-464. Disponível em https://www.ijcsmc.com/docs/papers/November2014/V3I11201499a19.pdf. Acesso em: 10 de Outubro de 2020

PAPOULIS A. e PILLAI S. U. **Probability, Random Variables, and Stochastic Processes**, 2002. 1-850

McGraw-Hill Book Co., New York

RAKHSHANDEHROO, G. et al. Forecasting groundwater level in Shiraz plain using artificial neural networks. **Arabian Journal for Science and Engineering**, 08 de Maio de 2012. 1871-1883. DOI: https://doi.org/10.1007/s13369-012-0291-5

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, 1958. 386-408. DOI: https://doi.org/10.1037/h0042519

SAGE, Andrew P. (Ed.). **Concise Encyclopedia of Information Processing in Systems & [and] Organizations**. 1990. New York, NY: Pergamon Press

SANTOS JUNIOR, M. M.. **Identificação de adulteração de leite bovino pelo método de propagação de onda em meio material utilizando diafragmas piezelétricos**. Dissertação (Mestrado) – Universidade Estadual Paulista. Faculdade de Engenharia, Bauru, SP, 2019. Disponível em https://repositorio.unesp.br/bitstream/handle/11449/191427/santosjunior_mm_me_bauru.pdf?sequence=3&isAllowed=y. Acesso em 21 de Abril de 2020

SHARMA, S. Activation functions in neural networks. **Towards Data Science**, Abril de 2020. 1-7. Disponível em https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf. Acesso em: 14 de Outubro de 2020

SPECHT, D. F. Probabilistic neural networks and the polynomial adaline as complementary techniques for classification. **IEEE Transactions on Neural Networks**, Março de 1990. 111-121. DOI: https://doi.org/10.1109/72.80210

SPECHT, D. F. A general regression neural network. **IEEE transactions on neural networks**, Novembro de 1991. 568-576. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.458.4175&rep=rep1&type=pdf. Acesso em 14 de Dezembro de 2020

STAPLES, C. R. e LOUGH, D. S. Efficacy of supplemental dietary neutralizing agents for lactating dairy cows. A review. **Animal Feed Science and Technology**, Maio de 1989. 277-303. DOI: https://doi.org/10.1016/0377-8401(89)90050-3

STATHAKIS, D. How many hidden layers and nodes? **International Journal of Remote Sensing**, Abril de 2009. 2133-2147. DOI: http://dx.doi.org/10.1080/01431160802549278

SVOZIL, D. et al. Introduction to multi-layer feed-forward neural networks. **Chemometrics and intelligent laboratory systems**, Junho de 1997. 43-62. DOI: https://doi.org/10.1016/S0169-7439(97)00061-0

TOKO, K. RETRACTED: Electronic tongue. **Biosensors and Bioelectronics**. 1998. 701–709. DOI: https://doi.org/10.1016/S0956-5663(98)00025-6

TRIVEDI, U. B. et al. Potentiometric biosensor for urea determination in milk. **Sensors and Actuators B: Chemical**, Junho de 2009. 260-266. DOI: https://doi.org/10.1016/j.snb.2009.04.022

VAN DER WESTHUYZEN, P. J. **Probe characterisation, design and evaluation for the real-time quality Indication of milk**. Tese (Doutorado) – Stellenbosch: University of Stellenbosch. Department of Electrical and Electronic Engineering, Matieland, South Africa, 2006. Disponível em: http://scholar.sun.ac.za/bitstream/handle/10019.1/1804/vanderwesthuyzen_probe_2006.pdf. Acesso em: 22 de Agosto de 2021

VON ZUBEN, F. J. e DE CASTRO, L. N. Redes Neurais. **Notas de aula, FEEC, Universidade Estadual de Campinas**, 2003. Disponível em ftp://calhau.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_03/revisao/docs/tema25.pdf. Acesso em 13 de Dezembro de 2020

WARSITO, B. et al. Cascade forward neural network for time series prediction. In: **Journal of Physics: Conference Series**. IOP Publishing, 2018. 1-9. DOI: https://doi.org/10.1088/1742-6596/1025/1/012097

WASSERMAN, P. D. **Advanced methods in neural computing**. John Wiley & Sons, Inc., 1993.

WEISS, L. G. Wavelets and wideband correlation processing. **IEEE signal processing magazine**, 1994. 13-32. DOI: https://doi.org/10.1109/79.252866

WERBOS, Paul J. Backpropagation through time: what it does and how to do it. **Proceedings of the IEEE**, Outubro de 1990. 1550-1560. DOI: https://doi.org/10.1109/5.58337

YOICHI, M. Applications of piezoelectric actuator. **NEC Technical Journal**, 2006. 82-86. Disponível em https://www.nec.com/en/global/techrep/journal/g06/n05/pdf/t060519.pdf. Acesso em: 29 de Agosto de 2020

YU, H. e WILAMOWSKI, B. M. Levenberg-marquardt training. **Industrial electronics handbook**, 2011. 12.1-16. Disponível em http://www.eng.auburn.edu/~wilambm/pap/2011/K10149_C012.pdf. Acesso em: 16 de Outubro de 2020

YUAN, B. Efficient hardware architecture of softmax layer in deep neural network. In: **2016 29th IEEE International System-on-Chip Conference (SOCC)**. IEEE, 2016. 323-326. DOI: https://doi.org/10.1109/SOCC.2016.7905501

ZHANG, J. et al. Chromatic classification of RF signals produced by electrical discharges in HV transformers. **IEE Proceedings-Generation, Transmission and Distribution**, Setembro de 2005. 629-634. DOI: https://doi.org/10.1049/ip-gtd:20045076

Apêndice A

Esta seção apresenta os *scripts* criados no MATLAB para realização dos ensaios das RNAs. O item A.1 é comum para todas as redes, sendo o carregamento dos dados.

A.1 Carregamento e condicionamento dos dados

```
% Preparação do ambiente
clc; clear all; close all;


%% Leitura dos arquivos de amostras e preparação
load('dadosEnsaio160519_passa_alta_400hz_tc.mat');

puro1(1:length(puro_1)) = 1; puro_1_completo = [puro_1 puro1']; % Leite puro (amostra 1)
puro2(1:length(puro_2)) = 1; puro_2_completo = [puro_2 puro2']; % Leite puro (amostra 2)
meiopcbs1(1:length(meio_pc_bs_1)) = 2; meio_pc_bs_1_completo = [meio_pc_bs_1 meiopcbs1']; % Leite contaminado com 0,5 % de Bicarbonato de Sódio (amostra 1)
meiopcbs2(1:length(meio_pc_bs_2)) = 2; meio_pc_bs_2_completo = [meio_pc_bs_2 meiopcbs2']; % Leite contaminado com 0,5 % de Bicarbonato de Sódio (amostra 2)
umpcbs1(1:length(um_pc_bs_1)) = 3; um_pc_bs_1_completo = [um_pc_bs_1 umpcbs1']; % Leite contaminado com 1 % de Bicarbonato de Sódio (amostra 1)
umpcbs2(1:length(um_pc_bs_2)) = 3; um_pc_bs_2_completo = [um_pc_bs_2 umpcbs2']; % Leite contaminado com 1 % de Bicarbonato de Sódio (amostra 2)
meiopcureia1(1:length(meio_pc_ureia_1)) = 4; meio_pc_ureia_1_completo = [meio_pc_ureia_1
meiopcureia1']; % Leite contaminado com 0,5 % de Ureia (amostra 1)
meiopcureia2(1:length(meio_pc_ureia_2)) = 4; meio_pc_ureia_2_completo = [meio_pc_ureia_2
meiopcureia2']; % Leite contaminado com 0,5 % de Ureia (amostra 2)
umpcureia1(1:length(um_pc_ureia_1)) = 5; um_pc_ureia_1_completo = [um_pc_ureia_1 umpcureia1']; % Leite contaminado com 1 % de Ureia (amostra 1)
umpcureia2(1:length(um_pc_ureia_2)) = 5; um_pc_ureia_2_completo = [um_pc_ureia_2 umpcureia2']; % Leite contaminado com 1 % de Ureia (amostra 2)
meiopcao1(1:length(meio_pc_ao_1)) = 6; meio_pc_ao_1_completo = [meio_pc_ao_1 meiopcao1']; % Leite contaminado com 0,5 % de Água Oxigenada (amostra 1)
meiopcao2(1:length(meio_pc_ao_2)) = 6; meio_pc_ao_2_completo = [meio_pc_ao_2 meiopcao2']; % Leite contaminado com 0,5 % de Água Oxigenada (amostra 2)
umpcao1(1:length(um_pc_ao_1)) = 7; um_pc_ao_1_completo = [um_pc_ao_1 umpcao1']; % Leite contaminado com 1 % de Água Oxigenada (amostra 1)
umpcao2(1:length(um_pc_ao_2)) = 7; um_pc_ao_2_completo = [um_pc_ao_2 umpcao2']; % Leite contaminado
```

```
com 1 % de Água Oxigenada (amostra 2)

Dados1 = [puro_1_completo; meio_pc_bs_1_completo; um_pc_bs_1_completo; meio_pc_ureia_1_completo;
um_pc_ureia_1_completo; meio_pc_ao_1_completo; um_pc_ao_1_completo];
Dados2 = [puro_2_completo; meio_pc_bs_2_completo; um_pc_bs_2_completo; meio_pc_ureia_2_completo;
um_pc_ureia_2_completo; meio_pc_ao_2_completo; um_pc_ao_2_completo];
Dados = [puro_1_completo; puro_2_completo; meio_pc_bs_1_completo; meio_pc_bs_2_completo;
um_pc_bs_1_completo; um_pc_bs_2_completo; meio_pc_ureia_1_completo; meio_pc_ureia_2_completo;
um_pc_ureia_1_completo; um_pc_ureia_2_completo; meio_pc_ao_1_completo; meio_pc_ao_2_completo;
um_pc_ao_1_completo; um_pc_ao_2_completo];
```

A.1.1 Rede Perceptron Multicamadas

```
% Definição dos dados
x = Dados(:,1:3); % Entrada => energia, banda média e largura de banda equivalente
d = Dados(:,4); % Saída desejada

%% Seleção de parâmetro para o ensaio
% N = [3]; NumeroEnsaio = 1; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [5]; NumeroEnsaio = 2; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [10]; NumeroEnsaio = 3; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15]; NumeroEnsaio = 4; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [30]; NumeroEnsaio = 5; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [10 5]; NumeroEnsaio = 6; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15 10]; NumeroEnsaio = 7; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [20 15]; NumeroEnsaio = 8; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15 10 5]; NumeroEnsaio = 9; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [20 15 10]; NumeroEnsaio = 10; % Quantidade de neurônios e camadas escondidas / Número do ensaio

for j = 1:10 % Número de execuções da rede
%% Carregamento de RNA
net = feedforwardnet(N); % Rede Perceptron Multicamadas Feed-Forward
net.trainFcn = 'trainlm'; % Algoritmo de aprendizado
net.divideFcn = 'divideint'; % Divisão de dados para treinamento e para validação cruzada
net.divideParam.trainRatio = 0.5; % Porcentagem de treinamento
net.divideParam.valRatio = 0.0; % Porcentagem de validação
net.divideParam.testRatio = 0.5; % Porcentagem de teste
net.trainParam.epochs = 2000; % Número máximo de épocas de treinamento
net.trainParam.goal = 1e-7; % Erro desejado
net.trainParam.time = inf; % Tempo máximo para o treinamento, em segundos
```

```
 net = init(net); % Inicializa a rede com pesos diferentes
[net, tr] = train(net,x',d'); % Treinamento da rede
y = net(x'); % Executa a simulação da rede

%% Regressão dos dados de treinamento, validação e teste
DadosEntrada = x'; % Dados de entrada
DadosAlvos = d'; % Dados alvos (saídas desejadas)

DadosSaidas = net(DadosEntrada); % Dados de saídas calculados pela rede
trOut = DadosSaidas(tr.trainInd); % Vetor de saídas retornadas do treinamento
vOut = DadosSaidas(tr.valInd); % Vetor de saídas retornadas da validação
tsOut = DadosSaidas(tr.testInd); % Vetor de saídas retornadas do teste
trTarg = DadosAlvos(tr.trainInd); % Vetor de alvos utilizados no treinamento
vTarg = DadosAlvos(tr.valInd); % Vetor de alvos utilizados na validação
tsTarg = DadosAlvos(tr.testInd); % Vetor de alvos utilizados no teste

figure; % Cria janela com figura para apresentar a regressão
Regressao{j} = plotregression(trTarg, trOut, 'Treinamento', vTarg, vOut, 'Validacao', tsTarg, tsOut, 'Teste'); %
Gráficos de regressão individual para cada execução de rede

trTargs{j} = [trTarg]; % Célula de vetor(es) de alvos utilizados no treinamento
trOuts{j} = [trOut]; % Célula de vetor(es) de saídas retornadas do treinamento
vTargs{j} = [vTarg]; % Célula de vetor(es) de alvos utilizados na validação
vOuts{j} = [vOut]; % Célula de vetor(es) de saídas retornadas da validação
tsTargs{j} = [tsTarg]; % Célula de vetor(es) de alvos utilizados no teste
tsOuts{j} = [tsOut]; % Célula de vetor(es) de saídas retornadas do teste

% Separando todos os dados das regressões para criar regressão geral do Ensaio
trTamanho = length(trOut); % Tamanho do vetor utilizado no treinamento
vTamanho = length(vOut); % Tamanho do vetor utilizado na validação
tsTamanho = length(tsOut); % Tamanho do vetor utilizado no teste

trTargsLinha = cell2mat(trTargs); trTargsMatriz = reshape(trTargsLinha, [], j); % Organiza matriz de alvos do
treinamento
trOutsLinha = cell2mat(trOuts); trOutsMatriz = reshape(trOutsLinha, [], j); % Organiza matriz de saídas do
treinamento
vTargsLinha = cell2mat(vTargs); vTargsMatriz = reshape(vTargsLinha, [], j); % Organiza matriz de alvos da
validação
vOutsLinha = cell2mat(vOuts); vOutsMatriz = reshape(vOutsLinha, [], j); % Organiza matriz de saídas da
validação
```

```
tsTargsLinha = cell2mat(tsTargs); tsTargsMatriz = reshape(tsTargsLinha, [], j); % Organiza matriz de alvos do teste
tsOutsLinha = cell2mat(tsOuts); tsOutsMatriz = reshape(tsOutsLinha, [], j); % Organiza matriz de saídas do teste

trTargsMatrizMedia = mean(trTargsMatriz,2); % Calcula um vetor médio da matriz de alvos do treinamento
trOutsMatrizMedia = mean(trOutsMatriz,2); % Calcula um vetor médio da matriz de saídas do treinamento
vTargsMatrizMedia = mean(vTargsMatriz,2); % Calcula um vetor médio da matriz de alvos da validação
vOutsMatrizMedia = mean(vOutsMatriz,2); % Calcula um vetor médio da matriz de saídas da validação
tsTargsMatrizMedia = mean(tsTargsMatriz,2); % Calcula um vetor médio da matriz de alvos do teste
tsOutsMatrizMedia = mean(tsOutsMatriz,2); % Calcula um vetor médio da matriz de saídas do teste

% Coeficiente de regressão 'R'
trCyt = corrcoef(trTarg', trOut'); trR{j} = trCyt(2,1); % Coeficiente 'R' do treinamento
vCyt = corrcoef(vTarg', vOut'); vR{j} = vCyt(2,1); % Coeficiente 'R' da validação
tsCyt = corrcoef(tsTarg', tsOut'); tsR{j} = tsCyt(2,1); % Coeficiente 'R' do teste

tsOutArr = round(tsOut); % Arredondamento do vetor de saída utilizado no teste para facilitar comparação
Diferenca = tsTarg' - tsOutArr'; % Calcula a diferença entre o vetor de alvo para o vetor de saída do teste

% Erro Quadrático Médio
MSEtr{j} = mean((trTarg - trOut).^2); % EQM do treinamento
MSEv{j} = mean((vTarg - vOut).^2); % EQM da validação
MSEts{j} = mean((tsTarg - tsOut).^2); % EQM do teste

Comparacao{j} = [tsTarg' tsOutArr' Diferenca]; % Matriz para comparar saída, alvo e a diferença entre ambos

QtdAcertos{j} = sum(Diferenca == 0); % Conta quantos acertos a rede conseguiu obter
PercentualAcertos = (100*(QtdAcertos{j}))/(length(Comparacao{j})); % Calcula o percentual de acertos
PercentuaisAcertos{j} = PercentualAcertos; % Célula de vetor(es) de percentual(is) de acertos
Ns{j} = N; % Célula de vetor(es) do(s) parâmetro(s) principal(is) da rede para exibição
tsTamanhos{j} = tsTamanho; % Célula de vetor(es) da(s) quantidade(s) de elementos usados no teste da rede, para exibição

%% Organização dos resultados para exibição final
Resumo1 = [tsTarg' tsOutArr' Diferenca]; % Matriz para resumir a comparação entre alvo, saída e diferença entre elas
Tabela1{j} = array2table(Resumo1,'VariableNames',{'Alvo','Resultado','Diferenca'}); % Tabela para resumir a comparação entre alvo, saída e diferença entre elas para todos os ciclos da rede
Resumo2 = [Ns' QtdAcertos' tsTamanhos' PercentuaisAcertos' MSEtr' MSEv' MSEts' trR' vR' tsR']; % Matriz
```

```
com informações gerais das execuções das redes para comparação
Tabela2 =
array2table(Resumo2,'VariableNames',{'Parametro','Quantidade_Acertos','Total','Percentual_Acertos','MSE_Tre
inamento','MSE_Validacao','MSE_Teste','Regressao_Treinamento','Regressao_Validacao','Regressao_Teste'});
% Tabela com informações gerais das execuções das redes para comparação
end;
```

A.1.2 Rede de Retropropagação em Cascata

```
% Definição dos dados
x = Dados(:,1:3); % Entrada => energia, banda média e largura de banda equivalente
d = Dados(:,4); % Saída desejada

%% Seleção de parâmetro para o ensaio
% N = [3]; NumeroEnsaio = 1; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [5]; NumeroEnsaio = 2; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [10]; NumeroEnsaio = 3; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15]; NumeroEnsaio = 4; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [30]; NumeroEnsaio = 5; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [10 5]; NumeroEnsaio = 6; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15 10]; NumeroEnsaio = 7; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [20 15]; NumeroEnsaio = 8; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15 10 5]; NumeroEnsaio = 9; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [20 15 10]; NumeroEnsaio = 10; % Quantidade de neurônios e camadas escondidas / Número do ensaio

for j = 1:10 % Número de execuções da rede
%% Carregamento de RNA
net = newcf(x',d',N); % Rede Neural de Retropropagação em Cascata
net.trainFcn = 'trainlm'; % Algoritmo de aprendizado
net.divideFcn = 'divideint'; % Divisão de dados para treinamento e para validação cruzada
net.divideParam.trainRatio = 0.5; % Porcentagem de treinamento
net.divideParam.valRatio = 0.0; % Porcentagem de validação
net.divideParam.testRatio = 0.5; % Porcentagem de teste
net.trainParam.epochs = 2000; % Número máximo de épocas de treinamento
net.trainParam.goal = 1e-7; % Erro desejado
net.trainParam.time = inf; % Tempo máximo para o treinamento, em segundos

net = init(net); % Inicializa a rede com pesos diferentes
[net, tr] = train(net,x',d'); % Treinamento da rede
y = net(x'); % Executa a simulação da rede
```

```
%% Regressão dos dados de treinamento, validação e teste
DadosEntrada = x'; % Dados de entrada
DadosAlvos = d'; % Dados alvos (saídas desejadas)

DadosSaidas = net(DadosEntrada); % Dados de saídas calculados pela rede
trOut = DadosSaidas(tr.trainInd); % Vetor de saídas retornadas do treinamento
vOut = DadosSaidas(tr.valInd); % Vetor de saídas retornadas da validação
tsOut = DadosSaidas(tr.testInd); % Vetor de saídas retornadas do teste
trTarg = DadosAlvos(tr.trainInd); % Vetor de alvos utilizados no treinamento
vTarg = DadosAlvos(tr.valInd); % Vetor de alvos utilizados na validação
tsTarg = DadosAlvos(tr.testInd); % Vetor de alvos utilizados no teste

figure; % Cria janela com figura para apresentar a regressão
Regressao{j} = plotregression(trTarg, trOut, 'Treinamento', vTarg, vOut, 'Validacao', tsTarg, tsOut, 'Teste'); %
Gráficos de regressão individual para cada execução de rede

trTargs{j} = [trTarg]; % Célula de vetor(es) de alvos utilizados no treinamento
trOuts{j} = [trOut]; % Célula de vetor(es) de saídas retornadas do treinamento
vTargs{j} = [vTarg]; % Célula de vetor(es) de alvos utilizados na validação
vOuts{j} = [vOut]; % Célula de vetor(es) de saídas retornadas da validação
tsTargs{j} = [tsTarg]; % Célula de vetor(es) de alvos utilizados no teste
tsOuts{j} = [tsOut]; % Célula de vetor(es) de saídas retornadas do teste

% Separando todos os dados das regressões para criar regressão geral do Ensaio
trTamanho = length(trOut); % Tamanho do vetor utilizado no treinamento
vTamanho = length(vOut); % Tamanho do vetor utilizado na validação
tsTamanho = length(tsOut); % Tamanho do vetor utilizado no teste

trTargsLinha = cell2mat(trTargs); trTargsMatriz = reshape(trTargsLinha, [], j); % Organiza matriz de alvos do
treinamento
trOutsLinha = cell2mat(trOuts); trOutsMatriz = reshape(trOutsLinha, [], j); % Organiza matriz de saídas do
treinamento
vTargsLinha = cell2mat(vTargs); vTargsMatriz = reshape(vTargsLinha, [], j); % Organiza matriz de alvos da
validação
vOutsLinha = cell2mat(vOuts); vOutsMatriz = reshape(vOutsLinha, [], j); % Organiza matriz de saídas da
validação
tsTargsLinha = cell2mat(tsTargs); tsTargsMatriz = reshape(tsTargsLinha, [], j); % Organiza matriz de alvos do
teste
tsOutsLinha = cell2mat(tsOuts); tsOutsMatriz = reshape(tsOutsLinha, [], j); % Organiza matriz de saídas do
teste
```

```
trTargsMatrizMedia = mean(trTargsMatriz,2); % Calcula um vetor médio da matriz de alvos do treinamento
trOutsMatrizMedia = mean(trOutsMatriz,2); % Calcula um vetor médio da matriz de saídas do treinamento
vTargsMatrizMedia = mean(vTargsMatriz,2); % Calcula um vetor médio da matriz de alvos da validação
vOutsMatrizMedia = mean(vOutsMatriz,2); % Calcula um vetor médio da matriz de saídas da validação
tsTargsMatrizMedia = mean(tsTargsMatriz,2); % Calcula um vetor médio da matriz de alvos do teste
tsOutsMatrizMedia = mean(tsOutsMatriz,2); % Calcula um vetor médio da matriz de saídas do teste

% Coeficiente de regressão 'R'
trCyt = corrcoef(trTarg', trOut'); trR{j} = trCyt(2,1); % Coeficiente 'R' do treinamento
vCyt = corrcoef(vTarg', vOut'); vR{j} = vCyt(2,1); % Coeficiente 'R' da validação
tsCyt = corrcoef(tsTarg', tsOut'); tsR{j} = tsCyt(2,1); % Coeficiente 'R' do teste

tsOutArr = round(tsOut); % Arredondamento do vetor de saída utilizado no teste para facilitar comparação
Diferenca = tsTarg' - tsOutArr'; % Calcula a diferença entre o vetor de alvo para o vetor de saída do teste

% Erro Quadrático Médio
MSEtr{j} = mean((trTarg - trOut).^2); % EQM do treinamento
MSEv{j} = mean((vTarg - vOut).^2); % EQM da validação
MSEts{j} = mean((tsTarg - tsOut).^2); % EQM do teste

Comparacao{j} = [tsTarg' tsOutArr' Diferenca]; % Matriz para comparar saída, alvo e a diferença entre ambos

QtdAcertos{j} = sum(Diferenca == 0); % Conta quantos acertos a rede conseguiu obter
PercentualAcertos = (100*(QtdAcertos{j}))/(length(Comparacao{j})); % Calcula o percentual de acertos
PercentuaisAcertos{j} = PercentualAcertos; % Célula de vetor(es) de percentual(is) de acertos
Ns{j} = N; % Célula de vetor(es) do(s) parâmetro(s) principal(is) da rede para exibição
tsTamanhos{j} = tsTamanho; % Célula de vetor(es) da(s) quantidade(s) de elementos usados no teste da rede,
para exibição

%% Organização dos resultados para exibição final
Resumo1 = [tsTarg' tsOutArr' Diferenca]; % Matriz para resumir a comparação entre alvo, saída e diferença
entre elas
Tabela1{j} = array2table(Resumo1,'VariableNames',{'Alvo','Resultado','Diferenca'}); % Tabela para resumir a
comparação entre alvo, saída e diferença entre elas para todos os ciclos da rede
Resumo2 = [Ns' QtdAcertos' tsTamanhos' PercentuaisAcertos' MSEtr' MSEv' MSEts' trR' vR' tsR']; % Matriz
com informações gerais das execuções das redes para comparação
Tabela2 =
array2table(Resumo2,'VariableNames',{'Parametro','Quantidade_Acertos','Total','Percentual_Acertos','MSE_Tre
inamento','MSE_Validacao','MSE_Teste','Regressao_Treinamento','Regressao_Validacao','Regressao_Teste'});
% Tabela com informações gerais das execuções das redes para comparação
```

```
end;

%% Armazenamento em arquivos
SalvaArquivo = sprintf('redenewcf_AreaDeTrabalho_Ensaio_%d.mat',NumeroEnsaio); save(SalvaArquivo); % Salva arquivo com todas as variáveis da área de trabalho
writetable(Tabela2,sprintf('redenewcf_TabelaResumo_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de resultados finais
figure; % Cria janela com figura para apresentar a regressão média de todas os ciclos das redes
RegressaoMedia = plotregression(trTargsMatrizMedia', trOutsMatrizMedia', 'Média Trein.', vTargsMatrizMedia', vOutsMatrizMedia', 'Média Valid.', tsTargsMatrizMedia', tsOutsMatrizMedia', 'Média Teste'); % Regressão média de todas os ciclos da rede
NomeArquivoFigura = sprintf('redenewcf_RegressaoMedia_Ensaio_%d',NumeroEnsaio); % Cria nome para salvar em arquivo a figura da regressão média dos ciclos da rede
saveas(gca,NomeArquivoFigura,'jpeg'); % Salva figura de regressão média dos ciclos da rede em formato '.jpeg'

%% Criação de gráfico comparativo "Alvos X Resultados" para a melhor parametrização da RNA
figure; % Cria janela com figura para apresentar a relação de alvos X resultados
plot(tsTarg,'r'); % Traça as amostras no gráfico, na cor vermelha
xlabel('Índices das Amostras'); % Rotula o eixo x: "Índices das Amostras"
axis([0 tsTamanho 0 8]); % Define os tamanhos máximos dos eixos "x" e "y" do gráfico
Escala = tsTamanho/7; % Calcula a escala de exibição do eixo "x" de acordo com a quantidade de amostras
xticks(0:Escala:tsTamanho); % Define o tamanho do eixo "x" com sua escala
ylabel('Amostras'); % Rotula o eixo y: "Amostras"
hold on; % "Trava" o gráfico para inserir o tracejado dos resultados
plot(tsOutsMatrizMedia,'b'); % Traça os resultados no gráfico, na cor azul
legend('Alvo','Resultado','location','nw'); % Define os itens da legenda e posiciona no canto superior esquerdo
print -dpng GraficoMelhorParametrizacao_redenewcf.png % Armazena o gráfico pronto em arquivo
```

A.1.3 Rede Elman de Retropropagação

```
% Definição dos dados
x = Dados(:,1:3); % Entrada => energia, banda média e largura de banda equivalente
d = Dados(:,4); % Saída desejada

%% Seleção de parâmetro para o ensaio
% N = [3]; NumeroEnsaio = 1; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [5]; NumeroEnsaio = 2; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [10]; NumeroEnsaio = 3; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15]; NumeroEnsaio = 4; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [30]; NumeroEnsaio = 5; % Quantidade de neurônios e camadas escondidas / Número do ensaio
```

```
% N = [10 5]; NumeroEnsaio = 6; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15 10]; NumeroEnsaio = 7; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [20 15]; NumeroEnsaio = 8; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15 10 5]; NumeroEnsaio = 9; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [20 15 10]; NumeroEnsaio = 10; % Quantidade de neurônios e camadas escondidas / Número do ensaio

for j = 1:10 % Número de execuções da rede
%% Carregamento de RNA
net = newelm(x',d',N); % Rede Elman de Retropropagação
net.trainFcn = 'trainlm'; % Algoritmo de aprendizado
net.divideFcn = 'divideint'; % Divisão de dados para treinamento e para validação cruzada
net.divideParam.trainRatio = 0.5; % Porcentagem de treinamento
net.divideParam.valRatio = 0.0; % Porcentagem de validação
net.divideParam.testRatio = 0.5; % Porcentagem de teste
net.trainParam.epochs = 2000; % Número máximo de épocas de treinamento
net.trainParam.goal = 1e-7; % Erro desejado
net.trainParam.time = inf; % Tempo máximo para o treinamento, em segundos

net = init(net); % Inicializa a rede com pesos diferentes
[net, tr] = train(net,x',d'); % Treinamento da rede
y = net(x'); % Executa a simulação da rede

%% Regressão dos dados de treinamento, validação e teste
DadosEntrada = x'; % Dados de entrada
DadosAlvos = d'; % Dados alvos (saídas desejadas)

DadosSaidas = net(DadosEntrada); % Dados de saídas calculados pela rede
trOut = DadosSaidas(tr.trainInd); % Vetor de saídas retornadas do treinamento
vOut = DadosSaidas(tr.valInd); % Vetor de saídas retornadas da validação
tsOut = DadosSaidas(tr.testInd); % Vetor de saídas retornadas do teste
trTarg = DadosAlvos(tr.trainInd); % Vetor de alvos utilizados no treinamento
vTarg = DadosAlvos(tr.valInd); % Vetor de alvos utilizados na validação
tsTarg = DadosAlvos(tr.testInd); % Vetor de alvos utilizados no teste

figure; % Cria janela com figura para apresentar a regressão
Regressao{j} = plotregression(trTarg, trOut, 'Treinamento', vTarg, vOut, 'Validacao', tsTarg, tsOut, 'Teste'); %
Gráficos de regressão individual para cada execução de rede

trTargs{j} = [trTarg]; % Célula de vetor(es) de alvos utilizados no treinamento
trOuts{j} = [trOut]; % Célula de vetor(es) de saídas retornadas do treinamento
```

```
vTargs{j} = [vTarg]; % Célula de vetor(es) de alvos utilizados na validação
vOuts{j} = [vOut]; % Célula de vetor(es) de saídas retornadas da validação
tsTargs{j} = [tsTarg]; % Célula de vetor(es) de alvos utilizados no teste
tsOuts{j} = [tsOut]; % Célula de vetor(es) de saídas retornadas do teste

% Separando todos os dados das regressões para criar regressão geral do Ensaio
trTamanho = length(trOut); % Tamanho do vetor utilizado no treinamento
vTamanho = length(vOut); % Tamanho do vetor utilizado na validação
tsTamanho = length(tsOut); % Tamanho do vetor utilizado no teste

trTargsLinha = cell2mat(trTargs); trTargsMatriz = reshape(trTargsLinha, [], j); % Organiza matriz de alvos do treinamento
trOutsLinha = cell2mat(trOuts); trOutsMatriz = reshape(trOutsLinha, [], j); % Organiza matriz de saídas do treinamento
vTargsLinha = cell2mat(vTargs); vTargsMatriz = reshape(vTargsLinha, [], j); % Organiza matriz de alvos da validação
vOutsLinha = cell2mat(vOuts); vOutsMatriz = reshape(vOutsLinha, [], j); % Organiza matriz de saídas da validação
tsTargsLinha = cell2mat(tsTargs); tsTargsMatriz = reshape(tsTargsLinha, [], j); % Organiza matriz de alvos do teste
tsOutsLinha = cell2mat(tsOuts); tsOutsMatriz = reshape(tsOutsLinha, [], j); % Organiza matriz de saídas do teste

trTargsMatrizMedia = mean(trTargsMatriz,2); % Calcula um vetor médio da matriz de alvos do treinamento
trOutsMatrizMedia = mean(trOutsMatriz,2); % Calcula um vetor médio da matriz de saídas do treinamento
vTargsMatrizMedia = mean(vTargsMatriz,2); % Calcula um vetor médio da matriz de alvos da validação
vOutsMatrizMedia = mean(vOutsMatriz,2); % Calcula um vetor médio da matriz de saídas da validação
tsTargsMatrizMedia = mean(tsTargsMatriz,2); % Calcula um vetor médio da matriz de alvos do teste
tsOutsMatrizMedia = mean(tsOutsMatriz,2); % Calcula um vetor médio da matriz de saídas do teste

% Coeficiente de regressão 'R'
trCyt = corrcoef(trTarg', trOut'); trR{j} = trCyt(2,1); % Coeficiente 'R' do treinamento
vCyt = corrcoef(vTarg', vOut'); vR{j} = vCyt(2,1); % Coeficiente 'R' da validação
tsCyt = corrcoef(tsTarg', tsOut'); tsR{j} = tsCyt(2,1); % Coeficiente 'R' do teste

tsOutArr = round(tsOut); % Arredondamento do vetor de saída utilizado no teste para facilitar comparação
Diferenca = tsTarg' - tsOutArr'; % Calcula a diferença entre o vetor de alvo para o vetor de saída do teste

% Erro Quadrático Médio
MSEtr{j} = mean((trTarg - trOut).^2); % EQM do treinamento
```

```
MSEv{j} = mean((vTarg - vOut).^2); % EQM da validação
MSEts{j} = mean((tsTarg - tsOut).^2); % EQM do teste

Comparacao{j} = [tsTarg' tsOutArr' Diferenca]; % Matriz para comparar saída, alvo e a diferença entre ambos

QtdAcertos{j} = sum(Diferenca == 0); % Conta quantos acertos a rede conseguiu obter
PercentualAcertos = (100*(QtdAcertos{j}))/(length(Comparacao{j})); % Calcula o percentual de acertos
PercentuaisAcertos{j} = PercentualAcertos; % Célula de vetor(es) de percentual(is) de acertos
Ns{j} = N; % Célula de vetor(es) do(s) parâmetro(s) principal(is) da rede para exibição
tsTamanhos{j} = tsTamanho; % Célula de vetor(es) da(s) quantidade(s) de elementos usados no teste da rede, para exibição

%% Organização dos resultados para exibição final
Resumo1 = [tsTarg' tsOutArr' Diferenca]; % Matriz para resumir a comparação entre alvo, saída e diferença entre elas
Tabela1{j} = array2table(Resumo1,'VariableNames',{'Alvo','Resultado','Diferenca'}); % Tabela para resumir a comparação entre alvo, saída e diferença entre elas para todos os ciclos da rede
Resumo2 = [Ns' QtdAcertos' tsTamanhos' PercentuaisAcertos' MSEtr' MSEv' MSEts' trR' vR' tsR']; % Matriz com informações gerais das execuções das redes para comparação
Tabela2 =
array2table(Resumo2,'VariableNames',{'Parametro','Quantidade_Acertos','Total','Percentual_Acertos','MSE_Treinamento','MSE_Validacao','MSE_Teste','Regressao_Treinamento','Regressao_Validacao','Regressao_Teste'});
% Tabela com informações gerais das execuções das redes para comparação
end;

%% Armazenamento em arquivos
SalvaArquivo = sprintf('redenewelm_AreaDeTrabalho_Ensaio_%d.mat',NumeroEnsaio);
save(SalvaArquivo); % Salva arquivo com todas as variáveis da área de trabalho
writetable(Tabela2,sprintf('redenewelm_TabelaResumo_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de resultados finais
figure; % Cria janela com figura para apresentar a regressão média de todas os ciclos das redes
RegressaoMedia = plotregression(trTargsMatrizMedia', trOutsMatrizMedia', 'Média Trein.',
vTargsMatrizMedia', vOutsMatrizMedia', 'Média Valid.', tsTargsMatrizMedia', tsOutsMatrizMedia', 'Média Teste'); % Regressão média de todas os ciclos da rede
NomeArquivoFigura = sprintf('redenewelm_RegressaoMedia_Ensaio_%d',NumeroEnsaio); % Cria nome para salvar em arquivo a figura da regressão média dos ciclos da rede
saveas(gca,NomeArquivoFigura,'jpeg'); % Salva figura de regressão média dos ciclos da rede em formato '.jpeg'

%% Criação de gráfico comparativo "Alvos X Resultados" para a melhor parametrização da RNA
figure; % Cria janela com figura para apresentar a relação de alvos X resultados
```

```
plot(tsTarg,'r'); % Traça as amostras no gráfico, na cor vermelha
xlabel('Índices das Amostras'); % Rotula o eixo x: "Índices das Amostras"
axis([0 tsTamanho 0 8]); % Define os tamanhos máximos dos eixos "x" e "y" do gráfico
Escala = tsTamanho/7; % Calcula a escala de exibição do eixo "x" de acordo com a quantidade de amostras
xticks(0:Escala:tsTamanho); % Define o tamanho do eixo "x" com sua escala
ylabel('Amostras'); % Rotula o eixo y: "Amostras"
hold on; % "Trava" o gráfico para inserir o tracejado dos resultados
plot(tsOutsMatrizMedia,'b'); % Traça os resultados no gráfico, na cor azul
legend('Alvo','Resultado','location','nw'); % Define os itens da legenda e posiciona no canto superior esquerdo
print -dpng GraficoMelhorParametrizacao_redenewelm.png % Armazena o gráfico pronto em arquivo
```

A.1.4 Rede de Reconhecimento de Padrões

```
% Definição dos dados
x = Dados(:,1:3); % Entrada => energia, banda média e largura de banda equivalente
d = Dados(:,4); % Saída desejada

%% Seleção de parâmetro para o ensaio
% N = [3]; NumeroEnsaio = 1; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [5]; NumeroEnsaio = 2; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [10]; NumeroEnsaio = 3; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15]; NumeroEnsaio = 4; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [30]; NumeroEnsaio = 5; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [10 5]; NumeroEnsaio = 6; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15 10]; NumeroEnsaio = 7; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [20 15]; NumeroEnsaio = 8; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [15 10 5]; NumeroEnsaio = 9; % Quantidade de neurônios e camadas escondidas / Número do ensaio
% N = [20 15 10]; NumeroEnsaio = 10; % Quantidade de neurônios e camadas escondidas / Número do ensaio

for j = 1:10 % Número de execuções da rede
%% Carregamento de RNA
net = patternnet(N); % Rede de Reconhecimento de Padrões
net.trainFcn = 'trainlm'; % Algoritmo de aprendizado
net.divideFcn = 'divideint'; % Divisão de dados para treinamento e para validação cruzada
net.divideParam.trainRatio = 0.5; % Porcentagem de treinamento
net.divideParam.valRatio = 0.0; % Porcentagem de validação
net.divideParam.testRatio = 0.5; % Porcentagem de teste
net.trainParam.epochs = 2000; % Número máximo de épocas de treinamento
net.trainParam.goal = 1e-7; % Erro desejado
net.trainParam.time = inf; % Tempo máximo para o treinamento, em segundos
```

```
net = init(net); % Inicializa a rede com pesos diferentes
[net, tr] = train(net,x',d'); % Treinamento da rede
y = net(x'); % Executa a simulação da rede

%% Regressão dos dados de treinamento, validação e teste
DadosEntrada = x'; % Dados de entrada
DadosAlvos = d'; % Dados alvos (saídas desejadas)

DadosSaidas = net(DadosEntrada); % Dados de saídas calculados pela rede
trOut = DadosSaidas(tr.trainInd); % Vetor de saídas retornadas do treinamento
vOut = DadosSaidas(tr.valInd); % Vetor de saídas retornadas da validação
tsOut = DadosSaidas(tr.testInd); % Vetor de saídas retornadas do teste
trTarg = DadosAlvos(tr.trainInd); % Vetor de alvos utilizados no treinamento
vTarg = DadosAlvos(tr.valInd); % Vetor de alvos utilizados na validação
tsTarg = DadosAlvos(tr.testInd); % Vetor de alvos utilizados no teste

figure; % Cria janela com figura para apresentar a regressão
Regressao{j} = plotregression(trTarg, trOut, 'Treinamento', vTarg, vOut, 'Validacao', tsTarg, tsOut, 'Teste'); %
Gráficos de regressão individual para cada execução de rede

trTargs{j} = [trTarg]; % Célula de vetor(es) de alvos utilizados no treinamento
trOuts{j} = [trOut]; % Célula de vetor(es) de saídas retornadas do treinamento
vTargs{j} = [vTarg]; % Célula de vetor(es) de alvos utilizados na validação
vOuts{j} = [vOut]; % Célula de vetor(es) de saídas retornadas da validação
tsTargs{j} = [tsTarg]; % Célula de vetor(es) de alvos utilizados no teste
tsOuts{j} = [tsOut]; % Célula de vetor(es) de saídas retornadas do teste

% Separando todos os dados das regressões para criar regressão geral do Ensaio
trTamanho = length(trOut); % Tamanho do vetor utilizado no treinamento
vTamanho = length(vOut); % Tamanho do vetor utilizado na validação
tsTamanho = length(tsOut); % Tamanho do vetor utilizado no teste

trTargsLinha = cell2mat(trTargs); trTargsMatriz = reshape(trTargsLinha, [], j); % Organiza matriz de alvos do
treinamento
trOutsLinha = cell2mat(trOuts); trOutsMatriz = reshape(trOutsLinha, [], j); % Organiza matriz de saídas do
treinamento
vTargsLinha = cell2mat(vTargs); vTargsMatriz = reshape(vTargsLinha, [], j); % Organiza matriz de alvos da
validação
vOutsLinha = cell2mat(vOuts); vOutsMatriz = reshape(vOutsLinha, [], j); % Organiza matriz de saídas da
validação
```

```
tsTargsLinha = cell2mat(tsTargs); tsTargsMatriz = reshape(tsTargsLinha, [], j); % Organiza matriz de alvos do teste
tsOutsLinha = cell2mat(tsOuts); tsOutsMatriz = reshape(tsOutsLinha, [], j); % Organiza matriz de saídas do teste

trTargsMatrizMedia = mean(trTargsMatriz,2); % Calcula um vetor médio da matriz de alvos do treinamento
trOutsMatrizMedia = mean(trOutsMatriz,2); % Calcula um vetor médio da matriz de saídas do treinamento
vTargsMatrizMedia = mean(vTargsMatriz,2); % Calcula um vetor médio da matriz de alvos da validação
vOutsMatrizMedia = mean(vOutsMatriz,2); % Calcula um vetor médio da matriz de saídas da validação
tsTargsMatrizMedia = mean(tsTargsMatriz,2); % Calcula um vetor médio da matriz de alvos do teste
tsOutsMatrizMedia = mean(tsOutsMatriz,2); % Calcula um vetor médio da matriz de saídas do teste

% Coeficiente de regressão 'R'
trCyt = corrcoef(trTarg', trOut'); trR{j} = trCyt(2,1); % Coeficiente 'R' do treinamento
vCyt = corrcoef(vTarg', vOut'); vR{j} = vCyt(2,1); % Coeficiente 'R' da validação
tsCyt = corrcoef(tsTarg', tsOut'); tsR{j} = tsCyt(2,1); % Coeficiente 'R' do teste

tsOutArr = round(tsOut); % Arredondamento do vetor de saída utilizado no teste para facilitar comparação
Diferenca = tsTarg' - tsOutArr'; % Calcula a diferença entre o vetor de alvo para o vetor de saída do teste

% Erro Quadrático Médio
MSEtr{j} = mean((trTarg - trOut).^2); % EQM do treinamento
MSEv{j} = mean((vTarg - vOut).^2); % EQM da validação
MSEts{j} = mean((tsTarg - tsOut).^2); % EQM do teste

Comparacao{j} = [tsTarg' tsOutArr' Diferenca]; % Matriz para comparar saída, alvo e a diferença entre ambos

QtdAcertos{j} = sum(Diferenca == 0); % Conta quantos acertos a rede conseguiu obter
PercentualAcertos = (100*(QtdAcertos{j}))/(length(Comparacao{j})); % Calcula o percentual de acertos
PercentuaisAcertos{j} = PercentualAcertos; % Célula de vetor(es) de percentual(is) de acertos
Ns{j} = N; % Célula de vetor(es) do(s) parâmetro(s) principal(is) da rede para exibição
tsTamanhos{j} = tsTamanho; % Célula de vetor(es) da(s) quantidade(s) de elementos usados no teste da rede, para exibição

%% Organização dos resultados para exibição final
Resumo1 = [tsTarg' tsOutArr' Diferenca]; % Matriz para resumir a comparação entre alvo, saída e diferença entre elas
Tabela1{j} = array2table(Resumo1,'VariableNames',{'Alvo','Resultado','Diferenca'}); % Tabela para resumir a comparação entre alvo, saída e diferença entre elas para todos os ciclos da rede
Resumo2 = [Ns' QtdAcertos' tsTamanhos' PercentuaisAcertos' MSEtr' MSEv' MSEts' trR' vR' tsR']; % Matriz
```

```
com informações gerais das execuções das redes para comparação
Tabela2 =
array2table(Resumo2,'VariableNames',{'Parametro','Quantidade_Acertos','Total','Percentual_Acertos','MSE_Treinamento','MSE_Validacao','MSE_Teste','Regressao_Treinamento','Regressao_Validacao','Regressao_Teste'});
% Tabela com informações gerais das execuções das redes para comparação
end;

%% Armazenamento em arquivos
SalvaArquivo = sprintf('redepatternnet_AreaDeTrabalho_Ensaio_%d.mat',NumeroEnsaio);
save(SalvaArquivo); % Salva arquivo com todas as variáveis da área de trabalho
writetable(Tabela2,sprintf('redepatternnet_TabelaResumo_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de resultados finais
figure; % Cria janela com figura para apresentar a regressão média de todas os ciclos das redes
RegressaoMedia = plotregression(trTargsMatrizMedia', trOutsMatrizMedia', 'Média Trein.',
vTargsMatrizMedia', vOutsMatrizMedia', 'Média Valid.', tsTargsMatrizMedia', tsOutsMatrizMedia', 'Média Teste'); % Regressão média de todas os ciclos da rede
NomeArquivoFigura = sprintf('redepatternnet_RegressaoMedia_Ensaio_%d',NumeroEnsaio); % Cria nome para salvar em arquivo a figura da regressão média dos ciclos da rede
saveas(gca,NomeArquivoFigura,'jpeg'); % Salva figura de regressão média dos ciclos da rede em formato '.jpeg'

%% Criação de gráfico comparativo "Alvos X Resultados" para a melhor parametrização da RNA
figure; % Cria janela com figura para apresentar a relação de alvos X resultados
plot(tsTarg,'r'); % Traça as amostras no gráfico, na cor vermelha
xlabel('Índices das Amostras'); % Rotula o eixo x: "Índices das Amostras"
axis([0 tsTamanho 0 8]); % Define os tamanhos máximos dos eixos "x" e "y" do gráfico
Escala = tsTamanho/7; % Calcula a escala de exibição do eixo "x" de acordo com a quantidade de amostras
xticks(0:Escala:tsTamanho); % Define o tamanho do eixo "x" com sua escala
ylabel('Amostras'); % Rotula o eixo y: "Amostras"
hold on; % "Trava" o gráfico para inserir o tracejado dos resultados
plot(tsOutsMatrizMedia,'b'); % Traça os resultados no gráfico, na cor azul
legend('Alvo','Resultado','location','nw'); % Define os itens da legenda e posiciona no canto superior esquerdo
print -dpng GraficoMelhorParametrizacao_redepatternnet.png % Armazena o gráfico pronto em arquivo
```

A.1.5 Rede Neural Probabilística

```
% Definição dos dados
DadosPares = Dados(2:2:end,:); % Separa as linhas pares de todos os dados
DadosImpares = Dados(1:2:end,:); % Separa as linhas ímpares de todos os dados
xPares = DadosPares(:,1:3); % Entrada => energia, banda média e largura de banda equivalente (para criação da RNA)
```

```
xImpares = DadosImpares(:,1:3); % Entrada => energia, banda média e largura de banda equivalente (para testar manualmente a RNA)
dPares = DadosPares(:,4); % Saída desejada das linhas pares para utilizar no "treinamento"
DPares = ind2vec(dPares'); % Saída desejada das linhas pares para utilizar no "treinamento" (conversão dos índices em vetores binários)
dImpares = DadosImpares(:,4); % Saída desejada das linhas ímpares para utilizar no teste
DImpares = ind2vec(dImpares'); % Saída desejada das linhas pares para utilizar no "treinamento" (conversão dos índices em vetores binários)

%% Seleção de parâmetro para o ensaio
% Spread = 0.01; NumeroEnsaio = 1; % Taxa de propagação / Número do ensaio
% Spread = 0.1; NumeroEnsaio = 2; % Taxa de propagação / Número do ensaio
% Spread = 0.2; NumeroEnsaio = 3; % Taxa de propagação / Número do ensaio
% Spread = 0.4; NumeroEnsaio = 4; % Taxa de propagação / Número do ensaio
% Spread = 1; NumeroEnsaio = 5; % Taxa de propagação / Número do ensaio
% Spread = 1.2; NumeroEnsaio = 6; % Taxa de propagação / Número do ensaio
% Spread = 2; NumeroEnsaio = 7; % Taxa de propagação / Número do ensaio
% Spread = 10; NumeroEnsaio = 8; % Taxa de propagação / Número do ensaio
% Spread = 20; NumeroEnsaio = 9; % Taxa de propagação / Número do ensaio * Melhor parametrização
% Spread = 30; NumeroEnsaio = 10; % Taxa de propagação / Número do ensaio

%% Carregamento de RNA
net = newpnn(xImpares',DImpares,Spread); % Rede Neural Probabilística
y = net(xPares'); % Executa a simulação da rede
Y = vec2ind(y); % Converte vetores binários em índices
tsOutDouble = vec2ind(y); % Converte vetores binários em índices

%% Regressão dos dados de teste
tsOut = Y; % Vetor de saídas retornadas do teste
tsTarg = dPares'; % Vetor de alvos utilizados no teste

tsTamanho = length(tsOutDouble); % Tamanho do vetor utilizado no teste

% Coeficiente de regressão 'R'
tsCyt = corrcoef(tsTarg', tsOut'); tsR = tsCyt(2,1); % Coeficiente 'R' do teste
tsCytMedio = corrcoef(tsTarg', tsOut'); tsRMedio = tsCytMedio(2,1);  % Média de coeficiente(s) 'R' do teste

tsOutArr = round(tsOutDouble); % Arredondamento do vetor de saída utilizado no teste para facilitar comparação
Diferenca = dPares' - (tsOutArr); % Calcula a diferença entre o vetor de alvo para o vetor de saída do teste
```

```
% Erro Quadrático Médio do teste
MSEts = mean((tsTarg - tsOut).^2);

Comparacao = [dPares'; tsOutDouble; Diferenca]'; % Matriz para comparar saída, alvo e a diferença entre ambos

QtdAcertos = sum(Diferenca == 0); % Conta quantos acertos a rede conseguiu obter
PercentualAcertos = (100*(QtdAcertos))/(length(Comparacao)); % Calcula o percentual de acertos
PercentuaisAcertos = PercentualAcertos; % Célula de vetor(es) de percentual(is) de acertos
Spreads = Spread; % Célula de vetor(es) do(s) parâmetro(s) principal(is) da rede para exibição
tsTamanhos = tsTamanho; % Célula de vetor(es) da(s) quantidade(s) de elementos usados no teste da rede, para exibição

%% Organização dos resultados para exibição final
Resumo1 = [dPares'; tsOutDouble; Diferenca]'; % Matriz para resumir a comparação entre alvo, saída e diferença entre elas
Tabela1 = array2table(Resumo1,'VariableNames',{'Alvo','Resultado','Diferenca'}); % Tabela para resumir a comparação entre alvo, saída e diferença entre elas para todos os ciclos da rede
Resumo2 = [Spreads' QtdAcertos' tsTamanhos' PercentuaisAcertos' MSEts' tsR']; % Matriz com informações gerais das execuções das redes para comparação
Tabela2 =
array2table(Resumo2,'VariableNames',{'Parametro','Quantidade_Acertos','Total','Percentual_Acertos','MSE_Teste','Regressao_Teste'});  % Tabela com informações gerais das execuções das redes para comparação

%% Armazenamento em arquivos
SalvaArquivo = sprintf('redenewpnn_AreaDeTrabalho_Ensaio_%d.mat',NumeroEnsaio);
save(SalvaArquivo); % Salva arquivo com todas as variáveis da área de trabalho
writetable(Tabela1,sprintf('redenewpnn_TabelaResultados_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de todos os resultados obtidos
writetable(Tabela2,sprintf('redenewpnn_TabelaResumo_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de resultados finais
figure; % Cria janela com figura para apresentar a regressão
Regressao = plotregression(tsTarg, tsOut,'Teste'); % Gráficos de regressão individual para cada execução de rede
NomeArquivoFigura = sprintf('redenewpnn_RegressaoMedia_Ensaio_%d',NumeroEnsaio); % Cria nome para salvar em arquivo a figura da regressão média dos ciclos da rede
saveas(gca,NomeArquivoFigura,'jpeg'); % Salva figura de regressão média dos ciclos da rede em formato '.jpeg'
```

A.1.6 Rede de Regressão Generalizada

```
% Definição dos dados
DadosPares = Dados(2:2:end,:); % Separa as linhas pares de todos os dados
DadosImpares = Dados(1:2:end,:); % Separa as linhas ímpares de todos os dados
xPares = DadosPares(:,1:3); % Entrada => energia, banda média e largura de banda equivalente (para criação da RNA)
xImpares = DadosImpares(:,1:3); % Entrada => energia, banda média e largura de banda equivalente (para testar manualmente a RNA)
dPares = DadosPares(:,4); % Saída desejada das linhas pares para utilizar no "treinamento"
DPares = ind2vec(dPares'); % Saída desejada das linhas pares para utilizar no "treinamento" (conversão dos índices em vetores binários)
dImpares = DadosImpares(:,4); % Saída desejada das linhas ímpares para utilizar no teste
DImpares = ind2vec(dImpares'); % Saída desejada das linhas pares para utilizar no "treinamento" (conversão dos índices em vetores binários)

%% Seleção de parâmetro para o ensaio
% Spread = 0.01; NumeroEnsaio = 1; % Taxa de propagação / Número do ensaio
% Spread = 0.1; NumeroEnsaio = 2; % Taxa de propagação / Número do ensaio
% Spread = 0.2; NumeroEnsaio = 3; % Taxa de propagação / Número do ensaio
% Spread = 0.4; NumeroEnsaio = 4; % Taxa de propagação / Número do ensaio
% Spread = 1; NumeroEnsaio = 5; % Taxa de propagação / Número do ensaio
% Spread = 1.2; NumeroEnsaio = 6; % Taxa de propagação / Número do ensaio
% Spread = 2; NumeroEnsaio = 7; % Taxa de propagação / Número do ensaio
% Spread = 10; NumeroEnsaio = 8; % Taxa de propagação / Número do ensaio
% Spread = 20; NumeroEnsaio = 9; % Taxa de propagação / Número do ensaio
% Spread = 100; NumeroEnsaio = 10; % Taxa de propagação / Número do ensaio

%% Carregamento de RNA
net = newgrnn(xImpares',DImpares,Spread); % Rede Generalized Regression
y = net(xPares'); % Executa a simulação da rede
Y = vec2ind(y); % Converte vetores binários em índices
tsOutDouble = vec2ind(y); % Converte vetores binários em índices

%% Regressão dos dados de teste
tsOut = Y; % Vetor de saídas retornadas do teste
tsTarg = dPares'; % Vetor de alvos utilizados no teste

tsTamanho = length(tsOutDouble); % Tamanho do vetor utilizado no teste
```

```
% Coeficiente de regressão 'R'
tsCyt = corrcoef(tsTarg', tsOut'); tsR = tsCyt(2,1); % Coeficiente 'R' do teste
tsCytMedio = corrcoef(tsTarg', tsOut'); tsRMedio = tsCytMedio(2,1);  % Média de coeficiente(s) 'R' do teste

tsOutArr = round(tsOutDouble); % Arredondamento do vetor de saída utilizado no teste para facilitar comparação
Diferenca = dPares' - (tsOutArr); % Calcula a diferença entre o vetor de alvo para o vetor de saída do teste

% Erro Quadrático Médio do teste
MSEts = mean((tsTarg - tsOut).^2);

Comparacao = [dPares'; tsOutDouble; Diferenca]'; % Matriz para comparar saída, alvo e a diferença entre ambos

QtdAcertos = sum(Diferenca == 0); % Conta quantos acertos a rede conseguiu obter
PercentualAcertos = (100*(QtdAcertos))/(length(Comparacao)); % Calcula o percentual de acertos
PercentuaisAcertos = PercentualAcertos; % Célula de vetor(es) de percentual(is) de acertos
Spreads = Spread; % Célula de vetor(es) do(s) parâmetro(s) principal(is) da rede para exibição
tsTamanhos = tsTamanho; % Célula de vetor(es) da(s) quantidade(s) de elementos usados no teste da rede, para exibição

%% Organização dos resultados para exibição final
Resumo1 = [dPares'; tsOutDouble; Diferenca]'; % Matriz para resumir a comparação entre alvo, saída e diferença entre elas
Tabela1 = array2table(Resumo1,'VariableNames',{'Alvo','Resultado','Diferenca'}); % Tabela para resumir a comparação entre alvo, saída e diferença entre elas para todos os ciclos da rede
Resumo2 = [Spreads' QtdAcertos' tsTamanhos' PercentuaisAcertos' MSEts' tsR']; % Matriz com informações gerais das execuções das redes para comparação
Tabela2 =
array2table(Resumo2,'VariableNames',{'Parametro','Quantidade_Acertos','Total','Percentual_Acertos','MSE_Teste','Regressao_Teste'});  % Tabela com informações gerais das execuções das redes para comparação

%% Armazenamento em arquivos
SalvaArquivo = sprintf('redenewgrnn_AreaDeTrabalho_Ensaio_%d.mat',NumeroEnsaio);
save(SalvaArquivo); % Salva arquivo com todas as variáveis da área de trabalho
writetable(Tabela1,sprintf('redenewgrnn_TabelaResultados_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de todos os resultados obtidos
writetable(Tabela2,sprintf('redenewgrnn_TabelaResumo_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de resultados finais
figure; % Cria janela com figura para apresentar a regressão
```

```
Regressao = plotregression(tsTarg, tsOut,'Teste'); % Gráficos de regressão individual para cada execução de rede
NomeArquivoFigura = sprintf('redenewgrnn_RegressaoMedia_Ensaio_%d',NumeroEnsaio); % Cria nome para salvar em arquivo a figura da regressão média dos ciclos da rede
saveas(gca,NomeArquivoFigura,'jpeg'); % Salva figura de regressão média dos ciclos da rede em formato '.jpeg'
```

A.1.7 Rede de Base Radial

```
% Definição dos dados
DadosPares = Dados(2:2:end,:); % Separa as linhas pares de todos os dados
DadosImpares = Dados(1:2:end,:); % Separa as linhas ímpares de todos os dados
xPares = DadosPares(:,1:3); % Entrada => energia, banda média e largura de banda equivalente (para criação da RNA)
xImpares = DadosImpares(:,1:3); % Entrada => energia, banda média e largura de banda equivalente (para testar manualmente a RNA)
dPares = DadosPares(:,4); % Saída desejada das linhas pares para utilizar no "treinamento"
dImpares = DadosImpares(:,4); % Saída desejada das linhas ímpares para utilizar no teste
DImpares = ind2vec(dImpares'); % Saída desejada das linhas pares para utilizar no "treinamento" (conversão dos índices em vetores binários)

%% Seleção de parâmetro para o ensaio
% Spread = 0.01; MN = 100; DF = 1; NumeroEnsaio = 1; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio
% Spread = 0.1; MN = 100; DF = 1; NumeroEnsaio = 2; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio
% Spread = 1; MN = 100; DF = 1; NumeroEnsaio = 3; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio
% Spread = 10; MN = 100; DF = 1; NumeroEnsaio = 4; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio
% Spread = 1500; MN = 100; DF = 1; NumeroEnsaio = 5; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio
% Spread = 1650; MN = 100; DF = 1; NumeroEnsaio = 6; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio
% Spread = 1800; MN = 80; DF = 1; NumeroEnsaio = 7; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio * Melhor parametrização
% Spread = 1850; MN = 90; DF = 1; NumeroEnsaio = 8; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio
% Spread = 1850; MN = 100; DF = 1; NumeroEnsaio = 9; % Taxa de propagação / Máximo nº de neurônios / Nº de neurônios entre telas / Número do ensaio
% Spread = 2000; MN = 30; DF = 1; NumeroEnsaio = 10; % Taxa de propagação / Máximo nº de neurônios /
```

```
Nº de neurônios entre telas / Número do ensaio

%% Carregamento de RNA
goal = 1e-7; % Erro desejado
net = newrb(xImpares',dImpares', goal, Spread, MN, DF); % Rede de Base Radial
y = net(xPares'); % Executa a simulação da rede
%% Regressão dos dados de teste
tsOut = y; % Vetor de saídas retornadas do teste
tsTarg = dPares'; % Vetor de alvos utilizados no teste

tsTamanho = length(y); % Tamanho do vetor utilizado no teste

% Coeficiente de regressão 'R'
tsCyt = corrcoef(tsTarg', tsOut'); tsR = tsCyt(2,1); % Coeficiente 'R' do teste
tsCytMedio = corrcoef(tsTarg', tsOut'); tsRMedio = tsCytMedio(2,1);  % Média de coeficiente(s) 'R' do teste

tsOutArr = round(y); % Arredondamento do vetor de saída utilizado no teste para facilitar comparação
Diferenca = dPares' - (tsOutArr); % Calcula a diferença entre o vetor de alvo para o vetor de saída do teste

% Erro Quadrático Médio do teste
MSEts = mean((tsTarg - tsOut).^2);

Comparacao = [dPares'; tsOutArr; Diferenca]'; % Matriz para comparar saída, alvo e a diferença entre ambos

QtdAcertos = sum(Diferenca == 0); % Conta quantos acertos a rede conseguiu obter
PercentualAcertos = (100*(QtdAcertos))/(length(Comparacao)); % Calcula o percentual de acertos
PercentuaisAcertos = PercentualAcertos; % Célula de vetor(es) de percentual(is) de acertos
Spreads = Spread; % Célula de vetor(es) do(s) parâmetro(s) principal(is) (Spread) da rede para exibição
MNs = MN; % Célula de vetor(es) do(s) parâmetro(s) principal(is) (MN) da rede para exibição
DFs = DF; % Célula de vetor(es) do(s) parâmetro(s) principal(is) (DF) da rede para exibição
tsTamanhos = tsTamanho; % Célula de vetor(es) da(s) quantidade(s) de elementos usados no teste da rede, para
exibição

%% Organização dos resultados para exibição final
Resumo1 = [dPares'; tsOutArr; Diferenca]'; % Matriz para resumir a comparação entre alvo, saída e diferença
entre elas
Tabela1 = array2table(Resumo1,'VariableNames',{'Alvo','Resultado','Diferenca'}); % Tabela para resumir a
comparação entre alvo, saída e diferença entre elas para todos os ciclos da rede
Resumo2 = [Spreads' MNs' DFs' QtdAcertos' tsTamanhos' PercentuaisAcertos' MSEts' tsR']; % Matriz com
informações gerais das execuções das redes para comparação
```

```
Tabela2 =
array2table(Resumo2,'VariableNames',{'Parametro_Spread','Parametro_MN','Parametro_DF','Quantidade_Acertos','Total','Percentual_Acertos','MSE_Teste','Regressao_Teste'});  % Tabela com informações gerais das execuções das redes para comparação

%% Armazenamento em arquivos
SalvaArquivo = sprintf('redenewrb_AreaDeTrabalho_Ensaio_%d.mat',NumeroEnsaio); save(SalvaArquivo); % Salva arquivo com todas as variáveis da área de trabalho
writetable(Tabela1,sprintf('redenewrb_TabelaResultados_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de todos os resultados obtidos
writetable(Tabela2,sprintf('redenewrb_TabelaResumo_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de resultados finais
figure; % Cria janela com figura para apresentar a regressão
Regressao = plotregression(tsTarg, tsOut,'Teste'); % Gráficos de regressão individual para cada execução de rede
NomeArquivoFigura = sprintf('redenewrb_RegressaoMedia_Ensaio_%d',NumeroEnsaio); % Cria nome para salvar em arquivo a figura da regressão média dos ciclos da rede
saveas(gca,NomeArquivoFigura,'jpeg'); % Salva figura de regressão média dos ciclos da rede em formato '.jpeg'
```

A.1.8 Rede de Base Radial Exata

```
% Definição dos dados
DadosPares = Dados(2:2:end,:); % Separa as linhas pares de todos os dados
DadosImpares = Dados(1:2:end,:); % Separa as linhas ímpares de todos os dados
xPares = DadosPares(:,1:3); % Entrada => energia, banda média e largura de banda equivalente (para criação da RNA)
xImpares = DadosImpares(:,1:3); % Entrada => energia, banda média e largura de banda equivalente (para testar manualmente a RNA)
dPares = DadosPares(:,4); % Saída desejada das linhas pares para utilizar no "treinamento"
dImpares = DadosImpares(:,4); % Saída desejada das linhas ímpares para utilizar no teste
DImpares = ind2vec(dImpares'); % Saída desejada das linhas pares para utilizar no "treinamento" (conversão dos índices em vetores binários)

%% Seleção de parâmetro para o ensaio
% Spread = 0.01; NumeroEnsaio = 1; % Taxa de propagação / Número do ensaio
% Spread = 0.1; NumeroEnsaio = 2; % Taxa de propagação / Número do ensaio
% Spread = 1; NumeroEnsaio = 3; % Taxa de propagação / Número do ensaio
% Spread = 10; NumeroEnsaio = 4; % Taxa de propagação / Número do ensaio
% Spread = 13; NumeroEnsaio = 5; % Taxa de propagação / Número do ensaio
% Spread = 18; NumeroEnsaio = 6; % Taxa de propagação / Número do ensaio
```

```
% Spread = 20; NumeroEnsaio = 7; % Taxa de propagação / Número do ensaio
% Spread = 22; NumeroEnsaio = 8; % Taxa de propagação / Número do ensaio
% Spread = 25; NumeroEnsaio = 9; % Taxa de propagação / Número do ensaio
% Spread = 100; NumeroEnsaio = 10; % Taxa de propagação / Número do ensaio

%% Carregamento de RNA
net = newrbe(xImpares',dImpares',Spread); % Rede de Base Radial Exata
y = net(xPares'); % Executa a simulação da rede

%% Regressão dos dados de teste
tsOut = y; % Vetor de saídas retornadas do teste
tsTarg = dPares'; % Vetor de alvos utilizados no teste

tsTamanho = length(y); % Tamanho do vetor utilizado no teste

% Coeficiente de regressão 'R'
tsCyt = corrcoef(tsTarg', tsOut'); tsR = tsCyt(2,1); % Coeficiente 'R' do teste
tsCytMedio = corrcoef(tsTarg', tsOut'); tsRMedio = tsCytMedio(2,1);  % Média de coeficiente(s) 'R' do teste

tsOutArr = round(y); % Arredondamento do vetor de saída utilizado no teste para facilitar comparação
Diferenca = dPares' - (tsOutArr); % Calcula a diferença entre o vetor de alvo para o vetor de saída do teste

% Erro Quadrático Médio do teste
MSEts = mean((tsTarg - tsOut).^2);

Comparacao = [dPares'; tsOutArr; Diferenca]'; % Matriz para comparar saída, alvo e a diferença entre ambos

QtdAcertos = sum(Diferenca == 0); % Conta quantos acertos a rede conseguiu obter
PercentualAcertos = (100*(QtdAcertos))/(length(Comparacao)); % Calcula o percentual de acertos
PercentuaisAcertos = PercentualAcertos; % Célula de vetor(es) de percentual(is) de acertos
Spreads = Spread; % Célula de vetor(es) do(s) parâmetro(s) principal(is) da rede para exibição
tsTamanhos = tsTamanho; % Célula de vetor(es) da(s) quantidade(s) de elementos usados no teste da rede, para
exibição

%% Organização dos resultados para exibição final
Resumo1 = [dPares'; tsOutArr; Diferenca]'; % Matriz para resumir a comparação entre alvo, saída e diferença
entre elas
Tabela1 = array2table(Resumo1,'VariableNames',{'Alvo','Resultado','Diferenca'}); % Tabela para resumir a
comparação entre alvo, saída e diferença entre elas para todos os ciclos da rede
Resumo2 = [Spreads' QtdAcertos' tsTamanhos' PercentuaisAcertos' MSEts' tsR']; % Matriz com informações
```

```
gerais das execuções das redes para comparação
Tabela2 =
array2table(Resumo2,'VariableNames',{'Parametro','Quantidade_Acertos','Total','Percentual_Acertos','MSE_Teste','Regressao_Teste'});  % Tabela com informações gerais das execuções das redes para comparação

%% Armazenamento em arquivos
SalvaArquivo = sprintf('redenewrbe_AreaDeTrabalho_Ensaio_%d.mat',NumeroEnsaio);
save(SalvaArquivo); % Salva arquivo com todas as variáveis da área de trabalho
writetable(Tabela1,sprintf('redenewrbe_TabelaResultados_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de todos os resultados obtidos
writetable(Tabela2,sprintf('redenewrbe_TabelaResumo_Ensaio_%d.xlsx',NumeroEnsaio)); % Salva tabela de resultados finais
figure; % Cria janela com figura para apresentar a regressão
Regressao = plotregression(tsTarg, tsOut,'Teste'); % Gráficos de regressão individual para cada execução de rede
NomeArquivoFigura = sprintf('redenewrbe_RegressaoMedia_Ensaio_%d',NumeroEnsaio); % Cria nome para salvar em arquivo a figura da regressão média dos ciclos da rede
saveas(gca,NomeArquivoFigura,'jpeg'); % Salva figura de regressão média dos ciclos da rede em formato '.jpeg'
```