

UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

**Comparação entre as Redes Neurais Artificiais MLP,
RBF e LVQ na Classificação de Dados**

Fernando Nunes Bonifácio

CASCABEL

2010

FERNANDO NUNES BONIFÁCIO

**COMPARAÇÃO ENTRE AS REDES NEURAIS ARTIFICIAIS MLP, RBF
E LVQ NA CLASSIFICAÇÃO DE DADOS**

Monografia apresentada como requisito parcial
para obtenção do grau de Bacharel em Ciência
da Computação, do Centro de Ciências Exatas
e Tecnológicas da Universidade Estadual do
Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Clodis Boscaroli

CASCADEL

2010

FERNANDO NUNES BONIFÁCIO

**COMPARAÇÃO ENTRE AS REDES NEURAIS ARTIFICIAIS MLP, RBF
E LVQ NA CLASSIFICAÇÃO DE DADOS**

Monografia apresentada como requisito parcial para obtenção do Título de *Bacharel em Ciência da Computação*, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Clodis Boscaroli (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Marcio Seiji Oyamada
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Jorge Bidarra
Colegiado de Ciência da Computação,
UNIOESTE

Cascavel, Novembro de 2010.

Lista de Figuras

2.1	Fluxo geral do processo KDD.....	4
2.2	Tarefa de Classificação de Dados	8
2.3	Estimativa de acuidade pelo método <i>Holdout</i>	14
3.1	Célula nervosa ou neurônio biológico	17
3.2	Modelo matemático de um neurônio artificial.....	18
3.3	Modelo neural de McCulloch e Pitts	19
3.4	Representação de uma função limiar	20
3.5	Representação de uma função linear.....	21
3.6	Representação de uma função linear por partes.....	21
3.7	Representação de uma função sigmóide	22
3.8	Arquiteturas de RNAs.....	23
3.9	Arquitetura Rede MLP.....	26
3.10	Regiões definidas pelo processamento da segunda camada intermediária de uma Rede MLP de duas camadas ocultas	27
3.11	Regiões definidas pelo processamento da camada de saída de uma Rede MLP de duas camadas ocultas	27
3.12	Arquitetura Rede RBF	30
3.13	Gráfico de função gaussiana para entrada unidimensional (esquerda) e bidimensional (direita).....	32
3.14	Classificação por Rede MLP (esquerda) e por Rede RBF (direita).....	32
3.15	Arquitetura Rede LVQ.....	35
3.16	Diagrama de Voronoi	36
3.17	Comportamento geométrico dos vetores envolvidos no algoritmo LVQ	38

4.1	Diagrama de Classes da <i>Interface ClassifierModuleInterface</i>	41
4.2	Imagem da ferramenta YADMT com os módulos/técnicas de Classificação disponíveis	43
4.3	Parâmetros de configuração da Rede MLP	44
4.4	Parâmetros de configuração da Rede RBF	45
4.5	Parâmetros de configuração da Rede LVQ	46

Lista de Tabelas

2.1	Exemplo de matriz de confusão	12
2.2	Matriz de confusão para registros positivas e negativas	12
5.1	Bases de dados escolhidas para comparação das Redes Neurais Artificiais implementadas	47
5.2	Mapeamento atributo <i>Course Instructor</i> da base de dados <i>Teaching Assitant Evaluation</i>	49
5.3	Acuidade das Redes MLP nas 10 bases de dados escolhidas	51
5.4	Tempos de treinamentos das Redes MLP	52
5.5	Acuidade das Redes RBF nas 10 bases de dados escolhidas	54
5.6	Tempos de treinamentos das Redes RBF	56
5.7	Acuidade das Redes LVQ nas 10 bases de dados escolhidas	58
5.8	Tempos de treinamentos das Redes LVQ	59
A.1	Distribuição de classes da Base de Dados <i>Abalone</i>	66
A.2	Distribuição de classes da Base de Dados <i>Acute Inflammations</i>	67
A.3	Distribuição de classes da Base de Dados <i>Ecoli</i>	68
A.4	Distribuição de classes da Base de Dados <i>Iris</i>	68
A.5	Distribuição de classes da Base de Dados <i>LIBRAS Movement</i>	69
A.6	Distribuição de classes da Base de Dados <i>Spambase</i>	70
A.7	Distribuição de classes da Base de Dados <i>Teaching Assistant Evaluation</i> ..	70
A.8	Atributos da Base de Dados <i>Statlog (Image Segmentation)</i>	71

A.9	Distribuição de classes da Base de Dados <i>Statlog (Image Segmentation)</i> ...	72
A.10	Distribuição de classes da Base de Dados <i>Statlog (Landsat Satellite)</i>	72
A.11	Distribuição de classes da Base de Dados <i>Wine Quality</i>	73

Lista de Abreviaturas e Siglas

DM	<i>Data Mining</i>
KDD	<i>Knowledge Discovery in Databases</i>
K-NN	<i>K-Nearest Neighbors</i>
LVQ	<i>Learning Vector Quantization</i>
MLP	<i>Multilayer Perceptron</i>
RBF	<i>Radial Basis Function</i>
RNAs	Redes Neurais Artificiais
SBGD	Sistema Gerenciador de Banco de Dados
SVM	<i>Support Vector Machines</i>
VLSI	<i>Very Large Scale Integration</i>
YADMT	<i>Yet Another Data Mining Tool</i>

Lista de Símbolos

b	<i>Bias</i> associado a um neurônio artificial
d	Saída desejada de uma Rede Neural Artificial
$dist_{max}$	Distância máxima entre um conjunto de centros de função de base radial
$dist_{min}$	Distância mínima entre um conjunto de centros de função de base radial
$dist$	Distância entre dois centros de função de base radial
e	Erro entre a saída de uma Rede Neural Artificial e a saída desejada
i	Índice inteiro qualquer
j	Índice inteiro qualquer
k	Índice inteiro qualquer
m	Ordem de uma matriz quadrada
ms	Tempo em microssegundos
n	Dimensão do registro de dados
neg	Número de registros classificados como negativo
p	Um número inteiro qualquer
pos	Número de registros classificados como positivo
q	Um número inteiro qualquer
r	Um valor pertencente a um registro de dados
t	Um inteiro representando a iteração de um Classificador em treinamento
v	Potencial de ativação de um neurônio artificial
x	Um valor pertencente a um registro de dados
v_{pos}	Número de registros classificados corretamente como positivo
v_{neg}	Número de registros classificados corretamente como negativo
y	Saída de uma Rede Neural Artificial
Acc	Taxas de acuidade de um Classificador
M	Matriz quadrada de ordem m
W	Conjunto dos valores dos pesos de um neurônio artificial
X	Registro com os valores de entrada para de um neurônio artificial
R	Registro qualquer com valores de entrada de uma Rede Neural Artificial

φ	Função de ativação de um neurônio artificial
θ	Valor limiar do neurônio artificial
μ	Centro de uma função radial
σ	Largura de uma função radial
η	Taxa de aprendizado de um classificador
α	Termo momentum
ε	Erro médio quadrático

Sumário

Lista de Figuras	iv
Lista de Tabelas	vi
Lista de Abreviaturas e Siglas	viii
Lista de Símbolos	ix
Sumário	xi
1 Introdução	1
1.1 Justificativas.....	2
1.2 Objetivos.....	3
1.3 Organização do texto.....	3
2 Conceitos de Fundamentação	4
2.1 Tarefas de Mineração de Dados.....	6
2.2 Classificação.....	7
2.2.1 Algoritmos para Classificação de Dados.....	9
2.2.2 Métodos de Avaliação de Classificação.....	11
3 Redes Neurais Artificiais	15
3.1 Fundamentos Redes Neurais Artificiais.....	16
3.1.1 Tipos de função de ativação.....	20
3.1.2 Arquitetura de Redes Neurais Artificiais.....	22
3.1.3 Aprendizado.....	23

3.1.4	Aprendizado por correção de erros	24
3.2	Redes <i>Multilayer Perceptron</i> – MLP	25
3.2.1	Treinamento	28
3.3	Redes <i>Radial Basis Function</i> – RBF	29
3.3.1	Treinamento	32
3.4	Redes <i>Learning Vector Quantization</i> – LVQ	35
3.4.1	Treinamento	36
4	Implementação das RNAs para a Ferramenta YADMT	40
4.1	Características do Módulo de Classificação	41
4.2	Implementações dos RNAs.....	44
5	Testes Realizados	47
5.1	Testes com Redes MLP	49
5.2	Testes com Redes RBF	53
5.3	Testes com Redes LVQ	56
5.4	Considerações Finais	60
6	Conclusões e Trabalhos Futuros	63
A	Descrição das Bases de Dados	65
A.1	Base de Dados <i>Abalone</i>	65
A.2	Base de Dados <i>Acute Inflammations</i>	66
A.3	Base de Dados <i>Ecoli</i>	67
A.4	Base de Dados <i>Iris</i>	68
A.5	Base de Dados <i>LIBRAS Movement</i>	68
A.6	Base de Dados <i>Spambase</i>	69
A.7	Base de Dados <i>Teaching Assistant Evaluation</i>	70
A.8	Base de Dados <i>Statlog (Image Segmentation)</i>	71

A.9 Base de Dados <i>Statlog (Landsat Satellite)</i>	72
A.10 Base de Dados <i>Wine Quality</i>	72
Referências	74

Resumo

Atualmente o volume de dados armazenados é grande e continua crescendo rapidamente. Devido à limitação humana de interpretar esta grande quantidade de dados, muita informação e conhecimento podem estar sendo desperdiçados, ficando ocultos nas bases de dados. Surgem, então, novas ferramentas e técnicas de extração de conhecimento em um processo conhecido como Descoberta de Conhecimento em Banco de Dados (*Knowledge Discovery in Databases - KDD*). Dentre as técnicas utilizadas neste processo, mais especificamente na Mineração de Dados, destacam-se aquelas baseadas em Redes Neurais Artificiais utilizadas na tarefa de classificação de dados, uma tarefa que realiza aprendizado supervisionado em uma base de dados, gera um modelo que descreve e distingue classes e que pode ser utilizado para classificar novos dados. Neste sentido, as Redes Neurais Artificiais *Multilayer Perceptron*, *Radial Basis Function* e a *Learning Vector Quantization* foram implementadas como classificadores para uma ferramenta computacional livre de KDD, denominada YADMT e, posteriormente, foram testadas em bases de dados públicas.

Palavras-chave: Mineração de Dados, Classificação, Redes Neurais Artificiais, *Multilayer Perceptron*, *Radial Basis Function*, *Learning Vector Quantization*.

Capítulo 1

Introdução

A redução de custo para armazenamento de dados e o rápido processo de informatização de empresas públicas e privadas fez com que estas empresas passassem a armazenar cada vez mais seus dados em sistemas computacionais. O volume de dados armazenados por estes sistemas chegou a um nível onde a análise dos dados, por meio de processos manuais ou de técnicas tradicionais não explora toda a informação implícita disponível. Em 1997, Adriaans & Zantinge [1] já afirmavam que as pessoas não seriam mais capazes de analisar estes dados de forma manual, e que as organizações teriam melhores chances de sobreviver no mercado se utilizassem ferramentas automáticas para extração de conhecimento a partir de seus dados.

A utilização de recursos para este tipo de análise já vem sendo feita desde o final dos anos 80, em um processo chamado Descoberta de Conhecimento em Banco de Dados (*Knowledge Discovery in Databases - KDD*), definido por Fayyad *et al.* [22] como um processo não trivial, de várias etapas, interativo e iterativo, para identificação de padrões compreensíveis, válidos, novos e potencialmente úteis a partir de bases de dados. Dentre as diversas etapas que compõem o processo de KDD, é na etapa chamada Mineração de Dados (*Data Mining - DM*) que ocorre a aplicação de algoritmos específicos para extrair padrões (modelos) de dados que representam conhecimento útil [24].

DM é um campo multidisciplinar que se utiliza de conhecimento de muitas outras áreas como Banco de Dados, Estatística, Inteligência Artificial, Aprendizado de Máquina, Reconhecimento de Padrões e Métodos de Visualização de Dados [58].

Existem diversas tarefas atribuídas à Mineração de Dados como Regras de Associação, Análise de Agrupamento, Predição e Classificação [24]. Dentre essas tarefas, Zhang [84] afirma que a Classificação é uma das atividades humanas mais frequentes, por isto, é uma das áreas de pesquisas mais ativas, sendo utilizada com sucesso em muitas áreas do conhecimento, desde aplicações computacionais diversas até as de finalidades mais específicas, por exemplo, na área médica [84]. Classificação é uma tarefa que consiste em

identificar um classificador (modelo) que permita descrever e distinguir classes de dados ou conceitos que podem ser utilizados para previsões de classes [12]. O amplo uso desta tarefa se deve, principalmente, à variedade de métodos disponíveis para construção de modelos de classificação. Han e Kamber [30] citam como exemplos Árvores de Decisão, classificador Bayesiano, classificador SVM (*Support Vector Machines*), classificador K-NN (*k-Nearest Neighbors*) e Redes Neurais Artificiais (RNAs).

Embora haja esta grande quantidade de técnicas, nem todas são adequadas para certas bases de dados e também não existe uma única técnica que trate adequadamente todas as bases de dados. Por isso, uma das principais questões envolvendo Classificação e outras tarefas de DM é encontrar uma técnica adequada para a solução de certo problema. Na maioria das vezes precisa-se do auxílio de um especialista, mas a determinação do sucesso ou fracasso de alguma técnica acaba dependendo dos testes empíricos realizados.

1.1 Justificativas

A utilização de RNAs na tarefa de Classificação tem evidenciado bons resultados nas mais diversas áreas, entre os quais se incluem detecção de falhas [5] [35], diagnósticos médicos [7] [8], reconhecimento de voz [11], entre outros.

RNAs são técnicas computacionais que apresentam um modelo matemático complexo inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Existem diversos tipos de RNAs sendo que as mais indicadas para a tarefa de Classificação tem sido a Rede *Multilayer Perceptron* (MLP) [47] [49], a Rede *Radial Basis Function* (RBF) [15] [85] e a Rede *Learning Vector Quantization* (LVQ) [44] [72].

Atualmente existem diversas ferramentas que implementam técnicas de Classificação de dados, incluindo as baseadas em RNAs, algumas delas, como SAS Enterprise Miner [66], IBM Intelligent Miner [37], Oracle Data Mining [56], que são fortemente dependentes de Sistema Gerenciadores de Banco de Dados (SGBDs), outras não são dependentes de um SGBD específico, como Weka [80], Tanagra [75] e Orange [57], mas exigem a transformação dos dados para um formato de entrada específico. Alternativamente a estas aplicações, foi proposto o desenvolvimento de uma ferramenta livre e modular chamada YADMT – *Yet Another Data Mining Tool*, que tem como objetivo permitir a implementação em longo prazo de todas as fases de KDD, tornando-se uma ferramenta completa de forma que possa ser facilmente expandida, independente de SGBD e de formato de dados.

Desta forma, o presente trabalho consiste na implementação das Redes MLP, RBF e LVQ como módulos da ferramenta proposta e posterior análise exploratória e comparação do desempenho das RNAs em um conjunto de bases de dados públicas.

1.2 Objetivos

O objetivo geral do trabalho é implementar e analisar a aplicação de RNAs na tarefa de classificação de dados.

Este objetivo ramifica-se em:

1. Implementar os módulos dos algoritmos escolhidos para a ferramenta proposta.
2. Analisar os resultados obtidos com RNAs na tarefa de classificação em bases de dados públicas.

1.3 Organização do texto

Este trabalho está organizado da seguinte maneira:

O Capítulo 2 explora de forma geral todo o processo de KDD, apresentando as diversas tarefas pertencentes à etapa de Mineração de Dados e exemplos de como tem sido utilizada. É dado um foco especial na tarefa de Classificação, onde são apresentados alguns algoritmos para tal tarefa, bem como são apresentados alguns métodos de avaliação de classificadores.

O Capítulo 3 aborda Redes Neurais Artificiais. Este capítulo apresenta os principais fundamentos desta técnica e permite entender o seu funcionamento. Posteriormente, apresenta as características individuais das Redes MLP, RBF e LVQ.

O Capítulo 4 trata sobre implementação das RNAs para a ferramenta YADMT, definindo o que deve ser feito para implementar um módulo de Classificação para tal ferramenta.

No Capítulo 5 são apresentadas as bases de dados utilizadas para os testes das RNAs, bem como é feita uma análise individual sobre o desempenho de cada uma no conjunto de bases de dados. Por fim, o Capítulo 6 traz as conclusões e sugestões de trabalhos futuros desse estudo.

Capítulo 2

Conceitos de Fundamentação

Historicamente a noção de encontrar padrões úteis nos dados tem sido conhecida por uma variedade de nomes, incluindo Mineração de Dados, Extração de Conhecimento, Descoberta de Informação, Coleta de Informação e Processamento de Padrões de Dados. Foi somente em 1989 que o termo KDD (*Knowledge Discovery in Database* - Descoberta de Conhecimento em Banco de Dados) foi utilizado para enfatizar que o "conhecimento" é o produto final de um processo de descoberta orientado a dados, [25].

KDD é um processo interativo e iterativo de várias etapas, ou seja, é repetitivo e muitas vezes precisa do auxílio do usuário [30]. As etapas deste processo são abaixo descritas, e uma representação de seus relacionamentos pode ser verificada na Figura 2.1.

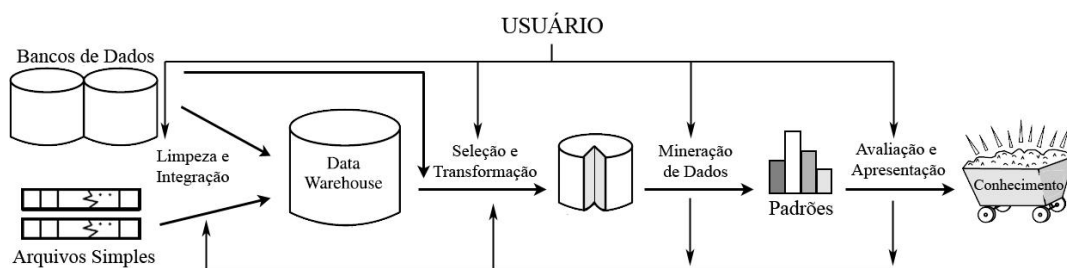


Figura 2.1: Fluxo geral do processo KDD. Adaptado de [30].

1. Limpeza dos dados: etapa onde ocorre a remoção de ruídos e dados inconsistentes.
2. Integração dos dados: etapa onde ocorre a combinação de várias fontes de dados.
3. Seleção dos dados: onde os dados relevantes para a tarefa de análise são recuperados do banco de dados.
4. Transformação dos dados: etapa onde os dados são transformados para formas apropriadas de acordo com o algoritmo a ser utilizado na etapa de mineração de dados, por exemplo, normalização dos dados na faixa de valores entre 0 e 1.

5. Mineração dos dados: etapa essencial onde métodos inteligentes são aplicados para extrair padrões de dados. É nesta etapa que estão situadas a tarefa de Classificação e as Redes Neurais Artificiais, foco deste trabalho.

6. Avaliação dos padrões: etapa para identificar padrões verdadeiramente interessantes representando algum conhecimento baseado em alguma medida de interesse, por exemplo, um padrão encontrado é interessante se é válido com algum grau de certeza quando aplicado a novos dados.

7. Apresentação do conhecimento: etapa onde o conhecimento descoberto na mineração de dados é apresentado ao usuário através de técnicas de visualização e representação de conhecimento, como proposições lógicas, regras do tipo “se-então”, redes semânticas, nas próprias conexões dos neurônios em uma Rede Neural Artificial, entre outras.

A aplicação de KDD não precisa seguir necessariamente estas etapas de forma rígida, qualquer uma delas pode ser pulada, retomada ou repetida, dependendo do problema a ser analisado e do resultado esperado.

Alguns autores, como Kort e Silberschatz, [43], preferem simplificar este processo definindo-o em três etapas bem distintas, Pré-Processamento, Mineração de dados e Pós-Processamento.

Nessa abordagem, o Pré-Processamento passa a agrupar os passos de limpeza, integração, seleção e transformação dos dados, enquanto cabe ao Pós-Processamento a avaliação dos padrões e a apresentação do conhecimento. Independente da taxonomia adotada, é no passo de Mineração de Dados que efetivamente ocorre a descoberta do conhecimento.

Embora DM seja a tarefa mais complexa, pois envolve diversas áreas - aprendizagem de máquina, reconhecimento de padrões, estatística e outras áreas de Inteligência Artificial - para encontrar padrões nos dados [24], as demais etapas também têm grande importância no processo, pois um algoritmo aplicado a uma base de dados que contenha valores discrepantes e que não tenha passado por pré-processamento pode identificar padrões de dados sem sentido.

A variabilidade de funções e os benefícios trazidos pelas aplicações de Mineração de Dados vêm permitindo sua aplicação nas mais diversas áreas.

A área de *marketing* tem sido uma das mais exploradas pelo fato do crédito de clientes, faturamento e compras estarem entre as primeiras transações de negócios automatizadas com computadores, logo, tem-se uma grande quantidade de dados disponíveis [27]. Aplicações

recentes foram feitas no sentido de detectar o comportamento do mercado, como força de mercado, liderança de preços, fatias de mercado (*market share*) e futura demanda [71] [2].

Na área de finanças tem sido amplamente usada para encontrar potenciais padrões de transações fraudulentas em cartões de créditos dos consumidores [55] e para classificar clientes quanto ao risco associado em uma transação importante [48]

Na área de saúde e médica, Mineração de Dados tem sido usada tanto no sentido de administração de serviços a pacientes quanto no diagnóstico e tratamento de doenças [10]. Podem ser encontrados trabalhos para automatizar o diagnóstico de câncer cervical [67], câncer de mama [77], ataques cardíacos [69], entre outros.

Nota-se que muitas das tarefas citadas consistem em atribuir rótulos a um conjunto de dados (identificação de transações fraudulentas, determinação de um diagnóstico como positivo ou negativo), ou seja, executar uma tarefa de Classificação, explicada com mais detalhes na Seção 2.2.

2.1 Tarefas de Mineração de Dados

De forma geral, as tarefas de Mineração de Dados podem ser classificadas, segundo Ozekes e Osman[58], em duas categorias: preditivas ou descritivas. A mineração descritiva fornece informação para entender o que está acontecendo nos dados. A mineração preditiva permite que o usuário submeta registros com valores de campos desconhecidos e o sistema irá “adivinhar” o valor desconhecido baseado nos padrões previamente descobertos do banco de dados analisado.

Estas tarefas executadas na etapa de DM podem ainda ser classificadas de acordo com suas funções: Classificação, Predição, Agrupamento e Regras de Associação [23]:

- Classificação: tem a função de mapear (classificar) um item de dado em uma de várias classes pré-determinadas [79].
- Predição: tem a função de mapear (predizer) um item de dado em um valor contínuo [30].
- Agrupamento: tem a função de identificar um conjunto finito de categorias ou grupos que descrevem os dados [39]. Diferentemente da classificação e da predição, agrupamento analisa os dados sem consultar uma classe predefinida, pois estas classes não existem no início do método e o agrupamento é utilizado para gerar estas classes [30].

- Regras de Associação: tem a função de descobrir elementos que ocorrem em comum dentro de um determinado conjunto de dados [51].

Segundo Fayyad *et al.* [24], qualquer algoritmo de mineração de dados para atender alguma das tarefas descritas acima é formado por três componentes principais: modelo, método de avaliação e algoritmo de busca.

O modelo tem dois fatores relevantes: a sua função e a sua forma de representação. A função do modelo diz respeito ao tipo de tarefa objetivo do algoritmo (classificação, agrupamento, etc.). A forma de representação do modelo é a linguagem utilizada para descrever os padrões descobertos, os quais incluem árvores de decisão, regras de decisão, modelos lineares, modelos não-lineares, modelos baseados em exemplos, modelos de atributos relacionais, entre outros. O modelo de representação determina a flexibilidade do modelo na representação dos dados e a sua interpretabilidade pelo usuário. Tipicamente, quanto mais complexo o modelo, melhor este consegue representar os dados, porém, é menos interpretável [21].

O método de avaliação define um critério que determina o quão bem um modelo particular e seus parâmetros atingiram os objetivos definidos; para isto é utilizada uma medida que determina quanto o modelo está adaptado aos dados, o qual deve ser suave o bastante para evitar o efeito de *overfitting* (conceito melhor explicado no Capítulo 3) e ter um grau de liberdade o suficiente para representar fielmente os dados analisados. Um exemplo deste método pode ser o cálculo da acuidade de uma predição em um conjunto de testes, o qual foi utilizado nas comparações dos algoritmos de Classificação do Capítulo 5.

O algoritmo de busca consiste na especificação de um algoritmo para encontrar modelos ou parâmetros de funções. Encontrar os melhores parâmetros é muitas vezes traduzido a um problema de otimização (por exemplo, encontrar o máximo global de uma função não linear no espaço de parâmetros). A busca por um modelo em espaço de modelos é geralmente tratado como um algoritmo de busca gulosa [24], ou seja, o algoritmo sempre escolhe a solução que parece ser a melhor no momento, fazendo uma escolha local na expectativa de que esta escolha leve à solução ótima global [16].

2.2 Classificação

A Classificação é uma tarefa de Mineração de Dados que ocorre quando um objeto precisa ser designado a um grupo ou uma classe predeterminada, baseado nos seus atributos [84].

Vista ou tratada pela ótica de Banco de Dados, Han e Kamber [30] definem a tarefa de Classificação como um processo de duas etapas (Figura 2.2).

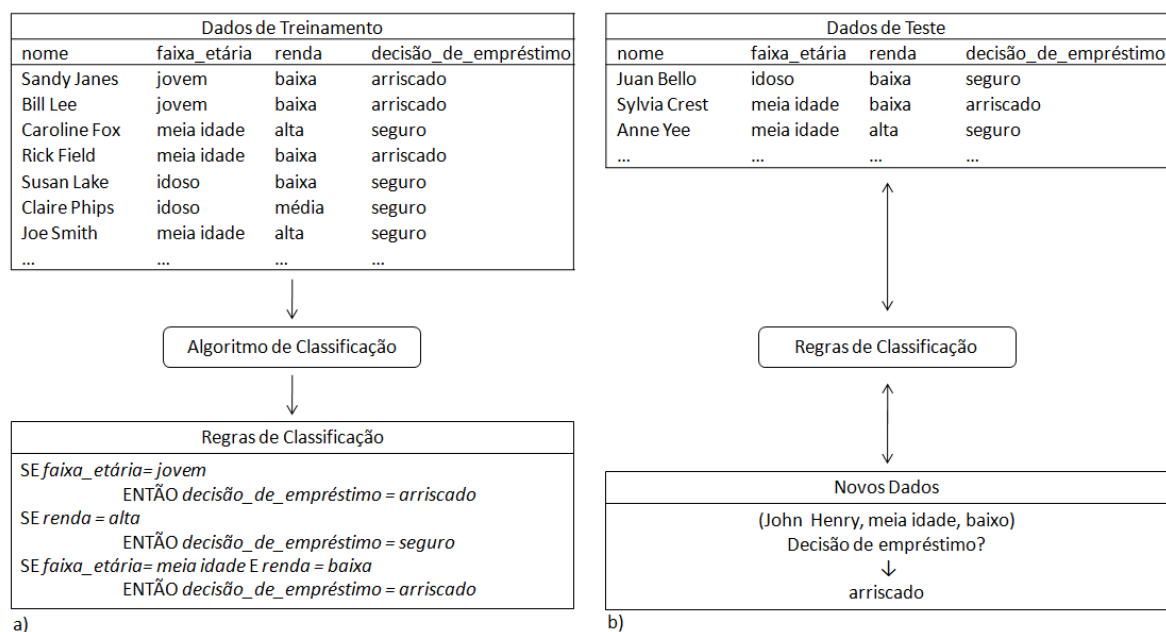


Figura 2.2: Tarefa de Classificação de Dados. a) Treinamento: algoritmo de classificação utiliza o conjunto de treinamento para gerar o modelo em forma de regras de associação, tendo que definir rótulo de classe do atributo *decisão_de_empréstimo*. b) Classificação: os dados de teste são utilizados para estimar a acuidade do classificador. Adaptado de [30]

Na primeira etapa, ou fase de treinamento, um classificador é construído para descrever um conjunto predeterminado de classes. Para isto, utiliza-se um algoritmo que "aprende" a partir do conjunto de treinamento, o qual é formado por conjunto de registros do banco de dados chamadas de registros de treinamento. Um registro¹, X , é definido por um vetor n -dimensional, que representa os atributos, $X = (x_1, x_2, \dots, x_n)$, e a cada registro X é associado a uma classe que também está definida por um atributo no banco de dados, chamado de rótulo de classe. Esta primeira etapa pode ser vista como o aprendizado de um mapeamento ou função, $y = f(X)$, que prediz um rótulo de classe y associado a um registro X . Desta forma, o que se deseja é aprender um mapeamento ou função que separa as classes de dados. Tipicamente, o mapeamento é representado na forma de regras de classificação, árvores de decisão ou fórmulas matemáticas.

¹ Embora registro, tupla, vetor de dados, vetor de características e padrão de entrada sejam sinônimos na literatura, neste trabalho será adotado o uso do termo registro.

A segunda etapa é onde o modelo vai ser efetivamente utilizado para a tarefa Classificação. Para isto a precisão de predição do classificador é avaliada. Esta avaliação é feita em um conjunto de testes, composto de registros de teste e respectivos rótulos de classe associados. Estes registros são selecionados randomicamente do conjunto de dados gerais e independentes dos registros de treinamento, ou seja, não são usadas para construir o classificador. O motivo de não utilizar o conjunto de treinamento para avaliação se deve ao fato que esta seria uma estimativa otimista, porque o classificador tende a adquirir as características específicas do conjunto de treinamento correndo assim o risco de sofrer um efeito chamado de *overfitting*, que é uma característica adquirida pelo classificador onde o mesmo perde a sua capacidade de generalização, incorporando alguma anomalia particular dos dados de treinamento que não está presente em outras partes do conjunto de dados, desta forma, o classificador prediz corretamente para os registros do conjunto de treinamento, mas quando utilizado para classificar registros que não fizeram parte do treinamento possui uma baixa acuidade.

Existem várias técnicas para executar a tarefa de Classificação, cada uma com suas características específicas. A subseção seguinte apresenta, de forma resumida, algumas destas técnicas, onde é possível verificar as diferenças de abordagem para um mesmo tipo de problema e objetivo.

2.2.1 Algoritmos para Classificação de Dados

Alguns dos algoritmos mais populares para Classificação são [58]: Árvores de Decisão, Algoritmos Genéticos, K-NN, Classificadores Bayesianos e Redes Neurais Artificiais. Alguns são brevemente descritos abaixo, outros podem ser consultados em [30] e [23].

Árvores de Decisão: é um tipo de fluxograma com estrutura de árvore onde os nós internos denotam um teste sobre um atributo, cada ramificação representa o resultado de um teste, e cada nó folha contém um rótulo de classe. Representa um tipo de algoritmo de aprendizado de máquina que utiliza a abordagem “dividir-para-conquistar” para classificar classes [73], onde o treinamento é mais rápido que outras técnicas e tem precisão similar quando comparados a outros métodos de classificação [81]. Han e Kamber [30] citam algumas características que fazem árvores de decisão ser um algoritmo bastante utilizado:

- facilidade de ser convertida para regras de classificação;
- capacidade de manipular dados de alta dimensionalidade;
- representação do conhecimento adquirido em forma de árvore, que é intuitivo e geralmente de fácil assimilação pelo usuário final.

Algoritmos Genéticos: é uma técnica de busca e otimização inspirada nos princípios de seleção natural e reprodução genética de Darwin [28]. Nestes princípios, a seleção privilegia os indivíduos mais aptos com maior longevidade, portanto, com maior probabilidade de reprodução. Indivíduos com mais descendentes têm mais chance de perpetuarem seus códigos genéticos nas próximas gerações, tais códigos genéticos constituem a identidade de cada indivíduo e estão representados nos cromossomos.

Estes princípios são imitados na construção de algoritmos computacionais que buscam uma melhor solução para um determinado problema através da evolução de populações de soluções codificadas por meio de cromossomos artificiais. Em Algoritmos Genéticos um cromossomo artificial é uma estrutura de dados que representa uma das possíveis soluções do espaço de busca do problema. Estes cromossomos são submetidos a um processo evolucionário que envolve avaliação, seleção, recombinação e mutação. Após vários ciclos de evolução a população deverá conter indivíduos mais aptos [60].

K-NN (*K-Nearest Neighbor*): é um método baseado no aprendizado por analogia, onde um registro de teste é comparado com um registro de treinamento quanto a similares entre si. Funciona da seguinte forma: suponha um conjunto D de registros de treinamento. Cada elemento de D é um registro $(x_1, x_2, \dots, x_n, c)$, onde c é a classe à qual pertence o registro (x_1, \dots, x_n) . O registro (x_1, \dots, x_n) pode ser vista como um ponto num espaço n -dimensional. Seja $R = (r_1, \dots, r_n)$ um novo registro, ainda não classificado. A fim de classificá-lo, calculam-se as distâncias de R a todos os registros de treinamento e consideram-se as k registros de treinamento mais próximos de R . Dentre estes k registros, verifica-se qual a classe que aparece com mais frequência. O registro R será classificado dentro desta classe mais frequente [4]. Esse método é bem custoso em termos computacionais quando aplicado a conjuntos de treinamento grandes.

Classificadores bayesianos: são classificadores estatísticos que classificam um objeto em uma determinada classe baseando-se na probabilidade deste objeto pertencer a esta classe. Classificadores Bayesianos supõem como hipótese de trabalho que o efeito do valor de um atributo não-classe é independente dos valores dos outros atributos, ou seja, o valor de um atributo não influencia o valor dos outros. Esta hipótese tem como objetivo facilitar os cálculos envolvidos na tarefa de classificação, produz resultados rapidamente e de grande correção quando aplicados a grandes volumes de dados, comparáveis aos resultados produzidos por árvores de decisão e Redes Neurais Artificiais [4].

Redes Neurais Artificiais: são estruturas inspiradas na neurobiologia, em especial nos níveis estruturais da organização do cérebro humano [32]. Pelo fato de ser um dos pontos de discussão deste trabalho, o próximo capítulo será dedicado a este assunto.

Muitas vezes somente aplicar os métodos citados às bases de dados não é suficiente para o processo de descoberta de conhecimento, os resultados precisam ser avaliados, comparados com o resultado de outras técnicas ou com a mesma técnica, porém, com parâmetros diferentes, a fim de verificar se os resultados foram bons, qual técnica foi melhor, entre outras características. Para isto, existem métodos de avaliação de classificadores, assunto que é abordado na subseção seguinte.

2.2.2 Métodos de Avaliação de Classificação

Classificadores podem ser validados ou comparados de acordo com vários critérios [30]:

Acuidade: refere-se à habilidade de um classificador prever corretamente o rótulo de classe de um novo dado (ou seja, de um registro sem a informação de rótulo de classe)

Velocidade: refere-se ao custo computacional envolvido na geração e uso de um dado classificador.

Robustez: habilidade de um classificador de fazer previsões corretas de dados, mesmo com ruídos ou valores faltando.

Escalabilidade: refere-se à habilidade de construir um classificador eficiente dado uma grande quantidade de dados.

Interpretabilidade: refere-se ao nível de entendimento e compreensão que é fornecido por um classificador. É muito subjetivo e, portanto, mais difícil de ser analisado.

Dentre estes critérios, a acuidade é um dos mais utilizados. Em termos matemáticos, corresponde à porcentagem registros do conjunto de testes que são classificadas corretamente pelo classificador. Pode-se falar também em taxa de erro de classificação, que é simplesmente definida como $1 - Acc(C)$, onde $Acc(C)$ é a acuidade do modelo classificador C .

Uma ferramenta útil para a avaliação da acuidade é a *matriz de confusão* que analisa o quão bem um classificador pode reconhecer registros de diferentes classes. Uma matriz de confusão de m classes é uma tabela, M , de tamanho m por m , onde uma entrada M_{ij} indica o número de registros da classe i que foi rotulada pelo classificador como classe j . Para um classificador ter boa acuidade, a maioria dos registros deve ser representada ao longo da diagonal principal da matriz de confusão, com o resto das entradas sendo igual a zero. A

tabela pode ter linhas e colunas adicionais para apresentar o total ou taxa de reconhecimento de cada classe.

Tabela 2.1: Exemplo de matriz de confusão.

		Classe Correta		
		A	B	C
Classe predita	A	3	1	0
	B	1	2	0
	C	0	0	4

A Tabela 2.1 traz um exemplo de matriz de confusão para um conjunto de 11 registros, onde os resultados mostrados podem ser interpretados da seguinte forma:

- 3 registros da classe A foram corretamente classificados;
- 1 registro da classe A foi erroneamente classificado como classe B;
- 1 registro da classe B foi erroneamente classificado como classe A;
- 2 registro da classe B foram corretamente classificados;
- 4 registro da classe C foram corretamente classificados;

Um tipo especial de matriz de confusão ocorre quando há apenas duas classes para classificar, classificação binária [18]. Neste tipo de matriz pode-se falar em termos de registros positivos *versus* registros negativos (Tabela 2.2) e nas seguintes definições:

Verdadeiro positivo: refere-se ao registro que foi corretamente rotulado pelo classificador.

Verdadeiro negativo: refere-se ao registro negativo que foi corretamente rotulado pelo classificador.

Falso positivo: refere-se ao registro negativo que foi incorretamente rotulado como positiva.

Falso negativo: refere-se registro positivo que foi incorretamente rotulado como negativa.

Tabela 2.2: Matriz de confusão para registros positivos e negativos.

		Classe classificada	
		C1	C2
Classe Atual	C1	verdadeiro positivo	falso negativo
	C2	falso positivo	verdadeiro negativo

A partir dos dados desta matriz de confusão pode-se definir a sensibilidade e a especificidade de uma classificação [3]. Sensibilidade é referida como taxa de reconhecimento de verdadeiros positivos, que corresponde à proporção de registros positivos que foram corretamente classificados. Especificidade é taxa de verdadeiros negativos, que corresponde à proporção de registros negativos que são corretamente identificados. Estas medidas estão definidas nas Equações 3.1 e 3.2.

$$\text{sensibilidade} = \frac{v_pos}{pos} \quad (3.1)$$

$$\text{especificidade} = \frac{v_neg}{neg} \quad (3.2)$$

onde:

v_pos : número de registros classificados corretamente como positivo (verdadeiro positivo);

pos : número de registros classificados como positivo (verdadeiro positivo + falso positivo);

v_neg : número de registros classificados corretamente como negativo (verdadeiro negativo);

neg : número de registros classificados como negativo (verdadeiro negativo + falso negativo);

Para garantir que o valor da acuidade de um classificador seja uma estimativa confiável utilizam-se algumas técnicas de avaliação, dentre as quais citam-se: *Holdout*, *Random Subsampling* e *k-fold-cross-validation* [30].

No método *Holdout*, os dados fornecidos para o algoritmo de classificação são randomicamente particionados em dois conjuntos independentes: conjunto de treinamento e conjunto de testes. Tipicamente, dois terços dos dados são alocados para o conjunto de treinamento e o restante é alocado para o conjunto de testes. O conjunto de treinamento é usado para gerar o modelo, o qual tem a acuidade estimada com o conjunto de teste (Figura 2.3). A estimativa é considerada pessimista porque só uma porção dos dados iniciais é usada para originar o modelo.

Random Subsampling é uma variação do método *Holdout* e se baseia em repetir o método *Holdout* várias vezes (iterações). A acuidade para este método é estimada como a média das acuidades de cada iteração.

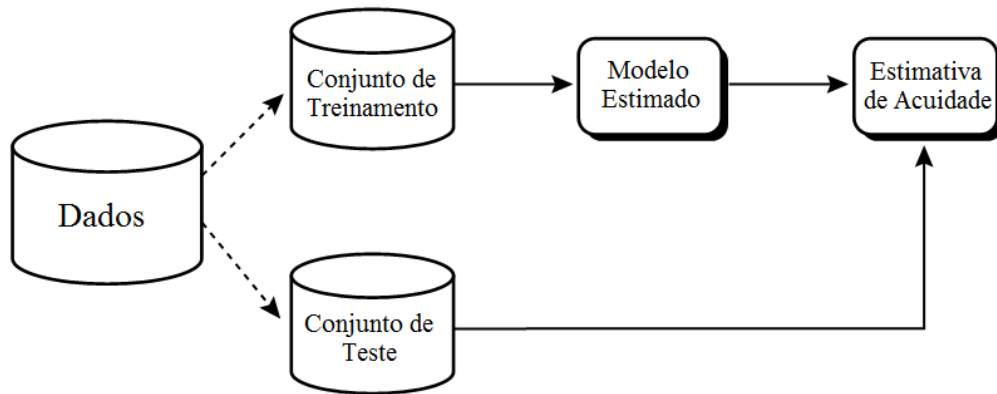


Figura 2.3: Estimativa de acuidade pelo método Holdout. Adaptado de [30].

No *k-fold-cross-validation* os dados são particionados em k subconjuntos exclusivos D_1, D_2, \dots, D_k , cada parte com tamanho aproximadamente igual. Treinos e testes são executados k vezes (iterações), em cada iteração i a partição D_i é reservada como conjunto de testes e os conjuntos restantes são usados para treinar o classificador. Para esta situação a acuidade é estimada pelo número total de classificações corretas das k iterações divididas pelo número total de registros no conjunto de dados inicial. Diferente de *Holdout* e de *Random Subsampling*, aqui cada registro do conjunto de dados é usado o mesmo número de vezes para treinar e uma vez para testar.

O método *k-fold-cross-validation* geralmente requer um esforço computacional maior que os demais métodos apresentados, porém tem sido amplamente utilizado para a estimativa da acuidade de classificadores, mais especificamente com k igual a 10 por apresentar pequenas variâncias e evitar o efeito de *overfitting* [29] [40]; devido a isto, será a abordagem utilizada nos testes realizados com Redes Neurais Artificiais no Capítulo 5.

Capítulo 3

Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs), também chamadas de redes conexionistas, processamento paralelo distribuído e computação neural, são caracterizadas por serem sistemas que, em algum nível, lembram a estrutura do cérebro humano [65].

Elas têm característica multidisciplinar (neurociência, matemática, estatística, física, ciência da computação e engenharia) o que tornou possível o desenvolvimento de muitos tipos RNAs e muitas aplicações envolvendo o seu uso, tanto na área acadêmica, industrial, comercial como de serviços [12]. Os tipos de aplicações são os mais diversos também, como modelagem, análise de séries temporais, reconhecimento de padrões, processamento de sinais de controle entre outros; tudo isto devido a uma importante propriedade: a de aprender a partir dos dados de entrada [32].

Segundo Haykin [32], as propriedades e capacidades que tornam RNAs potencialmente úteis são:

- Não-linearidade: um neurônio artificial pode utilizar funções lineares ou não-lineares.
- Adaptabilidade: capacidade da RNA adaptar suas estruturas a partir de modificações do meio ambiente.
- Mapeamento de Entrada-Saída: com base em exemplos de entrada e saída a RNA é capaz de se adaptar para minimizar o erro de mapeamento.
- Resposta a Evidências: capacidade de retornar não somente a resposta a uma entrada, mas também a crença ou confiança nesta decisão tomada.
- Informação Textual: o conhecimento que a RNA possui é representado pela própria estrutura e estado da rede.

- Tolerância a Falhas: quando implementada em forma física (hardware) é capaz de continuar a realizar computação robusta em situações de falha, tendo seu desempenho se degradando de forma suave.
- Implementação em VLSI: sua natureza paralela torna adequada a implementação utilizando tecnologia de integração em escala muito ampla (VLSI – *Very Large-Scale Integration*) que permite capturar o comportamentos realmente complexos de alguns problemas.
- Uniformidade de Análise e Projeto: a mesma notação é usada em todos os domínios que envolvem a sua aplicação, tornando possível o compartilhamento de teorias e algoritmos de aprendizagem em diferentes RNAs.
- Analogia Neurobiológica: engenheiros da área de RNAs olham para a neurobiologia procurando por novas idéias para resolver problemas mais complexos do que aqueles baseados em técnicas convencionais de projeto por conexões fixa, por outro lado, neurobiólogos olham para RNAs como uma ferramenta de pesquisa para interpretação de fenômenos neurobiológicos.

Nas seções seguintes serão apresentados os fundamentos gerais das RNAs que tornam possíveis as características apresentadas acima, bem como as características específicas de três tipos de RNA: MLP, RBF e LVQ, as quais serão implementadas para a ferramenta de mineração de dados proposta e posteriormente comparadas.

3.1 Fundamentos Redes Neurais Artificiais

Redes Neurais Artificiais são modelos matemáticos computacionais inspirados na estrutura neural de organismos inteligentes que adquirem conhecimento através da experiência, ou seja, através daquilo que se viveu, que se vivenciou. Haykin [32] define RNA como:

“Um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos:

- 1. O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.*
- 2. Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.”*

As unidades de processamento simples, ou o neurônio artificial, são baseados nas

características do neurônio biológico (Figura 3.1).

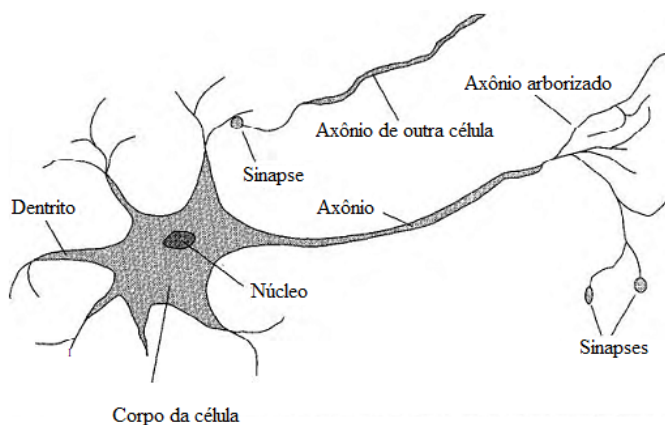


Figura 3.1: Célula nervosa ou neurônio biológico . Adaptado de [65].

Um neurônio biológico tem três partes principais: o corpo da célula, os dendritos e o axônio. Os dendritos são os responsáveis por receber os impulsos nervosos (a informação) provenientes de outros neurônios (neurônios pré-sinápticos) e conduzi-los até o corpo celular, onde a informação é processada e comparada gerando um novo impulso. Caso o percentual em um intervalo curto de tempo seja suficientemente alto faz o neurônio “disparar”. Este impulso passa então pelo axônio do neurônio até os dendritos de outros neurônios (neurônios pós-sinápticos). O ponto de contato entre a terminação do axônio de um neurônio e o dendrito de outro é chamado de sinapse. É este o local que une funcionalmente os neurônios, formando as RNAs, funcionando como válvulas capazes de controlar a transmissão nos impulsos nervosos através da rede. O efeito das sinapses é variável, que é o que dá a capacidade de adaptação dos neurônios. Este sistema simples, somado à operação em paralelo de bilhões de neurônios, é o responsável pela maioria das funções executadas pelo nosso cérebro [12].

O modelo matemático de um neurônio artificial pode ser observado na Figura 3.2, onde é possível identificar três elementos básicos:

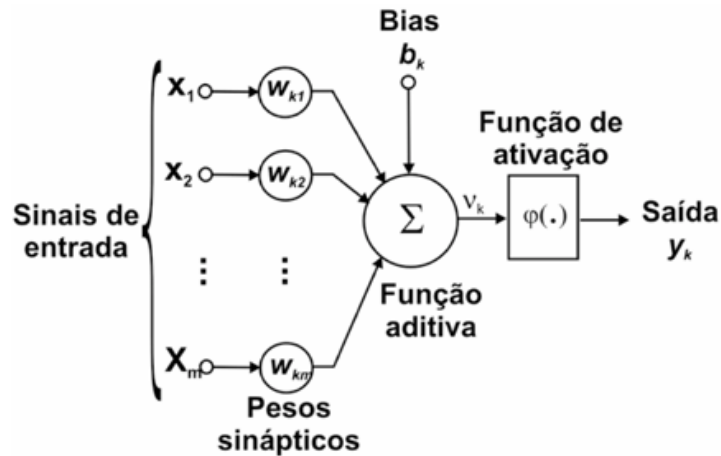


Figura 3.2: Modelo matemático de um neurônio artificial. Adaptado de [32].

1) um conjunto de sinapses, elos de conexão ou pesos sinápticos (w_{kj} , k se refere ao neurônio em questão, j ao terminal de entrada da sinapse ao qual o peso se refere): representa a força ou o peso da conexão, seu valor é multiplicado por um sinal x_j de entrada da sinapse j . Podem ser valores positivos ou negativos.

2) Um somador (Σ): combinador linear, pois faz a adição dos sinais de entrada ponderados pelos respectivos valores dos pesos sinápticos.

3) Uma função de ativação (φ): define o valor de saída do neurônio. Tipicamente no intervalo de $[0,1]$ ou $[-1,1]$.

Em termos matemáticos, pode-se escrever:

$$u_k = \sum_{j=1}^n w_{kj} x_j \quad (3.3)$$

$$v_k = u_k + b_k \quad (3.4)$$

$$y_k = \varphi(v_k) \quad (3.5)$$

onde:

b : valor *bias*, que tem o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo se é positivo ou negativo, respectivamente;

u : valor da combinação linear dos sinais de entrada ponderados pelo valor dos pesos sinápticos;

v : conhecido como campo local induzido ou potencial de ativação do neurônio k ;

y : sinal de saída do neurônio.

Nem sempre o modelo do neurônio artificial foi como o descrito acima, o primeiro modelo de RNA, conhecido como MCP, proposto por McCulloch e Pitts [49], é como o representado na Figura 3.3 abaixo.

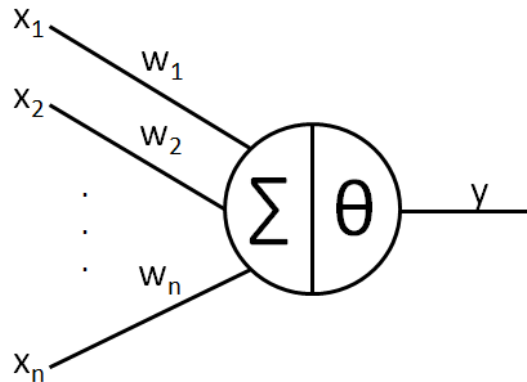


Figura 3.3: Modelo neural de McCulloch e Pitts [12].

onde:

- n : numero de entradas ou estímulos;
- w_i : peso associado à entrada x_i ;
- θ : limiar do neurônio.

O funcionamento deste modelo também era mais simples, apenas um dispositivo binário, cuja saída poderia ser pulso ou não-pulso (0 ou 1 , respectivamente), e as entradas tinham ganho arbitrário podendo ser excitatórias (positivos) ou inibitórias (negativos). Para determinar a saída do neurônio, calculava-se a soma ponderada das entradas com os respectivos fatores de ponderação, se este resultado fosse maior ou igual ao limiar θ então a saída do neurônio era pulso (1), caso contrário era não-pulso (0) [45]. Este modelo apresentava certas limitações, como apontado por Han e Kamber [30]:

- só conseguia implementar algumas funções booleanas e funções para dados linearmente separáveis², quando RNAs de apenas uma camada;
- pesos negativos eram mais adequados para representar disparos inibidores;

² Um conjunto de dados é dito linearmente separável se existir um hiperplano no espaço de entrada que separe os dados de classes diferentes [13]. Na geometria, um hiperplano pode ser um espaço vetorial, transformação afim ou o subespaço de dimensão $n-1$. Por exemplo, para um conjunto de dados tridimensional o hiperplano é um plano habitual, para um conjunto bidimensional o hiperplano é uma reta e para um conjunto unidimensional o hiperplano é um ponto.

- o modelo foi proposto com pesos fixos, não ajustáveis.

Atualmente, devido a novas descobertas na área computacional e na área neurológica, existem vários modelos de neurônios artificiais e estruturas de RNA os quais são capazes de aprender por meio do ajuste dos pesos sinápticos e implementar funções para dados não-linearmente separáveis, o que disponibiliza uma maior capacidade de processamento, possibilitando a resolução de problemas mais complexos, características essas que serão estudadas no decorrer desse capítulo.

3.1.1 Tipos de função de ativação

A função de ativação, representada por φ , define o valor de saída do neurônio tendo como parâmetro o campo local induzido v do neurônio. Apresentam-se abaixo quatro tipos básicos de função de ativação [12]:

- 1) Função limiar. Representada na Figura 3.4 e definida pela Equação 3.6.

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \quad (3.6)$$

Esta definição, segundo Haykin [32], descreve a propriedade “tudo-ou-nada” do modelo de neurônio de McCulloch e Pitts.

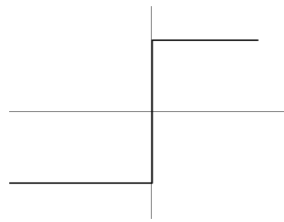


Figura 3.4: Representação de uma função limiar.

- 2) Função linear. Representada na Figura 3.5 e definida pela Equação 3.7.

$$\varphi(v) = \alpha v \quad (3.7)$$

Onde α é um número real que define a saída linear para os valores de entrada.

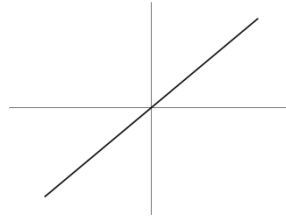


Figura 3.5: Representação de uma função linear.

3) Função linear por partes. Representada na Figura 3.6 e definida pela Equação 3.8.

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq +\frac{1}{2} \\ v & \text{se } +\frac{1}{2} > v > -\frac{1}{2} \\ 0 & \text{se } v \leq -\frac{1}{2} \end{cases} \quad (3.8)$$

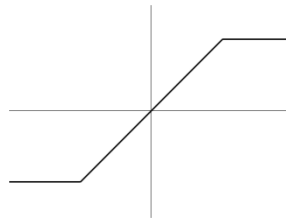


Figura 3.6: Representação de uma função linear por partes.

Uma função linear por partes é uma função que é linear até atingir os valores de saturação (valores máximos e mínimos de saída), para os demais valores se comporta como uma função linear.

4) Função sigmóide. Forma mais utilizada como função de ativação na construção de RNA. Também conhecida como *S-shape* (em forma de s), é uma função limitada, monotônica e que apresenta um balanceamento adequado entre comportamento linear e não-linear [12] [32]. Existem vários tipos de funções sigmóides, uma das mais importantes é a função logística representada na Figura 3.7 e definida pela Equação 3.9.

$$\varphi(v) = \frac{1}{1 + e^{-v}} \quad (3.9)$$

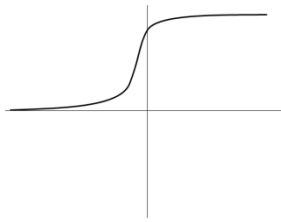


Figura 3.7: Representação de uma função sigmóide.

Além destas, RNAs podem utilizar várias outras funções de ativação, a exemplo da função identidade, que passa o valor do campo local induzido direto para a saída do neurônio, função tangente hiperbólica, também em forma de s , função seno, geralmente utilizada para dados distribuídos radialmente, função gaussiana, usadas principalmente em RNAs de base radial, entre outras. A escolha da função depende de qual tipo de RNA vai ser utilizada e o problema a ser resolvido.

3.1.2 Arquitetura de Redes Neurais Artificiais

A definição de uma arquitetura de RNA está diretamente ligada ao tipo de problema que a mesma deve resolver e com o algoritmo de treinamento que deve ser utilizado para treiná-la. Em geral, os seguintes itens fazem parte da definição da arquitetura de uma RNA: número de camadas da RNA, número de neurônios em cada camada, tipo de conexão entre os neurônios e conectividade [12].

Quanto ao número de camadas pode ser:

- RNA de camada única: entre qualquer sinal de entrada da rede e sua saída existe apenas um neurônio (Figura 3.8 a, e);
- RNA de múltiplas camadas: entre algum sinal de entrada e a saída da rede existe mais de um neurônio (Figura 3.8 b,c,d).

Quanto ao tipo de conexão entre os neurônios pode ser:

- *feedforward* (acíclica): a saída de um neurônio da camada i não pode ser usada como entrada em neurônios de camadas com índice igual ou menor que i (Figura 3.8 a,b,c);
- *feedback* (cíclica, recorrente [32]): a saída de um neurônio da camada i não pode ser usada como entrada em neurônios de camadas com índice igual ou menor que i (Figura 3.8 d, e).

Quanto à conectividade uma RNA pode ser:

- fracamente conectada: quando cada um dos nós de uma camada não está ligada a todos

os nós da camada posterior a sua (Figura 3.8 b, c, d);

- completamente conectada: quando cada um dos nós de uma camada (exceto, os da camada de saída) está ligado a todos os nós da camada posterior a sua. (Figura 3.8 a, e).

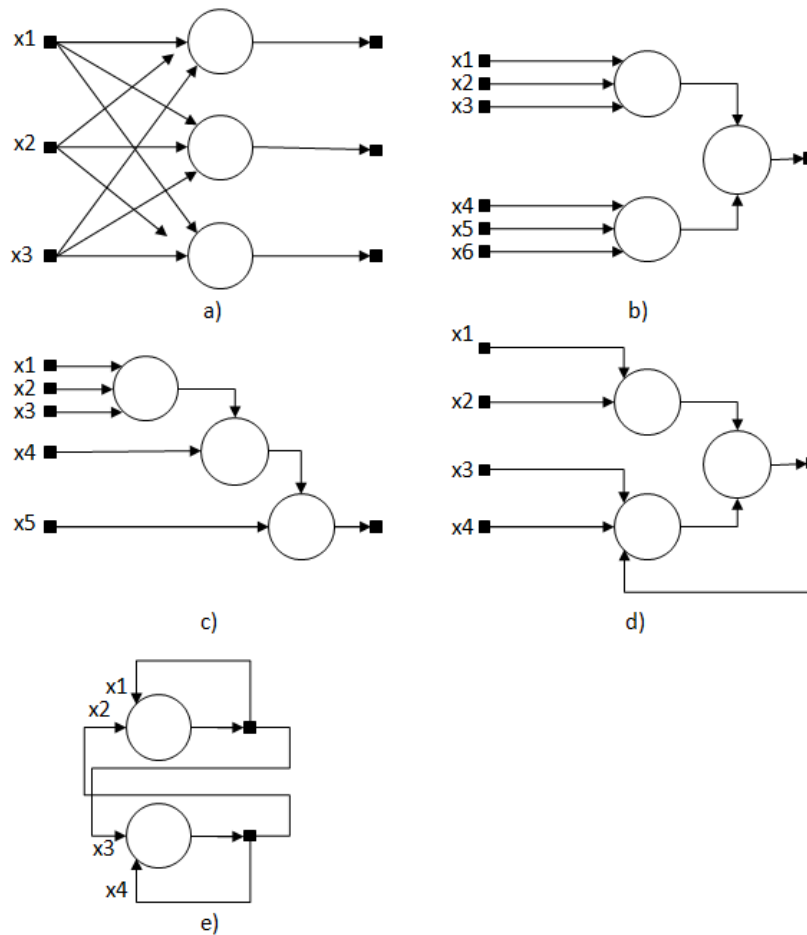


Figura 3.8: Arquiteturas de RNAs. Adaptado de [12].

3.1.3 Aprendizado

No contexto de RNAs define-se aprendizagem como [50]:

“... um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. O tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros ocorre.”.

De forma prática, esta definição se traduz no fato de que uma RNA aprende acerca do seu ambiente através de um processo iterativo de ajustes aplicados a seus pesos sinápticos e níveis

de *bias*, tornando-se mais instruída sobre seu ambiente após cada iteração do processo de aprendizagem [32].

Ao conjunto de procedimentos para adaptar os parâmetros da RNA para que a mesma possa aprender uma determinada função é chamado de algoritmo de aprendizado. Devido à quantidade de tipos de RNAs e tipos de problemas que as mesmas são utilizadas para executar, não existe um algoritmo único de aprendizado, e sim vários, cada um com suas vantagens e desvantagens, os quais diferenciam-se basicamente pela maneira na qual os ajustes dos pesos dos neurônios são feitos [12].

Os algoritmos de aprendizagem também podem ser classificados pela maneira com que se relacionam com seu ambiente: aprendizado supervisionado ou aprendizado não supervisionado. O paradigma do aprendizado supervisionado envolve a aprendizagem com o auxílio de um supervisor que fornece além da entrada a respectiva saída desejada para a RNA. O objetivo deste paradigma é fazer a correção dos pesos de acordo com a diferença entre a saída calculada pela RNA com a resposta esperada da respectiva entrada. O paradigma do aprendizado não-supervisionado envolve a aprendizagem sem o auxílio de um supervisor, onde a aprendizagem de padrões ocorre apenas com os valores de entrada fornecidos [65].

As três RNAs abordadas neste trabalho (MLP, RBF e LVQ) utilizam o paradigma de aprendizado supervisionado, por isto, descreve na subseção seguinte uma das principais regras de aprendizado deste paradigma, a aprendizagem por correção de erros [12], que está relacionada com os algoritmos de aprendizagem destas RNAs.

3.1.4 Aprendizado por correção de erros

A técnica da aprendizagem por correção de erros, também referida como regra delta ou regra de Widrow-Hoff [83], procura minimizar o erro entre resposta atual da RNA e a saída desejada. Este erro é definido pela equação:

$$e(t) = d(t) - y(t) \quad (3.10)$$

onde:

$e(t)$ = sinal de erro;

$d(t)$ = saída desejada;

$y(t)$ = saída real da rede.

Para a correção do erro, os pesos da RNA devem ser ajustados, de forma a aproximar a saída real à saída desejada. Os ajustes dependerão do próprio erro calculado, do valor do estímulo de entrada que é transmitido pelo peso a ser ajustado e também da taxa de aprendizado, a qual define o quão rápido o treinamento deve aprender. Para um dado neurônio k , um estímulo de entrada i e um passo de treinamento t tem-se:

$$\Delta w_{ki}(t) = \eta e_k(t) x_i(t) \quad (3.11)$$

$$w_i(t+1) = w_i(t) + \Delta w_{ki}(t) \quad (3.12)$$

onde:

$\Delta w_{ki}(t)$: valor de ajuste a ser acrescentado ao peso w_{ki} ;

η : taxa de aprendizado;

$e_k(t)$: valor do erro para o neurônio k ;

$x_i(t)$: valor do estímulo na entrada i do neurônio.

A dedução destas equações envolve a minimização da função de custo, definida como a soma dos erros quadráticos das saídas, conforme apresentado na Equação 3.13.

$$\varepsilon(t) = \frac{1}{2} e_k^2(t) \quad (3.13)$$

Com relação à superfície de erro obtida pela da Equação 3.13, dependente do tipo de neurônios de processamento utilizados na construção da RNA, se a RNA é formada inteiramente por unidades lineares, a superfície de erro possuirá um único valor mínimo. Por outro lado, se a rede é constituída por unidades não-lineares, podem ser encontrados diversos valores mínimos, chamados de mínimos locais, além do mínimo global [12].

3.2 Redes *Multilayer Perceptron* – MLP

Uma Rede MLP consiste de uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída (Figura 3.9). É do tipo *feedforward*, ou seja, nenhuma saída de um neurônio de uma camada k será sinal de entrada para um neurônio de uma camada menor ou igual a k , e é completamente conectada, tal que cada neurônio fornece sua saída para cada unidade da camada seguinte [30].

As funções de ativação dos neurônios devem ser não-lineares e diferenciáveis, ou seja, o gráfico da função não pode ser uma reta e deve ser possível calcular a derivada da função. A não linearidade serve para separar padrões que não são linearmente separáveis, a diferenciação permite o cálculo do gradiente da função, direcionando assim o ajuste dos pesos do neurônio durante o treinamento, uma função não-linear muito utilizada para esta RNA é a função logística (3.9) [12].

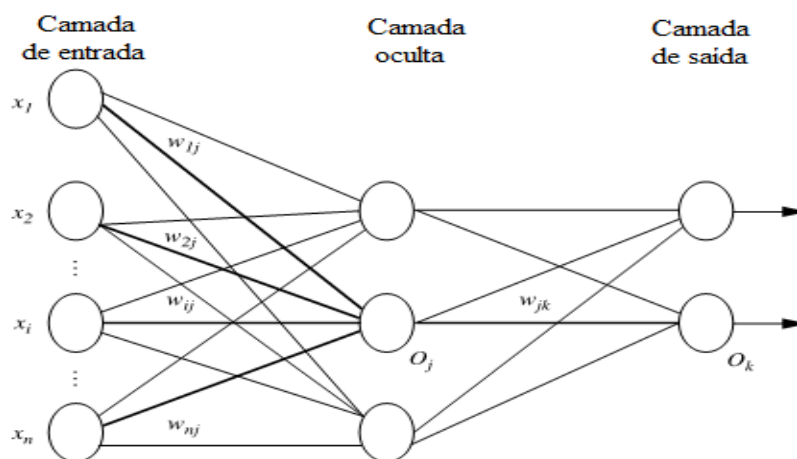


Figura 3.9: Arquitetura Rede MLP [30].

A estrutura ideal de uma Rede MLP já foi objeto de estudo de muitos autores, a exemplo de [20] [34] [26]. Os resultados obtidos têm mostrado que uma camada intermediária é o suficiente para aproximar qualquer função contínua e duas camadas intermediárias são suficientes para aproximar qualquer função matemática, sendo que o número de neurônios por camada oculta deve ser definido empiricamente [19] [20] [33]. Embora em alguns casos a utilização de mais de duas camadas intermediárias possa facilitar o treinamento, sua utilização não é recomendada, pois o erro propagado através da RNA se torna menos útil e preciso [12].

À medida que a quantidade de camadas aumenta em uma RNA multicamadas também fica mais complexo determinar qual a função implementada pela RNA e qual o papel de cada camada, o que se pode afirmar é que elas definem como é realizada a divisão do espaço de decisão. Para uma RNA com duas camadas ocultas pode-se dizer que ocorre o seguinte processamento [12]:

- Na primeira camada intermediária cada neurônio traça retas no espaço de padrões de treinamento;

- Na segunda camada intermediária cada neurônio combina as retas traçadas pela camada anterior conectados a ele, formando regiões convexas, onde o número de lados é definido pelo número de unidades a ele conectadas (Figura 3.10);
- Na camada de saída cada neurônio forma regiões que são combinações das regiões convexas formadas pela camada anterior, definindo, desta forma, regiões com formatos abstratos (Figura 3.11).

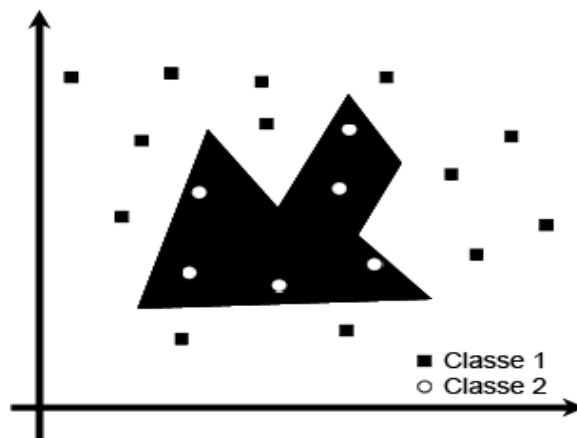


Figura 3.10: Regiões definidas pelo processamento da segunda camada intermediária de uma Rede MLP de duas camadas ocultas. Adaptado de [12].

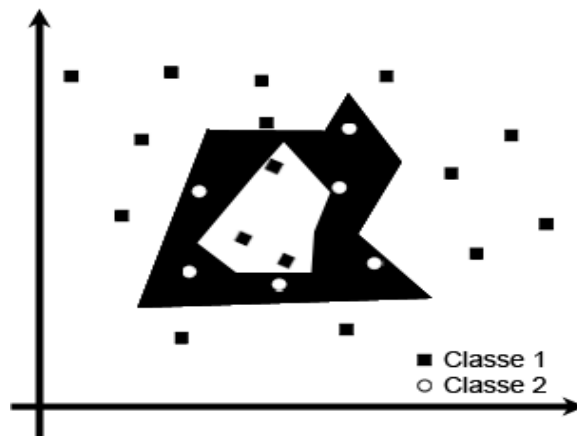


Figura 3.11: Regiões definidas pelo processamento da camada de saída de uma Rede MLP de duas camadas ocultas. Adaptado de [12].

Estas características, juntamente com a habilidade de aprender através de treinamento, fazem com que as Redes MLP apresentem um poder computacional muito maior que as RNAs sem camadas ocultas.

3.2.1 Treinamento

O algoritmo mais conhecido e utilizado para o treinamento de Redes MLP é o algoritmo *back-propagation* [64], um algoritmo com paradigma supervisionado cujo treinamento ocorre em duas fases: a fase *forward* e a fase *backward*.

Na fase *forward* a entrada é apresentada à primeira camada da RNA, que calcula seus sinais de saída e passa os valores para a camada seguinte. Esta camada calcula seus sinais de saída e passa para a próxima camada, isto vai acontecendo até a camada de saída calcular as saídas da RNA, as quais são comparadas com as saídas desejadas.

A fase *backward* percorre o caminho inverso. A partir da camada de saída até a camada de entrada os pesos dos neurônios vão sendo ajustados de forma a diminuir seus erros (os erros dos neurônios das camadas intermediárias são calculados utilizando o erro dos neurônios da camada seguinte ponderado pelo peso da conexão entre eles). Este processo é repetido até atingir algum critério de parada.

O algoritmo *backpropagation* é baseado na regra delta, ou aprendizagem por correção de erros descrita anteriormente na Subseção 3.1.4, sendo por isto chamada de regra delta generalizada. Desta forma, também se requer que as funções de ativação utilizadas pelos nós da RNA sejam contínuas, diferenciáveis e não-decrescentes [12]. Uma descrição passo-a-passo do algoritmo *backpropagation*, de uma forma bem útil para implementação, pode ser verificada no Algoritmo 3.1, onde D é um conjunto de dados consistindo de registros de treinamento e seus valores alvo associados, e l é taxa de aprendizado.

O algoritmo *backpropagation* tem sido aplicado com sucesso em várias situações e atingiu uma enorme popularidade, porém muitos problemas ainda o acompanham, entre eles: um longo período de treinamento, a possibilidade de ficar preso em um mínimo local na superfície de erro e não aprender com os dados de treinamento [59]. Devido a estas deficiências algumas soluções foram propostas, sendo que Rumelhart [63] sugeriu a adição de um termo, chamado *momentum*, à equação da regra delta mostrada na Equação 3.12, que possibilitou diminuir o tempo de treinamento e assegurar uma maior estabilidade para a RNA evitando que a mesma fique presa em um mínimo local. A Equação 3.12 com a adição do termo *momentum* está definida na Equação 3.14 e foi utilizada na implementação do algoritmo *backpropagation* para este trabalho.

$$w_i(t+1) = w_i(t) + \Delta w_{ki}(t) + \alpha(w_i(t) + w_i(t-1)) \quad (3.14)$$

onde α é a constante *momentum*.

Algoritmo 3.1: Algoritmo *Backpropagation*. Adaptado de [30].

1. Inicialize todos os pesos e *bias* na rede;
 2. Enquanto condição de término não é satisfeita
 3. Para cada registro de treinamento X de D faça
 - 3.1. Para cada neurônio j da camada de entrada faça
 - 3.1.1. $O_j = I_j$;
 - 3.2. Para cada neurônio j das camadas ocultas e da camada de saída faça
 - 3.2.1. $I_j = \sum_i w_{ij} O_i + \theta_j$;
 - 3.2.2. $O_j = \frac{1}{1 + e^{-I_j}}$;
 - 3.3. Para cada neurônio j da camada de saída faça
 - 3.3.1. $Err_j = O_j (1 - O_j) (T_j - O_j)$;
 - 3.4. Para cada neurônio j das camadas ocultas, a partir da última para a primeira faça
 - 3.4.1. $Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk}$;
 - 3.5. Para cada peso w_{ij} da rede faça
 - 3.5.1. $\Delta w_{ij} = (l) Err_j O_i$;
 - 3.5.2. $w_{ij} = w_{ij} + \Delta w_{ij}$;
-

3.3 Redes *Radial Basis Function* - RBF

Uma RNA similar à MLP é a Rede RBF (*Radial Basis Function Network*). Também constituída de várias camadas e que utiliza aprendizado supervisionado. Sua principal característica é a utilização de funções de base radial em todos os nós da camada oculta, que, ao invés de utilizar como argumento de função o produto escalar entre os valores do registro de entrada e os valores do registro de pesos do neurônio, utiliza a distância entre os valores do registro de entrada e o seu centro [12]. Como a Rede MLP, a Rede RBF é uma das RNAs que têm sido aplicadas com sucesso para tarefas de Classificação [74].

A Rede RBF na sua forma mais básica (Figura 3.12) consiste de uma camada de entrada, que conecta a RNA ao seu ambiente, apenas uma camada oculta, que aplica uma

transformação não-linear do espaço de entrada para um espaço oculto de alta dimensionalidade, e a camada de saída, que aplica uma transformação linear no espaço oculto fornecendo uma saída para a rede [32]. Além disto, Redes RBF também são do tipo *feedforward* e são completamente conectadas.

A justificativa de uma transformação não-linear seguida de uma transformação linear baseia-se no Teorema de Cover [17] sobre a separabilidade de padrões, onde o mesmo afirma que um problema complexo de classificação de padrões dispostos não-linearmente em um espaço de alta dimensionalidade tem maior probabilidade de ser linearmente separável do que em um espaço de baixa dimensionalidade.

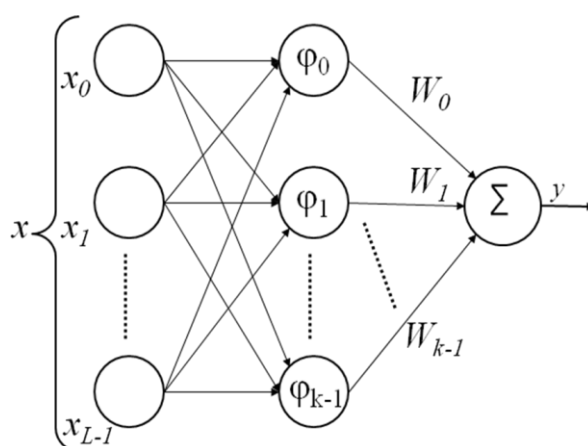


Figura 3.12: Arquitetura Rede RBF.

x : registro de entrada

φ : funções de base radial

W_i : pesos do neurônio da camada de saída

y : saída da rede

A função de cada camada da RNA é bem específica. A camada de entrada, cujos nós utilizam funções de base radial, agrupa os dados em grupos (*clusters*), transformando o conjunto dos padrões de entrada não linearmente separáveis em um conjunto de dados linearmente separáveis. A camada de saída da RNA procura classificar os padrões da camada anterior, o fato de serem linearmente separados, possibilita a utilização de operações simples nesta camada, a exemplo de uma combinação linear entre os valores de saída da camada oculta [12].

Embora a Figura 3.12 apresente apenas um neurônio de saída ela pode ter mais, sendo que a saída de cada um pode ser dada de acordo com a Equação 3.15.

$$y = w_0 + \sum_{i=1}^{n_k} w_i f(\|x - c_i\|) \quad (3.15)$$

onde: f são funções de base radial, w_i são os pesos do neurônio da camada de saída, w_0 é o *bias* do neurônio, x é o registro de entrada da função, c_i são os centros associados com as funções de base radial, n_k é o número de funções de base radial na rede e $\| \cdot \|$ denota a distância Euclidiana.

Funções de base radial são classes especiais de funções cujo valor aumenta ou diminui em relação à distância de um ponto central (Figura 3.13). Diferentes funções têm sido utilizadas na construção de Redes RBF, porém as mais comuns são [12]:

- Função gaussiana:

$$\varphi(u) = \exp\left(-\frac{v^2}{2\sigma^2}\right) \quad (3.16)$$

- Função multiquadrática:

$$\varphi(u) = \sqrt{(v^2 + \sigma^2)} \quad (3.17)$$

- Função *thin-plate-spline*:

$$\varphi(u) = v^2 \ln(v^2) \quad (3.18)$$

onde $v = \|x - \mu\|$ (distância euclidiana), x é o registro de entrada e μ e σ representam o centro e a largura da função radial, respectivamente.

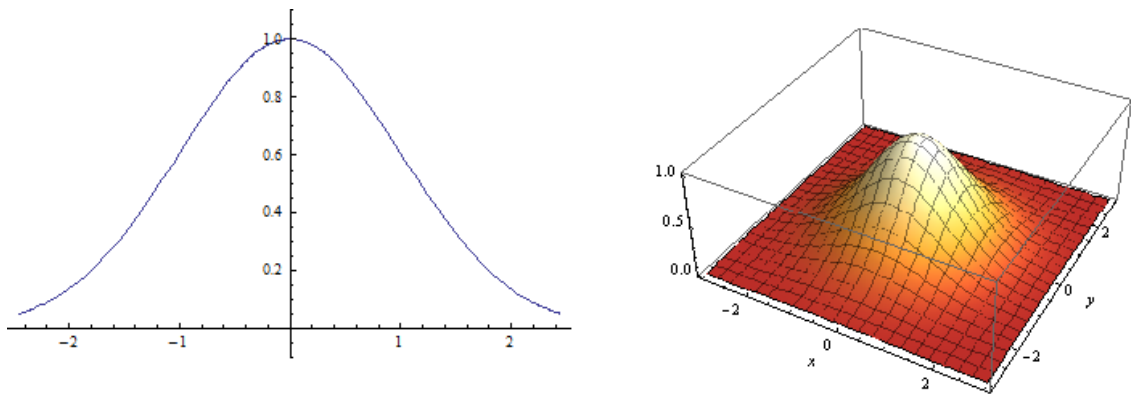


Figura 3.13: Gráfico de função gaussiana para entrada unidimensional (esquerda) e bidimensional (direita).

A principal diferença entre a operação de uma Rede RBF para uma Rede MLP pode ser observada de forma gráfica na Figura 3.14, que simula uma classificação hipotética de pacientes psiquiátricos. Enquanto MLP separa os dados por hiperplanos, RBF agrupa os dados em um número finito de regiões elipsóides, o que possibilita que uma Rede MLP tenha maior capacidade de generalização para regiões onde não há dados de treinamento [61].

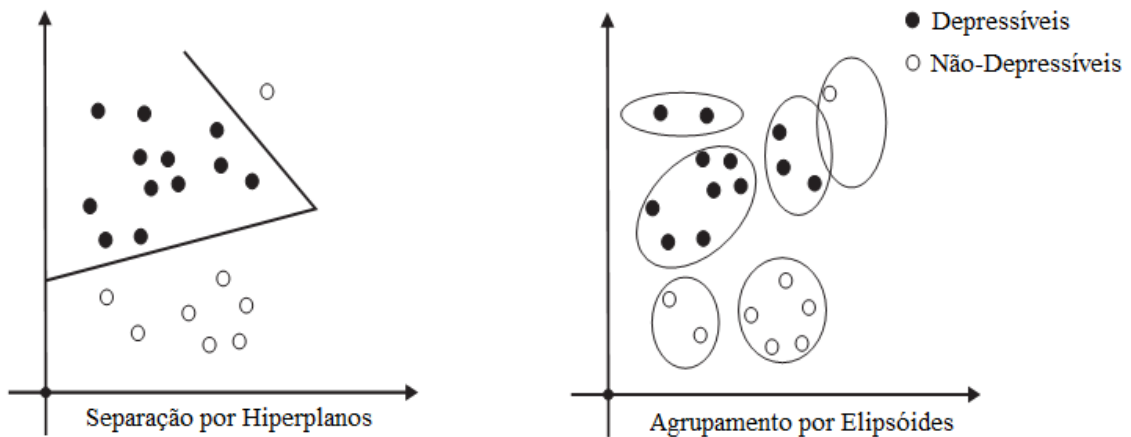


Figura 3.14: Classificação por Rede MLP (esquerda) e por Rede RBF (direita). Adaptado de [61]

3.3.1 Treinamento

O aprendizado em uma Rede RBF consiste em determinar os valores para o centro das funções de base radial, a largura das funções e os pesos da camada de saída. Para alguns algoritmos estes valores podem ser calculados separadamente, em mais de uma etapa, enquanto em outros todos os parâmetros são encontrados em conjunto, em uma mesma etapa [36].

Um algoritmo utilizado para a determinação de todos os parâmetros na mesma etapa é o

algoritmo supervisionado conhecido como Gradiente Estocástico [82] que utiliza a mesma abordagem do algoritmo *backpropagation* para a Rede MLP, ou seja, utiliza o erro entre a saída calculada da RNA e a saída desejada para ajustar todos os parâmetros da RNA de modo a minimizar o erro calculado. Porém, a abordagem mais comumente utilizada, a qual é adotada neste trabalho, baseia-se no procedimento descrito por Moody e Darken [52], onde os autores separam o treinamento de uma Rede RBF em três etapas executadas seqüencialmente. Na primeira etapa, utiliza-se um algoritmo não-supervisionado para encontrar os centros das funções de base radial, na segunda etapa utilizam-se métodos heurísticos para determinar a largura de cada função e, por fim, na terceira etapa, utiliza-se um algoritmo supervisionado para a determinação dos pesos da camada de saída da rede.

Alguns algoritmos não-supervisionados conhecidos para a determinação dos centros das funções de base radial são o SOM de Kohonen [41], o ART [14] e *k-means* [31], este último implementado neste trabalho. O funcionamento básico do algoritmo *k-means* é descrito no Algoritmo 3.2.

Algoritmo 3.2: Algoritmo *k-means*

1. Associe os primeiros K registros de entrada aos K centros
 2. Para cada novo registro de entrada faça
 - 2.1. Associe ao grupo mais próximo
 3. Calcule as novas posições dos centros
 4. Repita
 - 4.1. Para cada registro de entrada faça
 - 4.1.1. Calcule a distância para todos os centros
 - 4.1.2. Se o registro não pertence ao grupo mais próximo
 - 4.1.2.1. Troque o seu grupo pelo grupo mais próximo
 - 4.1.2.2. Recalcule o centro do grupo que perdeu e do grupo que ganhou o registro
-

Após a escolha dos centros o próximo passo é determinar o valor da largura de cada função de base radial, que pode ser feito também utilizando várias técnicas, sendo mais comuns as apresentadas a seguir [6]:

- 1) As funções utilizam um único valor de largura que é utilizado por todos os neurônios.

Neste caso, uma estratégia comum consiste em fazer a largura da função igual a uma fração da maior distância entre os centros de todos os neurônios, como definido na Equação 3.19.

$$\sigma = \frac{\text{dist}_{\max}(\mu_i, \mu_j)}{\sqrt{2q}}, \forall i \neq j \quad (3.19)$$

em que $\text{dist}_{\max}(\mu_i, \mu_j) = \max\{\|\mu_i - \mu_j\|\}$

2) Cada neurônio possui uma largura própria, que tem seu valor definido como metade da distância entre o centro do neurônio i e o centro mais próximo, como definido na Equação 3.20.

$$\sigma_i = \frac{\text{dist}_{\min}(\mu_i, \mu_j)}{2}, \forall i \neq j \quad (3.20)$$

em que $\text{dist}_{\min}(\mu_i, \mu_j) = \min\{\|\mu_i - \mu_j\|\}$

3) Esta heurística é uma variante do Caso 2, onde a largura da i -ésima função é a distância média do centro desta função para os K ($1 \ll K \ll q$) centros mais próximos. Em termos formais é definida de acordo com a Equação 3.21.

$$\sigma_i = \frac{\sum_{k=1}^K \text{dist}(\mu_i, \mu_{i_k})}{K} \quad (3.21)$$

onde i_k é o índice do k -ésimo centro mais próximo de μ_i , ou seja:

$$\|\mu_i - \mu_{i_1}\| < \dots < \|\mu_i - \mu_{i_k}\| < \dots < \|\mu_i - \mu_{i_K}\|$$

Para a terceira etapa uma das abordagens mais comuns consiste em utilizar as regras de aprendizagem usadas por RNAs de uma única camada, tais como a regra de aprendizagem do *Perceptron* Simples, a regra LMS (Adaline) ou ainda a regra Delta (*Perceptron* Logístico) [6]. Durante esta terceira etapa, os centros e os raios calculados nas duas etapas anteriores não tem os valores alterados. A regra de aprendizagem utilizada para este trabalho foi a regra Delta descrita na Subseção 3.1.4.

3.4 Redes *Learning Vector Quantization* – LVQ

A RNA *Learning Vector Quantization* (LVQ) é uma versão supervisionada de quantização vetorial³ baseada em trabalhos de Kohonen [42]. Ela é composta por três camadas: a camada de entrada, a camada oculta e a camada de saída (Figura 3.15). A camada de entrada é completamente conectada com a camada oculta, enquanto a camada oculta é parcialmente conectada com a camada de saída. Esta RNA apresenta valor fixo igual a 1 para os pesos das conexões entre a camada oculta e a camada de saída e valores binários (0 e 1) como saída dos neurônios da camada oculta e da camada de saída [76]. Ao receber um sinal na camada de entrada a rede transmite os valores para a camada oculta, a qual passa a resposta para a camada de saída, sendo assim esta RNA também é do tipo *feedforward*.

A camada oculta também é conhecida camada competitiva, pelo fato de que quando um registro de entrada é apresentado a rede, apenas um neurônio responderá a este sinal, o qual é denominado neurônio vencedor, e será o único que terá como saída o valor 1, os demais neurônios da camada terão como saída o valor 0 [68]. A definição do neurônio vencedor é estabelecida em termos da menor distância entre o registro de entrada de dados e o registro de pesos dos neurônios da camada oculta, esta distância é calculada pela distância Euclidiana, cuja fórmula esta definida na Equação 3.22.

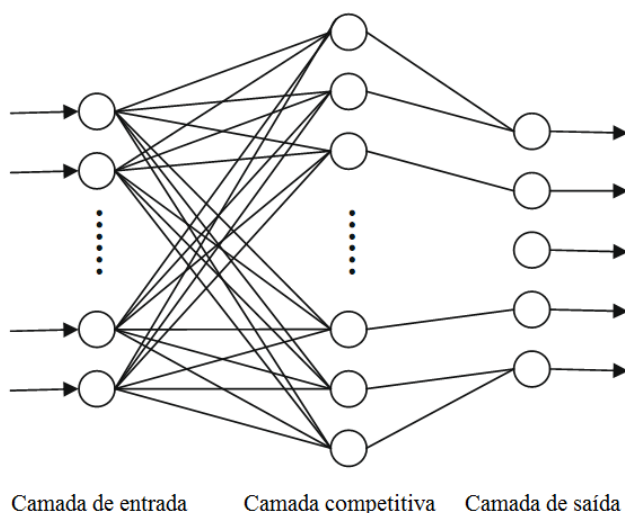


Figura 3.15: Arquitetura Rede LVQ. Adaptado de [76]

³ Quantização vetorial é uma técnica de quantização clássica de processamento de sinais que permite a modelagem de funções de densidade de probabilidade através da distribuição de vetores protótipos, originalmente utilizada para compressão de dados. [46]

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad (3.22)$$

A camada de saída de uma Rede LVQ faz uma transformação linear dos sinais recebidos da camada oculta para a classificação alvo definida pelo usuário.

3.4.1 Treinamento

Como citado anteriormente os pesos entre a camada oculta e a camada de saída são fixos, portanto, o treinamento em uma RNA do tipo LVQ baseia-se apenas em determinar os valores dos pesos entre a camada de entrada e a camada oculta. Este treinamento é feito através de um processo chamado aprendizagem por quantização vetorial que é uma técnica de aprendizagem competitiva e supervisionada que usa a informação sobre as classes dos registros de entrada para mover levemente os registros de peso de modo a melhorar a qualidade das regiões de decisão de um classificador [12]. O objetivo deste aprendizado é analisar o espaço de dados de entrada e dividi-lo em regiões distintas, definindo um vetor protótipo (também chamado vetor de reconstrução) para cada uma destas regiões, as quais são chamadas de células de Voronoi (Figura 3.16). Os pontos de dados de entrada que pertencem a determinada célula são definidos pela sua distância ao vetor protótipo baseado na distância euclidiana [32].

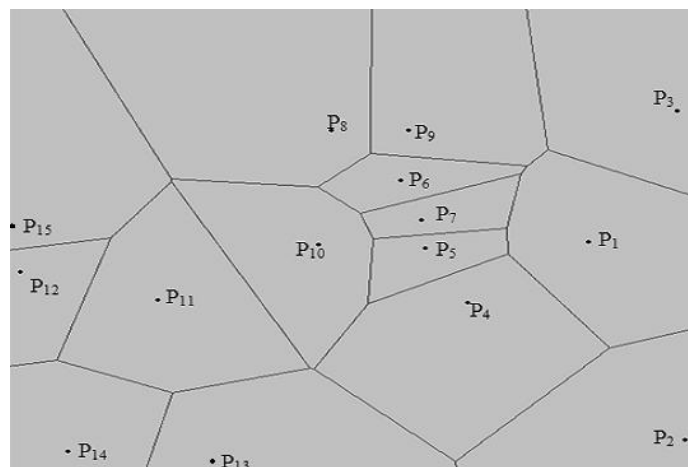


Figura 3.16: Diagrama de Voronoi. [78].

Durante o processo de aprendizado cada registro de entrada de dados é comparado com os vetores protótipos por meio de uma medida de similaridade. Se o rótulo de classe do registro de entrada de dados é igual ao rótulo do vetor protótipo mais a próximo a ele, este vetor é

deslocado em direção a localização do registro de dados, caso contrário o movimento se dá para a direção contrária [32]. A escolha do vetor mais próximo ao registro de entrada, dentro todo o conjunto de vetores protótipo caracteriza o aprendizado competitivo e o uso da informação de classe para direcionar o aprendizado caracteriza a supervisão.

Seja $X = \{x_1, x_2, x_3, \dots, x_n\}$ o conjunto de dados de entrada utilizado para treinamento da Rede LVQ; α a taxa de aprendizado da rede; $W = \{w_1, w_2, w_3, \dots, w_m\}$ o conjunto de vetores protótipos; $R = \{r_1, r_2, r_3, \dots, r_p\}$ o conjunto de rótulos aos quais estão associados a cada um dos registros de dados e dos vetores protótipos, onde r_j se refere ao rótulo do vetor j ; t um contador que indica em qual iteração se encontra o treinamento. A partir destes dados apresentados e do processo descrito acima pode-se, resumidamente, descrever o algoritmo LVQ pelo Algoritmo 3.3, extraído de [42].

Algoritmo 3.3: Algoritmo de treinamento LVQ.

1. Inicialize o conjunto de protótipos (neurônios) W aleatoriamente (ou use algum conhecimento *a priori*);
 2. Determine a classe r de cada vetor protótipo w ;
 3. Determine o coeficiente de aprendizado α ;
 4. Enquanto condição de parada é falsa faça
 - 4.1. Para todos os dados j , com $j=1,2,3..n$ faça
 - 4.1.1. Encontre o protótipo vencedor w_{I_v} , onde: $I_v = \arg \min_{i=1}^m d(w_i, x_j)$
 - 4.1.2. Se $(r_x = r_{w_{I_v}})$ faça
 - 4.1.2.1. $w(t+1) = w(t) + \alpha \| x_j - w_{I_v} \|$
 - 4.1.3. Se $(r_x \neq r_{w_{I_v}})$ faça
 - 4.1.3.1. $w(t+1) = w(t) - \alpha \| x_j - w_{I_v} \|$
 - 4.2. Atualize o coeficiente de aprendizado α
-

Este processo pode ser visualizado geometricamente na Figura 3.17, onde é possível verificar o comportamento dos vetores envolvidos durante a aprendizagem:

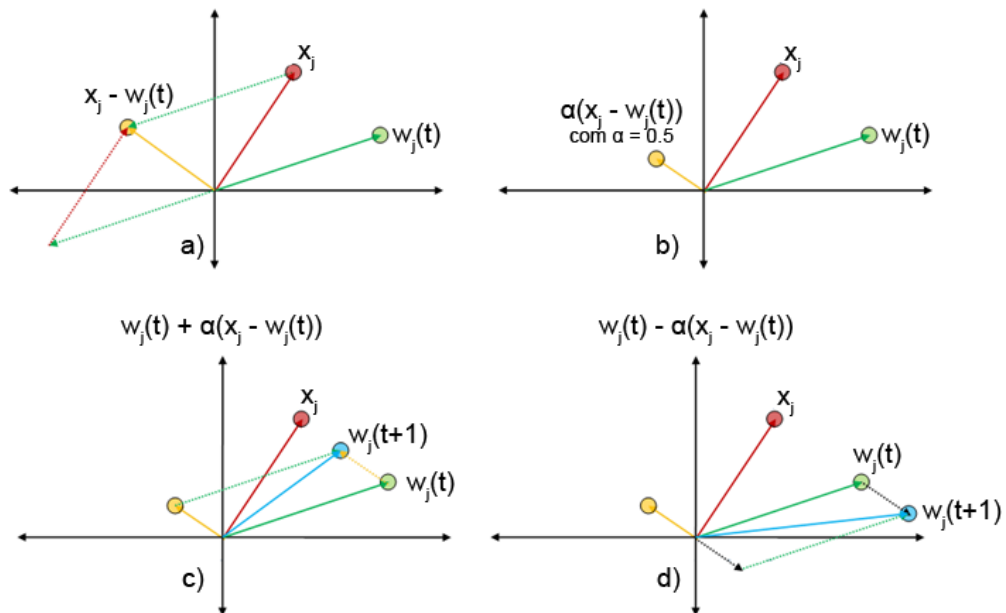


Figura 3.17: Comportamento geométrico dos vetores envolvidos no algoritmo LVQ [62].

- A Figura 3.17(a) representa a operação de subtração $(x_j - w_j)$ na iteração t , que determina o deslocamento máximo que o vetor protótipo pode ter;
- A Figura 3.17(b) representa o vetor resultante da multiplicação do coeficiente de aprendizado pelo vetor da subtração apresentado na Figura 3.17(a). Este vetor determina efetivamente o tamanho do deslocamento que o vetor protótipo sofrerá;
- A Figura 3.17(c) mostra o efeito da adição do vetor resultante da etapa anterior pelo vetor protótipo, nota-se que ocorreu uma aproximação entre o registro de entrada e o vetor protótipo, esta operação é utilizada quando ocorre a classificação correta do registro de entrada;
- A Figura 3.17(d) mostra o efeito da subtração do vetor resultante da etapa anterior pelo vetor protótipo, nota-se que ocorreu um afastamento entre o registro de entrada e o vetor protótipo, esta operação é utilizada quando ocorre a classificação errada do registro de entrada;

Um fator importante a ser observado no processo de aprendizagem é que a taxa de aprendizado está diretamente ligada ao deslocamento do vetor protótipo, por isto é desejável que a taxa de aprendizagem decresça monotonicamente com o número de iterações [32], isto permite que no início do treinamento os movimentos dos vetores protótipos sejam maiores, maximizando a exploração do espaço de entrada dos dados e, ao final, os movimentos realizados pelos vetores protótipos sejam menores, caracterizando um ajuste mais fino do seu

posicionamento.

Como critério de parada para o algoritmo LVQ pode se utilizar o número máximo de iterações, um valor mínimo para a taxa de aprendizagem ou o erro (distância) calculado entre os vetores, entre outros.

Capítulo 4

Implementação das RNAs para a Ferramenta YADMT

As Redes MLP, RBF e LVQ foram implementadas para uma ferramenta chamada YADMT - *Yet Another Data Mining Tool*, cujo desenvolvimento foi paralelo a este trabalho⁴. É uma ferramenta livre, que pretende implementar todas as etapas de um processo de KDD, a longo prazo. Cada técnica das etapas de KDD que é implementada representa um módulo no sistema, sendo que a mesma já tem desenvolvida os módulos de aquisição de dados do SGBD PostgreSQL e de arquivos no formato ARFF, que é um modelo de arquivos de dados utilizado pela ferramenta Weka. Além disto, possui também os métodos de avaliação *Holdout* e *k-fold-cross-validation*.

Os módulos implementados para a ferramenta utilizam o conceito de *plugin*, ou seja, está adicionando uma função específica a um programa maior, no caso deste trabalho, está-se adicionando a funcionalidade de Classificação utilizando algoritmos baseados em RNAs. A vantagem de utilizar este conceito para a evolução da ferramenta é que os módulos desenvolvidos geralmente são pequenos, leves e possibilitam a separação clara do código fonte dos diferentes componentes do programa, sendo que quando ocorrer algum problema envolvendo seu uso, basta desinstalá-lo do programa principal para que seja solucionado.

Para o perfeito funcionamento e comunicação entre a ferramenta e os módulos deve haver uma forma de registrar estes módulos no programa principal e um protocolo ou regras de desenvolvimento que possibilitem a correta troca de informações entre ambos. Para isto, foram utilizadas algumas tecnologias da linguagem de programação Java, a qual foi utilizada para o desenvolvimento da ferramenta.

A forma de registro de um módulo ocorre com a inserção do seu arquivo com a extensão

⁴ Outros detalhes sobre desenvolvimento e tecnologias utilizadas para a construção da ferramenta podem ser encontradas em [9].

JAR, o qual é gerado após a compilação do módulo pelo compilador Java, em uma pasta específica determinada pela arquitetura do programa, este arquivo contém classes que são então carregadas dinamicamente para a ferramenta. As regras de comunicação entre os módulos e a programa principal são estabelecidas através da definição de *Interfaces* que os módulos devem implementar e a inserção de *Annotation* (recurso de metadados da linguagem Java) em algumas classes.

Nas subseções seguintes detalha-se os procedimentos para a implementação de um módulo de Classificação para a ferramenta e algumas características dos módulos implementados.

4.1 Características do Módulo de Classificação

A primeira coisa a se fazer para desenvolver os módulos de Classificação para a ferramenta YADMT é a importação da biblioteca *InterfacesYADMT*, que possui as estruturas básicas de comunicação entre os módulos e a ferramenta, e a definição de *Interfaces* e *Annotations* necessárias à implementação dos módulos. Para o módulo de Classificação três componentes são essenciais, a *Interface ClassifierModuleInterface* e as *Annotations ClassifierModuleAnnotation* e *ModuleAnnotation*.

A Figura 4.1 mostra as assinaturas dos métodos da *Interface ClassifierModuleInterface* que os módulos devem implementar. Os dois métodos principais desta *Interface* são os métodos *train()* e *test()* que executam a principais funcionalidades de um módulo de Classificação. O método *train()* é responsável por executar o treinamento do algoritmo, para isto ele recebe dois parâmetros, *Object input[][]* e *Object output[]*, o primeiro é uma matriz que contém os valores de cada registro de dados de entrada do módulo e o segundo é um vetor de valores que corresponde à saída desejada correspondente a cada registro da entrada. O método *test()* recebe como parâmetro um vetor de valores, que representa um registro que vai ser classificado pelo módulo, e retorna a classificação do respectivo registro.

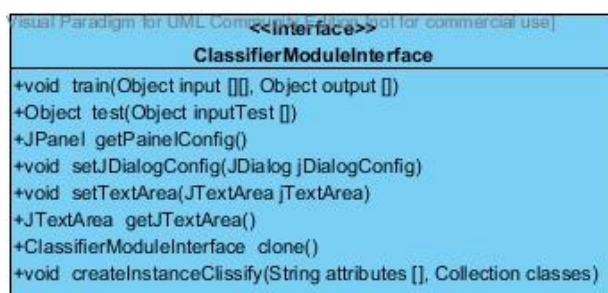


Figura 4.1: Diagrama de Classes da Interface ClassifierModuleInterface.

Os demais métodos da *Interface* que um módulo de Classificação tem que implementar ocupam funções secundárias no funcionamento entre o módulo e a ferramenta. São eles:

- *getPainelConfig()*: retorna um painel que tem as opções de configuração de criação e treinamento de um algoritmo de Classificação, que para RNAs são valores representando o número de neurônios, a taxa de aprendizagem, erro mínimo, taxa *momentum*, entre outros. Os detalhes de cada uma serão apresentados na subseção seguinte.
- *setJDialogConfig()*: recebe como parâmetro o *container* que é obtido do retorno de *getPainelConfig()*.
- *setTextArea()*: recebe um objeto do tipo *JTextArea* que serve para o módulo de Classificação imprimir alguns dados durante a sua execução.
- *createInstanceClassify()*: responsável pela instanciação do algoritmo de classificação, para isto recebe como parâmetros o nome dos atributos de entrada e uma *Collection* das classes que a base de dados possui.

Nos Algoritmos 4.1 e 4.2 tem-se a declaração das duas *Annotations* que um módulo de Classificação deve utilizar. A *Annotation ClassifierModuleAnnotation*, serve para identificar que o módulo implementado executa a tarefa de Classificação de dados. Técnicas de Agrupamento de dados, Predição, entre outras, teriam *Annotations* diferentes e específicas para cada uma. A *Annotation ModuleAnnotation* possibilita ao módulo colocar seu nome como parâmetro, o qual serve para a programa principal exibir aos usuários o nome do módulo implementado.

Algoritmo 4.1: *Annotation ClassifierModuleAnnotation*.

```
public @interface ClassifierModuleAnnotation{  
}
```

Algoritmo 4.2: *Annotation ModuleAnnotation*.

```
public @interface ModuleAnnotation{  
    String name();  
}
```

A utilização das *Annotations* e da *Interface* é mostrada no trecho de código Algoritmo 4.3.

A Figura 4.2 mostra a ferramenta YADMT com a aba da tarefa de Classificação ativa, nesta figura é possível verificar que foram carregados dinamicamente alguns módulos de Classificação, inclusive o módulo MLP implementado para este trabalho. Também é possível visualizar outros elementos, como o botão "Configurações", que permite a parametrização de criação e treinamento do módulo classificador; o botão "Treinar", que inicia o treinamento do módulo; o botão "Classificar", que faz a classificação de novos dados; o botão "Visualizar Modelo" que apresenta um modelo visual do módulo (quando o mesmo possuir); as opções de particionamento dos dados; os métodos de avaliação disponíveis e uma lista de classificadores já treinados.

Algoritmo 4.3. Utilização da *Interface ClassifierModuleInterface* e das *Annotations ClassifierModuleAnnotation* e *ModuleAnnotation*.

```

@ModuleAnnotation(name = "Multilayer Perceptron - MLP")
@ClassifierModuleAnnotation
public class ModuloMLP implements Serializable, ClassifierModuleInterface {
    (...)
}

```

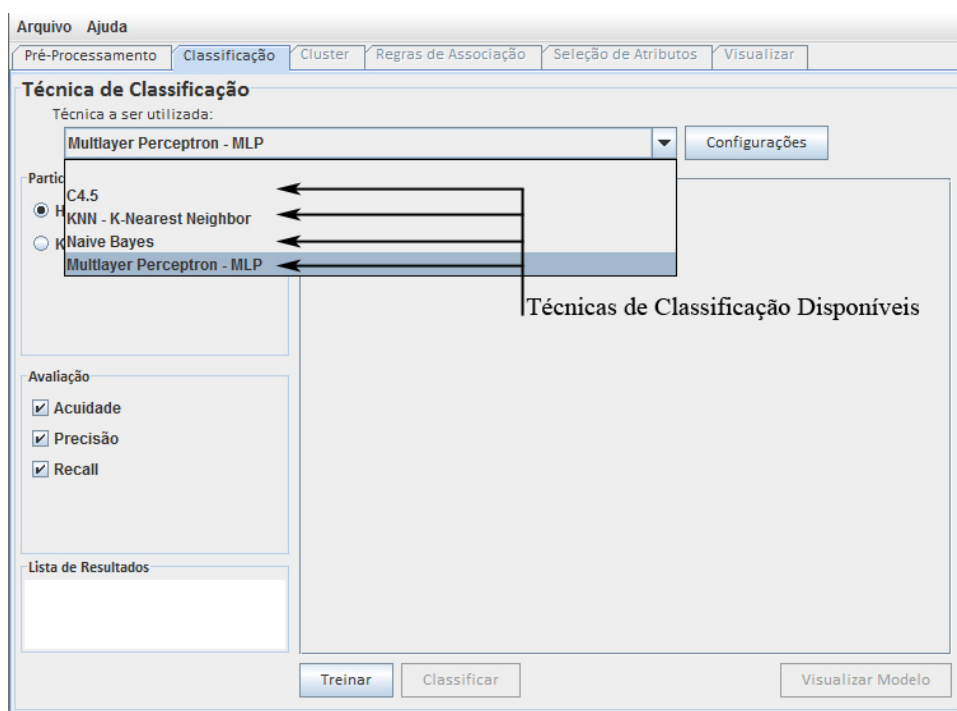


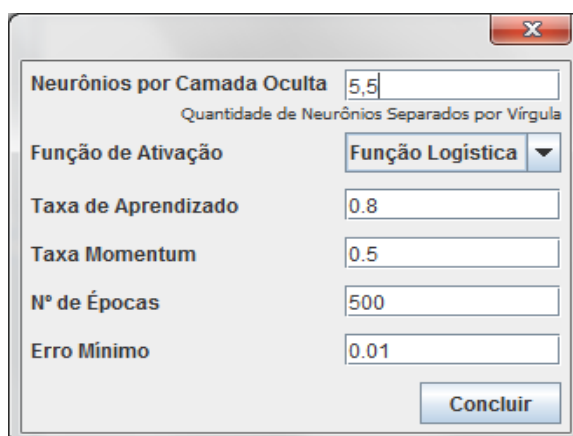
Figura 4.2: Imagem da ferramenta YADMT com os módulos/técnicas de Classificação disponíveis.

4.2 Implementações dos RNAs

Como verificado nas seções anteriores, todas as RNAs tem suas particularidades, ou seja, as suas arquiteturas e seus modos de treinamento. Logo, não possuem os mesmos atributos de configuração e geração. Desta forma, a implementação dos módulos teve que permitir a parametrização de cada RNA pelo usuário final. Os únicos parâmetros que são definidos automaticamente pelos módulos são: o número de neurônios da camada de entrada e o número e neurônios da camada de saída, os quais são determinados de acordo com a base de dados a ser utilizada.

Os parâmetros disponíveis para os usuários são de dois tipos, de criação da arquitetura da RNA e de configurações para o seu treinamento.

O módulo da Rede MLP implementada permite que o usuário defina o número de camada ocultas e o número de neurônios em cada camada. Para definir estes parâmetros o usuário deve digitar uma seqüência de números separados por vírgula, onde cada número digitado adiciona uma nova camada oculta na rede e o número corresponde à quantidade de neurônios naquela camada. Além disto, o módulo implementado também permite que o usuário defina o tipo de função de ativação utilizada nos neurônios da RNA, que para os testes deste trabalho foi implementado apenas função logística. O algoritmo de treinamento implementado para esta RNA foi o algoritmo *backpropagation* e os parâmetros que o usuário pode determinar são: taxa de aprendizagem, taxa *momentum*, número máximo de épocas e erro mínimo. Os dois últimos parâmetros definem o critério de parada do algoritmo de treinamento. A Figura 4.3 mostra a janela com os parâmetros de configuração da Rede MLP que o usuário deve que preencher.



Neurônios por Camada Oculta	5,5
Quantidade de Neurônios Separados por Vírgula	
Função de Ativação	Função Logística
Taxa de Aprendizado	0.8
Taxa Momentum	0.5
Nº de Épocas	500
Erro Mínimo	0.01
Concluir	

Figura 4.3: Parâmetros de configuração da Rede MLP.

O módulo da Rede RBF implementa muitos parâmetros parecidos com a Rede MLP. Como parâmetros de criação da arquitetura da Rede RBF o usuário deve definir: número de neurônios da única camada oculta e qual tipo de função de ativação estes neurônios devem ter, que para os testes a serem executados foi implementando a função Guassiana. Para o treinamento da camada oculta a Rede RBF implementou o algoritmo *k-means*, onde a quantidade de grupos *k* é igual ao número de neurônios da camada oculta e o usuário não tem nenhum parâmetro a definir. Para determinar a largura da função de ativação implementada foi utilizada a heurística do tamanho da metade da distância do neurônio mais próximo, porém, foi deixado apto para que sejam implementadas nova formas, semelhantes às descritas na Subseção 3.3.1. Para o treinamento da camada de saída foi implementado a regra Delta, onde o usuário deve determinar a taxa de aprendizado, número máximo de épocas e o erro mínimo. Os últimos dois parâmetros determinam quando a Rede RBF deve finalizar o treinamento. A Figura 4.4 mostra a janela com os itens de configuração da Rede RBF a serem definidos pelo usuário.

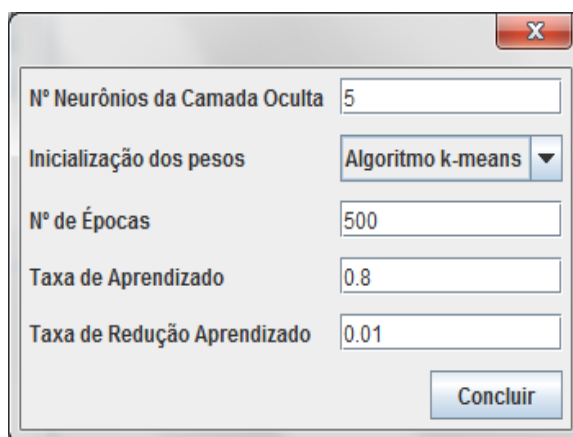
N° Neurônios da Camada Oculta	5
Função de Ativação	Função Guassiana
Largura Função de Ativação	Metade Menor Distância
Taxa de Aprendizado	0.8
N° de Épocas	500
Erro Mínimo	0.01

Concluir

Figura 4.4: Parâmetros de configuração da Rede RBF.

Para a implementação do módulo da Rede LVQ o único parâmetro a ser definido para a criação da arquitetura da rede é o número de neurônios a ser utilizado na camada oculta. Os demais parâmetros se referem ao treinamento, onde deve ser determinada a forma de inicialização dos pesos dos neurônios da camada oculta, o número máximo de épocas, a taxa de aprendizado e a taxa de decaimento do aprendizado a ser utilizado no algoritmo LVQ. Para a inicialização da camada oculta foi implementado o algoritmo *k-means*, o qual não tem nenhum parâmetro a ser definido pelo usuário e a quantidade de grupos *k* é igual ao número

de neurônios da camada oculta. A Figura 4.5 mostra a janela com os itens de configuração da Rede LVQ que o usuário deve que preencher.



N° Neurônios da Camada Oculta	5
Inicialização dos pesos	Algoritmo k-means ▼
N° de Épocas	500
Taxa de Aprendizado	0.8
Taxa de Redução Aprendizado	0.01

Concluir

Figura 4.5: Parâmetros de configuração da Rede LVQ.

Após a implementação, os módulos foram compilados e acoplados na ferramenta YADMT, conforme descrito no início deste Capítulo, passando-se então, à fase de testes de validação.

Capítulo 5

Testes Realizados

Para avaliar o desempenho das três RNAs foram feitas medições de acuidade e tempo de treinamento em dez bases de dados da UCI *Machine Learning Repository* [54], um repositório de bases de dados de acesso público utilizado pela comunidade de aprendizado de máquina para análise empírica de algoritmos de aprendizagem de máquina. As bases de dados escolhidas para testes foram as apresentadas na Tabela 5.1⁵.

Tabela 5.1: Bases de dados escolhidas para comparação das RNAs implementadas.

Nome da Base de Dados	Tamanho	Atributos	Classes	Atributos Categóricos
<i>Abalone</i>	4177	8	28	1
<i>Acute Inflammations</i>	120	6	2	5
<i>Ecoli</i>	336	7	8	0
<i>Iris</i>	150	4	3	0
<i>Libras Movement</i>	360	90	15	0
<i>Spambase</i>	4601	57	2	0
<i>Statlog (Image Segmentation)</i>	2310	19	6	0
<i>Statlog (Landsat Satellite)</i>	6435	36	7	0
<i>Teaching Assistant Evaluation</i>	151	5	3	3
<i>Wine Quality</i>	4898	11	7	0

Este conjunto de bases de dados possibilitou analisar o comportamento dos algoritmos em bases de dados pequenas, médias e grandes, com número de atributos pequenos, médios e grandes e com pequenas, médias e grandes quantidades de classes, além de bases com dados categóricos e bases somente com atributos numéricos. Neste trabalho a classificação adotada para as bases de dados é a proposta por Zheng [86], da seguinte forma:

- Quanto ao número de registros:

⁵ A descrição das bases de dados utilizadas consta no Apêndice A

- Pequena: menos de 210 registros (*Iris*, *Teaching Assistant Evaluation*, *Acute Inflammations*);
- Média: entre 210 e 3170 registros (*Ecoli*, *Statlog (Image Segmentation)*, *Libras Movement*);
- Grande: mais de 3170 registros (*Abalone*, *Wine Quality*, *Statlog (Landsat Satellite)*, *Spambase*);
- Quanto ao número de atributos:
 - Pequena: menos de 10 atributos (*Abalone*, *Ecoli*, *Iris*, *Teaching Assistant Evaluation*, *Acute Inflammations*);
 - Média: entre 10 e 30 atributos (*Wine Quality*, *Statlog (Image Segmentation)*);
 - Grande: mais 30 atributos (*Statlog (Landsat Satellite)*, *Spambase*, *Libras Movement*);
- Quanto ao número de classes:
 - Pequena (binário): duas classes (*Acute Inflammations*, *Spambase*);
 - Média: entre 3 e 10 classes (*Ecoli*, *Wine Quality*, *Iris*, *Teaching Assistant Evaluation*, *Statlog (Image Segmentation)*, *Statlog (Landsat Satellite)*);
 - Grande: mais de 10 classes (*Abalone*, *Libras Movement*).

Ainda, de acordo com Zheng [86], um classificador tem alta taxa de acuidade quando a porcentagem da taxa de acertos for maior que 75%, média quando estiver entre 40 e 75% e baixa quando for menor que 40%.

Uma RNA não trabalha com atributos categóricos, apenas números, por isto as bases de dados *Abalone*, *Acute* e *Teaching Assistant Evaluation* tiveram seus valores mapeados para valores binários, conforme proposto por Barreto [6]. Desta forma o atributo “*Sex*” da base de dados *Abalone*, teve os valores *F* e *M*, alterados para *0* e *1* respectivamente, bem como os atributos “*Occurrence of nausea*”, “*Lumbar pain*”, “*Urine pushing*”, “*Micturition pains*”, “*Burning of urethra, itch, swelling of urethra outlet*” da base de dados *Acute* tiveram os valores “*no*” e “*yes*” alterados para *0* e *1*.

A base de dados *Teaching Assitant Evaluation* apresenta dois atributos com dois valores categóricos distintos, o primeiro entre “*English speaker*” e “*Non-Speaker*”, o segundo entre “*Summer*” e “*Regular Semester*”, ambos mapeados para *0* e *1* respectivamente, além disso o atributo “*Course instructor*” teve seus valores mapeados de acordo coma a Tabela 5.2 e o atributo “*Course*” de forma equivalente. Os demais valores de todas as bases de dados foram

normalizados entre 0 e 1, pelo fato de utilizar a função logística na Rede MLP, mas também poderiam ser normalizados entre -1 e 1 caso utilizassem a função hiperbólica como função de ativação das RNAs. Este procedimento é sugerido para aumentar a acuidade do classificador.

Tabela 5.2: Mapeamento atributo *Course instructor* da base de dados *Teaching Assistant Evaluation*.

Atributo Course (ID)	Mapeamento Correspondente
1	0000000000000000000001
2	00000000000000000000010
3	000000000000000000000100
4	0000000000000000000001000
5	00000000000000000000010000
6	000000000000000000000100000
....	...
25	1000000000000000000000000000

As três RNAs foram treinadas utilizando o método *k-fold-cross-validation* (com *k* no valor de 10) por uma quantidade de 2000 épocas ou até atingir a soma dos erros quadráticos de cada época em valor inferior a 0.01. Durante os testes com estas bases de dados, as mesmas tiveram medidos o tempo de treinamento e o valor da acuidade, este foi obtido com base nas matrizes de confusões geradas, omitidas aqui por questões de espaço. A análise dos resultados, bem como as particularidades de teste de cada RNA podem ser verificadas nas subseções seguintes.

5.1 Testes com Redes MLP

Devido às diferenças nas bases de dados quanto ao número de entradas e à quantidade de rótulos de classes, não é possível analisar uma mesma arquitetura para todos os casos. Desta forma, para cada base de dados foram testadas três arquiteturas diferentes de Rede MLP, onde o número de neurônios da camada de entrada é sempre igual ao número de atributos de entrada e o número de neurônios da camada de saída é igual ao número de rótulos de classe da base de dados, o que muda, portanto, nas três arquiteturas é o número de camadas ocultas e o número de neurônios em cada uma destas camadas.

A primeira arquitetura testada foi uma MLP com uma camada oculta com número de neurônios igual a $2n+1$, onde n é o número de atributos de entrada. Esta quantidade de neurônios é sugerida pelo Teorema de Kolmogorov-Nielsen, apresentado por Kovacz [45],

onde o mesmo sugere que para uma rede de três camadas é possível representar qualquer função utilizando esta quantidade de neurônios na camada oculta.

A segunda arquitetura apresenta uma camada oculta com quantidade de neurônios igual ao número de atributos de entrada e foi escolhida a fim de verificar qual o desempenho da RNA quando o número de neurônios na camada oculta é menor que a metade da quantidade sugerida pelo Teorema de Kolmogorov-Nielsen.

A terceira arquitetura testada apresenta duas camadas ocultas com a primeira camada oculta com um número de neurônios igual a n , e a segunda camada oculta de tamanho $n+1$, escolhida a fim de verificar o comportamento da RNA quando tem um número de neurônios igual ao proposto pelo Teorema de Kolmogorov-Nielsen, porém, distribuído em duas camadas ocultas.

O resultado das taxas de acuidade das Redes MLP é apresentado na Tabela 5.3, onde a coluna “Média Acuidade” representa a média aritmética das seis taxas de acuidade (três dos treinamentos e três dos testes) da arquitetura de rede. Nesta tabela pode-se verificar que foram obtidas médias de acuidade baixas, médias e altas, sendo que metade das bases de dados tiveram uma média alta (maior que 75%) utilizando as arquiteturas propostas, com destaque para a base de dados *Acute Inflammations* que acertou todas as classificações.

Embora a maioria das bases de dados tenham tido a taxa de acuidade para treinamento próximo a taxa de acuidade de testes, houve exceções nas bases de dados *Ecoli* e *Libras Movement*. Ao verificar a distribuição dos dados destas bases de dados, verificou-se que as mesmas apresentavam os registros ordenados por classes, ou seja, todos os registros que pertenciam a certa classe estavam em seqüência. Desta forma, como foi utilizado o método *k-fold-cross-validation* com k igual a 10, cada grupo de testes teve tamanho 33 e 36 para as bases de dados *Ecoli* e *Libras Movement*, respectivamente, e o fato destes tamanhos serem maiores que a quantidade de registros de certas classes (vide Apêndice A), fez com que nenhum registro destas classes participasse do treinamento em algumas iterações, sendo possivelmente essa a causa da grande diferença de acuidade. Embora o método *k-fold-cross-validation* seja indicado para evitar *overfitting*, ou seja, que a RNA seja eficiente não apenas para classificar os dados de treinamento, mas também dados não vistos nesta etapa, não foi o que ocorreu nessas bases de dados.

Tabela 5.3: Acuidade das Redes MLP nas 10 bases de dados escolhidas.

Base de Dados	Número de Neurônios por Camadas				Acuidade	Acuidade	Média Acuidade
	Entrada	1ª Oculta	2ª Oculta	Saída	Treinamento (%)	Teste (%)	
<i>Abalone</i>	8	8	-	28	25.64	24.34	25.51
	8	17	-	28	27.29	24.89	
	8	8	9	28	26.23	24.70	
<i>Libras Movement</i>	90	90	-	15	42.15	12.50	26.27
	90	181	-	15	35.97	14.17	
	90	90	91	15	39.79	13.06	
<i>Wine Quality</i>	11	11	-	7	55.38	48.19	54.58
	11	23	-	7	58.56	52.24	
	11	11	12	7	59.56	53.57	
<i>Teaching Assistant Evaluation</i>	5	5	-	3	69.27	64.67	63.94
	5	11	-	3	79.93	72.00	
	5	5	6	3	52.43	45.33	
<i>Statlog (Landsat Satellite)</i>	36	36	-	6	87.73	86.69	65.83
	36	73	-	6	77.23	75.63	
	36	36	37	6	39.20	28.48	
<i>Ecoli</i>	7	7	-	8	93.53	64.78	79.19
	7	15	-	8	92.57	65.97	
	7	7	8	8	91.15	67.16	
<i>Iris</i>	4	4	-	3	86.67	83.33	84.92
	4	9	-	3	86.33	83.33	
	4	4	5	3	86.50	83.33	
<i>Spambase</i>	57	57	-	2	97.14	94.07	94.17
	57	115	-	2	96.87	93.76	
	57	57	58	2	96.33	86.85	
<i>Statlog (Image Segmentation)</i>	19	19	-	7	98.25	97.53	97.72
	19	39	-	7	97.84	97.40	
	19	19	20	7	97.86	97.45	
<i>Acute Inflammations</i>	6	6	-	2	100.00	100.00	100.00
	6	13	-	2	100.00	100.00	
	6	6	7	2	100.00	100.00	

Pode-se verificar que o desempenho das MLPs não dependeu da classificação das bases de dados quanto ao número de registros ou ao número de atributos, pois de acordo com a Tabela 5.3 houve alta taxa de acuidade tanto para bases de dados grandes (*Spambase*) e pequenas (*Acute* e *Iris*) em número de registros, quanto para bases de dados pequenas (*Iris*, *Acute*

Inflammations) e grandes (*Spambase*) em número de atributos. Onde pôde ser verificada uma relação mais direta foi quanto ao número de classes, pois as bases de dados pequenas (*Acute Inflammations*, *Spambase*) tiveram altas taxas de acuidade e as duas maiores (*Libras Movement* e *Abalone*) tiveram os piores desempenhos.

Tabela 5.4: Tempos de treinamentos das Redes MLP.

Base de Dados	Número de Neurônios por Camada				Conexões * Tamanho Base	Tempo de Treinamento (ms)
	Entrada	1ª Oculta	2ª Oculta	Saída		
<i>Iris</i>	4	4	-	3	4200	16640
<i>Acute Inflammations</i>	6	6	-	2	5760	18253
<i>Teaching Assistant Evaluation</i>	5	5	-	3	6040	21840
<i>Iris</i>	4	4	5	3	7650	30267
<i>Iris</i>	4	9	-	3	9450	32397
<i>Teaching Assistant Evaluation</i>	5	5	6	3	11023	39780
<i>Acute Inflammations</i>	6	6	7	2	11040	36870
<i>Acute Inflammations</i>	6	13	-	2	12480	37230
<i>Teaching Assistant Evaluation</i>	5	11	-	3	13288	42223
<i>Ecoli</i>	7	7	-	8	35280	111903
<i>Ecoli</i>	7	7	8	8	56784	183717
<i>Ecoli</i>	7	15	-	8	75600	219130
<i>Wine Quality</i>	11	11	-	7	969804	2619927
<i>Statlog (Image Segmentation)</i>	19	19	-	7	1039500	2474687
<i>Abalone</i>	8	8	-	28	1202976	3850717
<i>Abalone</i>	8	8	9	28	1620676	5095810
<i>Wine Quality</i>	11	11	12	7	1650626	4712517
<i>Statlog (Image Segmentation)</i>	19	19	20	7	1845690	4859210
<i>Wine Quality</i>	11	23	-	7	2027772	5194560
<i>Statlog (Image Segmentation)</i>	19	39	-	7	2136750	4985883
<i>Abalone</i>	8	17	-	28	2556324	7167030
<i>Libras Movement</i>	90	90	-	15	3402000	6780253
<i>Libras Movement</i>	90	90	91	15	6355800	14339673
<i>Libras Movement</i>	90	181	-	15	6841800	13506217
<i>Statlog (Landsat Satellite)</i>	36	36	-	6	9729720	20090897
<i>Spambase</i>	57	57	-	2	15473163	29880087
<i>Statlog (Landsat Satellite)</i>	36	36	37	6	18339750	42940037
<i>Statlog (Landsat Satellite)</i>	36	73	-	6	19729710	40768197
<i>Spambase</i>	57	57	58	2	30693271	57059760
<i>Spambase</i>	57	115	-	2	31217785	61041770

Na Tabela 5.4, observa-se o tempo de treinamento de cada RNA em cada base de dados e

pode-se verificar que também não houve uma relação direta quanto ao tempo de treinamento e a taxa de acuidade, haja vista que tanto *Spambase* e *Acute Inflammation* que tiveram tempos de treinamento bem diferentes tiveram valor alto de acuidade.

Nota-se aqui um inconveniente do algoritmo *backpropagation* para a Rede MLP já apontado por outros autores [12], que é o tempo de treinamento. O maior tempo registrado foi para a base de dados *Spambase* com um valor de 61041770 ms, ou seja, mais de 16 horas e 50 minutos, neste caso embora a RNA tenha demorado a treinar, teve uma taxa de acuidade de 93,76% nos testes, diferente da base de dados *Libras Movement*, que teve uma taxa de acuidade baixa, porém com treinamento consideravelmente alto (13506217 ms, ou 3 horas 45 minutos e 6.217 segundos) em relação a bases de dados que tiveram pouco tempo de treinamento e alta taxa de acuidade.

Embora o Teorema de Kolmogorov-Nielsen indique uma quantidade de $2n+1$ neurônios na camada oculta, na maioria das vezes obteve-se valores muitos próximos de acuidade para as demais arquiteturas em uma mesma base de dados, em alguns casos arquiteturas com menores quantidades de neurônios obtiveram maiores taxas de acuidade, a exemplo das bases de dados *Statlog (Image Segmentation)* e *Statlog (Landsat Satellite)*. Distribuir os $2n+1$ neurônios em mais camadas (2 nos testes realizados) também não garantiu redução ou aumento na taxa de acuidade, observou-se um menor tempo de treinamento já que o número de conexões da rede diminuiu.

5.2 Testes com Redes RBF

Assim como na Rede MLP, com Redes RBF também não é possível analisar a mesma arquitetura para todas as bases de dados escolhidas. De forma similar, foram testadas três arquiteturas de Redes RBF para cada base de dados, onde a camada de entrada tinha quantidade de neurônios igual ao número de atributos de entrada e a camada de saída tinha a mesma quantidade de neurônios que a quantidade de classes da base de dados, novamente o que mudou nas arquiteturas é a quantidade de neurônios na camada oculta.

A primeira arquitetura tinha número de neurônios na camada oculta igual a $2n+1$, onde n é o número de atributos de entrada da base de dados. Esta abordagem foi utilizada novamente pelo fato do Teorema de Kolmogorov-Nielsen se referir às RNAs de três camadas, não especificamente à Rede MLP. Na segunda arquitetura número de neurônios ocultos era igual ao número de atributos de entrada da base de dados e a terceira arquitetura tinha quantidade

igual ao número de neurônios da camada de saída. O resultado da acuidade dos testes das Redes RBF é apresentado na Tabela 5.5.

Tabela 5.5: Acuidade das Redes RBF nas 10 bases de dados escolhidas.

Base de Dados	Número de Neurônios por Camada			Acuidade Treinamento (%)	Acuidade Teste (%)	Média Acuidade
	Entrada	Oculto	Saída			
<i>Abalone</i>	8	8	28	21.21	20.98	22.32
	8	17	28	23.36	22.97	
	8	28	28	23.02	22.35	
<i>Teaching Assistant Evaluation</i>	5	5	3	40.29	29.33	40.37
	5	11	3	44.49	35.33	
	5	3	3	48.09	44.67	
<i>Ecoli</i>	7	7	8	36.67	30.61	43.92
	7	15	8	61.85	54.55	
	7	8	8	42.61	37.27	
<i>Libras Movement</i>	90	90	15	80.80	15.83	45.74
	90	181	15	94.60	25.00	
	90	15	15	47.65	10.56	
<i>Wine Quality</i>	11	11	7	49.02	48.86	48.93
	11	23	7	48.93	48.28	
	11	7	7	49.38	49.10	
<i>Spambase</i>	57	57	2	65.74	63.50	71.94
	57	115	2	76.40	71.96	
	57	2	2	77.46	76.61	
<i>Statlog (Image Segmentation)</i>	19	19	7	83.50	82.86	77.14
	19	39	7	89.29	89.13	
	19	7	7	59.04	59.05	
<i>Statlog (Landsat Satellite)</i>	36	36	6	83.30	82.50	77.38
	36	73	6	86.38	85.43	
	36	6	6	63.98	62.69	
<i>Iris</i>	4	4	3	77.78	78.00	79.69
	4	9	3	94.37	94.67	
	4	3	3	66.67	66.67	
<i>Acute Inflammations</i>	6	6	2	91.94	88.33	93.94
	6	13	2	100.00	100.00	
	6	2	2	91.67	91.67	

A Tabela 5.5 mostra que as médias de acuidade das Redes RBF foram parecidas com as médias da Rede MLP, apresentando taxas de acuidade baixas, médias e altas. A maioria das bases de dados teve a média das taxas de acuidade reduzidas, com exceção das bases de dados

Libras Movement e *Statlog (Landsat Satellite)*. Destaque-se a redução das taxas de acuidade para as bases de dados *Ecoli*, *Spambase* e *Statlog (Image Segmentation)*, sendo que a base de dados *Statlog (Image Segmentation)* reduziu aproximadamente 20% a média das acuidades.

Com os testes das Redes RBF foi possível verificar que a aplicação do método *k-fold-cross-validation* não foi efetiva novamente para evitar o efeito de *overfitting* nas bases de dados *Libras Movement* e *Ecoli*. Houve grande diferença entre as taxas de acuidade dos treinamentos e as taxas de acuidade dos testes, embora tenha sido mais suave para a base de dados *Ecoli*, que teve uma diferença mínima de 5,33% na estrutura RBF(7,8,8)⁶ e mínimo de 23,98% na estrutura MLP(7,7,8,8), foi mais acentuada para a base de dados *Libras Movement*, que teve diferença de 69,59% na estrutura RBF(90, 181, 15).

Da mesma forma que o desempenho das Redes MLP não dependeu do número de registros e do número de atributos das bases de dados, com Redes RBF também não houve esta dependência, nem com a quantidade de classes, onde havia ocorrido uma relação mais direta anteriormente. Foi possível verificar que bases de dados com pequena quantidade de registros tiveram tanto altas taxas de acuidade (*Iris*, *Acute Inflammations*) quanto taxas mais baixas (*Teaching Assistant Evaluation*). Bases de dados com grandes quantidades de registros também tiveram tanto altas taxas de acuidade (*Landsat Satellite*) quanto taxas mais baixas (*Abalone*, *Wine* e *Spambase*). Resultados similares ocorreram para as bases de dados quanto à quantidade de atributos de entrada e ao número de classes.

Na Tabela 5.6, observa-se o tempo de treinamento de cada arquitetura em cada base de dados. Neste tipo de RNA também não houve relação direta do tempo de treinamento com a acuidade, o fato de uma base de dados ter tido alta taxa de acuidade não significa que a mesma tenha sido treinada por um curto ou longo período de tempo, o mesmo ocorre para taxas baixas de acuidade. Nos tempos de treinamento apresentados nota-se que os quatro maiores tempo de treinamento tiveram as taxas de acuidade com média entre 71% e 78% e que a maior média de taxa de acuidade está entre os três menores tempos de treinamento.

Uma diferença significativamente importante nos testes realizados com as Redes MLP e RBF está relacionada ao tempo de treinamento das mesmas. Comparando apenas as arquiteturas que tinham a igual quantidade de camadas e número de neurônios, os testes realizados mostram que as Redes RBF demoraram em média 68,92% menos tempo do que as Redes MLP para finalizar o treinamento de uma mesma base de dados, esta diferença chegou

⁶ Em citações do tipo SIGLA($t_1, t_2, t_3, \dots, t_n$), SIGLA se refere à sigla do tipo de Rede Neural Artificial e t_i se refere ao número de neurônios de cada camada i .

a 84,40% para a base de dados *Spambase*, desigualdade de 51525315 ms, ou 14 horas, 18 minutos e 45 segundos, tempo consideravelmente alto em termos computacionais onde a maioria das operações que um usuário realiza é executada em poucos segundos.

Tabela 5.6: Tempos de treinamentos das Redes RBF.

Base de Dados	Número de Nerônios por Camada			Conexões * Tamanho Base	Tempo de Treinamento
	Entrada	Oculto	Saída		
<i>Acute Inflammations</i>	6	2	2	1920	3198
<i>Iris</i>	4	3	3	3150	5751
<i>Acute Inflammations</i>	6	13	2	12480	5866
<i>Teaching Assistant Evaluation</i>	5	3	3	3624	6080
<i>Iris</i>	4	4	3	4200	7511
<i>Acute Inflammations</i>	6	6	2	5760	8626
<i>Teaching Assistant Evaluation</i>	5	5	3	6040	9598
<i>Iris</i>	4	9	3	9450	14819
<i>Teaching Assistant Evaluation</i>	5	11	3	13288	18805
<i>Ecoli</i>	7	7	8	35280	41140
<i>Ecoli</i>	7	8	8	40320	47683
<i>Ecoli</i>	7	15	8	75600	83472
<i>Libras Movement</i>	90	15	15	567000	185703
<i>Spambase</i>	57	2	2	542918	191007
<i>Statlog (Image Segmentation)</i>	19	7	7	404250	288379
<i>Wine Quality</i>	11	7	7	617148	574270
<i>Statlog (Landsat Satellite)</i>	36	6	6	1621620	737306
<i>Statlog (Image Segmentation)</i>	19	19	7	1039500	744761
<i>Wine Quality</i>	11	11	7	969804	975497
<i>Libras Movement</i>	90	90	15	3402000	1120037
<i>Abalone</i>	8	8	28	1202976	1247978
<i>Statlog (Image Segmentation)</i>	19	39	7	2136750	1511585
<i>Wine Quality</i>	11	23	7	2027772	1823501
<i>Abalone</i>	8	17	28	2556324	2255843
<i>Libras Movement</i>	90	181	15	6841800	2321928
<i>Abalone</i>	8	28	28	4210416	3990427
<i>Statlog (Landsat Satellite)</i>	36	36	6	9729720	4353224
<i>Spambase</i>	57	57	2	15473163	4734024
<i>Statlog (Landsat Satellite)</i>	36	73	6	19729710	8453701
<i>Spambase</i>	57	115	2	31217785	9516455

5.3 Testes com Redes LVQ

A abordagem para realizar os testes com Redes LVQ não foi diferente das Redes MLP e

RBF, pois também não é possível analisar a mesma arquitetura para todas as bases de dados. De forma equivalente aos testes anteriores, as arquiteturas das Redes LVQ tinham a camada de entrada com tamanho igual ao número de atributos da base de dados e camada de saída com tamanho igual ao número de classes. Foram testadas três RNAs para cada base de dados, onde os números de neurônios da camada oculta foram igual a m , $2m$ e $3m$ para a primeira, segunda e terceira arquitetura, respectivamente, onde m é o número de classes da base de dados. Esta quantidade de neurônios foi escolhida pelo fato da Rede LVQ escolher um neurônio vencedor de sua estrutura, tal que cada classe da base de dados tinha a mesma quantidade de neurônios na estrutura da RNA.

O resultado da acuidade das Redes LVQ é apresentado na Tabela 5.7, a qual mostra que as Redes LVQ apresentaram pequenos, médios e altos valores de acuidade, assim como as Redes MLP e RBF. Porém esta foi a RNA que apresentou as menores taxas de acuidade entre as três estudadas neste trabalho, com as três piores médias de acuidade de todos os testes realizados (as bases de dados *Wine Quality*, *Abalone* e *Libras Movement*, tiveram valores abaixo de 16%), sendo que as bases de dados *Abalone* e *Wine Quality* tiveram valores próximos a 0.

Entre os resultados apresentados, as bases de dados *Acute Inflammations* e *Iris* tiveram diferenças relevantes em relação aos testes anteriores. Embora a base de dados *Acute Inflammations* tenha tido altas taxas de acuidade para todas as RNA, foi com Redes LVQ que obteve o menor valor, tendo média abaixo de 85%, taxa 15% menor dos 100% da Rede MLP. Esta redução deve-se principalmente a taxa de acuidade de 65,83% do treinamento e teste da arquitetura LVQ(6,6,2). Ao contrário da maioria das bases de dados que tiveram taxa de acuidade menor para a Rede LVQ, foi nesta RNA que a base de dados *Iris* obteve melhor média, aproximadamente 94%, valor próximo aos 100% da base de dados *Acute Inflammations* para a Rede MLP.

Assim como as Redes MLP e RBF, as Redes LVQ não tiveram dependência direta da taxa de acuidade de acordo com o número de registros, número de atributos ou quantidade de classes das bases de dados. O que pôde ser verificado é que as três bases de dados que obtiveram as melhores médias de taxa de acuidade (*Ecoli*, *Acute Inflammations* e *Iris*) não eram grandes com relação ao número de registros, número de atributos e número de classes, porém não se pode afirmar que uma base de dados nestas condições apresentará alta taxa de acuidade com a Rede LVQ, pois *Teaching Assistant Evaluation* e *Statlog(Image Segmentation)* apresentam estas características e não obtiveram altas taxas de acuidade.

Tabela 5.7: Acuidade das Redes LVQ nas 10 bases de dados escolhidas.

Base de Dados	Número de Neurônios por Camada			Acuidade Treinamento (%)	Acuidade Teste (%)	Média Acuidade
	Entrada	Oculto	Saída			
<i>Wine Quality</i>	11	7	7	0,22	0,22	0,19
	11	14	7	0,16	0,16	
	11	21	7	0,18	0,18	
<i>Abalone</i>	8	28	28	0,18	0,19	0,94
	8	56	28	0,36	0,36	
	8	84	28	2,28	2,30	
<i>Libras Movement</i>	90	15	15	11,05	9,72	15,76
	90	30	15	24,57	13,33	
	90	45	15	18,95	16,94	
<i>Statlog (Image Segmentation)</i>	19	7	7	34,52	34,07	33,07
	19	14	7	27,70	27,71	
	19	21	7	37,07	37,36	
<i>Teaching Assistant Evaluation</i>	5	3	3	52,72	36,67	46,36
	5	6	3	52,43	44,00	
	5	9	3	49,71	42,67	
<i>Spambase</i>	57	2	2	75,40	64,30	69,85
	57	4	2	75,40	64,30	
	57	6	2	75,40	64,30	
<i>Statlog (Landsat Satellite)</i>	36	6	6	76,06	75,47	71,31
	36	12	6	68,29	65,66	
	36	18	6	69,91	72,47	
<i>Ecoli</i>	7	8	8	67,82	67,88	75,97
	7	16	8	84,16	71,82	
	7	24	8	86,90	77,27	
<i>Acute Inflammations</i>	6	2	2	99,07	94,17	84,71
	6	4	2	91,67	91,67	
	6	6	2	65,83	65,83	
<i>Iris</i>	4	3	3	93,78	89,33	94,00
	4	6	3	96,74	93,33	
	4	9	3	96,81	94,00	

Tabela 5.8: Tempos de treinamentos das Redes LVQ.

Base de Dados	Número de Neurônios por Camada			Conexões * Tamanho Base	Tempo de Treinamento
	Entrada	Oculto	Saída		
<i>Acute Inflammations</i>	6	2	2	1680	939
<i>Acute Inflammations</i>	6	4	2	3360	1403
<i>Iris</i>	4	3	3	2250	1423
<i>Teaching Assistant Evaluation</i>	5	3	3	2718	1487
<i>Acute Inflammations</i>	6	6	2	5040	1808
<i>Iris</i>	4	6	3	4500	2062
<i>Teaching Assistant Evaluation</i>	5	6	3	5436	2222
<i>Iris</i>	4	9	3	6750	2661
<i>Teaching Assistant Evaluation</i>	5	9	3	8154	2835
<i>Ecoli</i>	7	8	8	21504	6767
<i>Ecoli</i>	7	16	8	43008	11721
<i>Ecoli</i>	7	24	8	64512	15636
<i>Libras Movement</i>	90	15	15	491400	81617
<i>Statlog (Image Segmentation)</i>	19	7	7	307230	82896
<i>Statlog (Image Segmentation)</i>	19	14	7	614460	128652
<i>Libras Movement</i>	90	30	15	982800	131326
<i>Wine Quality</i>	11	7	7	411432	144298
<i>Libras Movement</i>	90	45	15	1474200	174067
<i>Statlog (Image Segmentation)</i>	19	21	7	921690	175436
<i>Spambase</i>	57	2	2	533716	206149
<i>Wine Quality</i>	11	14	7	822864	209542
<i>Spambase</i>	57	4	2	1067432	276373
<i>Wine Quality</i>	11	21	7	1234296	296749
<i>Spambase</i>	57	6	2	1601148	308546
<i>Abalone</i>	8	28	28	1052604	311305
<i>Statlog (Landsat Satellite)</i>	36	6	6	1428570	331544
<i>Abalone</i>	8	56	28	2105208	510235
<i>Statlog (Landsat Satellite)</i>	36	12	6	2857140	538673
<i>Statlog (Landsat Satellite)</i>	36	18	6	4285710	693333
<i>Abalone</i>	8	84	28	3157812	719484

Na Tabela 5.8, observa-se o tempo de treinamento de cada Rede LVQ em cada base de dados. Foram nestes resultados que a Rede LVQ apresentou melhores resultados em relação às RNAs anteriores. O maior tempo de treinamento foi de 719484 ms, o que representa 7,46% do maior tempo de treinamento da Rede RBF que foi de 9516455 ms, e 1,17% do maior tempo de treinamento da Rede MLP (61041770 ms).

Semelhante aos testes anteriores, não foi observada relação entre tempo de treinamento e

taxa de acuidade.

5.4 Considerações Finais

Pode-se verificar que as três RNAs apresentaram algumas semelhanças e diferenças nos testes realizados. A principal semelhança refere-se à diversidade de valores de taxas de acuidade para o conjunto de bases de dados testadas, onde apresentaram valores baixos, médios e altos. Esta diversidade já era esperada, pois os resultados apresentados neste trabalho se assemelham aos observados em outros na literatura consultada.

Em [70] foram realizados testes em sete bases de dados, também disponíveis na UCI *Machine Learning Repository*, onde a Rede MLP apresentou tanto altas taxas de acuidade (78% na base de dados *Iris*, acuidade menor do que as registradas nos testes deste trabalho) quanto taxas mais baixas (36,57% e 36,74% nas bases de dados *Vowel* e *Thyroid*, respectivamente). A Rede LVQ apresentou alta taxa de acuidade (87,88% para a base de dados *Wiscosin*), porém foi a que apresentou o menor valor de todos os testes (11,11% na *Vowel*), a exemplo do que aconteceu aqui. A Rede RBF obteve os melhores resultados, atingindo um percentual de acuidade de 98,96% para a base de dados *2-Spiral*.

Em [38] foi realizada a comparação de quatro RNAs, três destas foram a Redes MLP, RBF e LVQ, em uma base de dados de câncer de mama, onde a Rede MLP obteve taxa de acuidade de 51,88%, a Rede RBF 49,79% e a Rede LVQ obteve o melhor resultado, 95,82%.

Para este trabalho, embora a maioria das bases de dados tenham tido grandes diferenças de taxas de acuidade nas três RNAs, as bases de dados *Iris* e *Acute Inflammations* foram as únicas bases de dados que tiveram sempre altas taxas de acuidade e a *Abalone* foi a única que apresentou taxas de acuidade baixas em todos os testes.

A fim de verificar a influência do número de épocas na média de acuidade, as três menores bases de dados com relação ao número de registros (*Iris*, *Acute Inflammations* e *Teaching Assistant Evaluation*) foram testadas com 4000 e 6000 épocas. A única RNA que não teve a média de acuidade prejudicada para nenhum caso foi a Rede MLP, onde a base de dados *Iris* teve média de acuidade acima de 99% para 4000 e 6000 épocas, a base de dados *Acute Inflammations* manteve os 100% de acerto e a base de dados *Teaching Assistant Evaluation* teve aumento de aproximadamente 2% nos dois casos. A Rede RBF apresentou pequenas reduções, não sendo superior a 5% para nenhuma das bases de dados, enquanto a Rede LVQ teve algumas diferenças mais significativas, tendo aumento de aproximadamente 11% da

média de acuidade para a base de dados *Acute Inflammations* com 4000 épocas e uma redução de 9% para a base de dados *Iris* treinada com 6000 épocas. Desta forma pode-se dizer que aumentar o número de épocas de treinamento beneficia a Rede MLP enquanto que para as Redes RBF e LVQ nada pode ser concluído, pois o resultado não apresentou um comportamento único no sentido de aumentar ou reduzir a média de acuidade.

A principal diferença observada entre as Redes MLP, RBF e LVQ refere-se ao tempo de treinamento. Para treinar todas as bases de dados em todas as arquiteturas, a Rede MLP demorou 91,15 horas, a Rede RBF 12,58 horas e a Rede LVQ 1,492 horas. Estas grandes diferenças devem-se aos algoritmos de treinamento das RNAs, pois enquanto no treinamento *backpropagation* da Rede MLP ocorre a correção dos valores dos pesos de todos os neurônios para cada registro de entrada em todas as épocas, no treinamento da rede RBF somente os pesos da camada de saída são atualizados para cada registro de entrada e a camada oculta é atualizada apenas uma vez para cada iteração do método *k-fold-cross-validation*, já o treinamento da rede LVQ executa a atualização dos pesos apenas de um neurônio por registro de entrada, o neurônio vencedor.

Algo que também pode ser verificado no treinamento das Redes MLP e RBF foi a relação direta entre a multiplicação do tamanho das bases de dados com o número de conexões da rede e o tempo de treinamento, conforme aumentava o valor deste número aumentava também o tempo de treinamento. Nestas duas RNAs, que são completamente conectadas, o número de conexões pode ser calculado através da Equação 5.1, onde m é o número de camadas da rede e $neurônios(i)$ indica o número de neurônios da camada i da RNA. Esta relação não foi possível verificar na Rede LVQ, possivelmente pelo fato de não ser completamente conectada.

$$\sum_{i=1}^{m-1} neurônios(i) * neurônio(i+1) \quad (5.1)$$

Pelo testes realizados foi possível verificar que ao selecionar uma base de dados (um problema) não é possível determinar com base em sua quantidade de registros, quantidade de classes e quantidade de atributos de entrada qual RNA é a mais adequada para aquele problema, nem mesmo prever qual será a sua taxa de acuidade, é algo que só pode ser determinado empiricamente.

O fato do desempenho de um algoritmo poder ser determinado apenas de forma prática já havia sido sugerido por Fayyad [24], que já afirmava que não existe uma técnica de

mineração de dados universal que se adapte a qualquer problema. Escolher um algoritmo muitas vezes depende da experiência do especialista que tem que resolver o problema dado, sendo que na prática grande parte do esforço se concentra em uma melhor definição do problema a ser resolvido. Logo, para determinar se uma RNA, ou outra técnica de mineração de dados, é adequada a um determinado problema, o mais indicado seria fazer um estudo de caso avaliando não só os parâmetros como taxa de aprendizado, taxa *momentum*, número de épocas e número de neurônios, mas também avaliar a determinação de quais atributos realmente influenciam no resultado de decisão da Classificação e de outras tarefas.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho tratou sobre a implementação das RNAs MLP, RBF e LVQ para a ferramenta YADMT e comparações das mesmas em diversas bases de dados, sendo que o principal objetivo do trabalho foi analisar as três RNAs sobre condições mais semelhantes possíveis, a fim de verificar potenciais semelhanças ou diferenças entre as mesmas, por isto elas foram testadas sobre as mesmas bases de dados, mesmo número de épocas, mesma taxa de aprendizado e mesma quantidade de neurônios quando possível.

Quanto à implementação, a ferramenta apresenta um protocolo simples que facilita o desenvolvimento de novos módulos para a tarefa Classificação, que consiste basicamente na importação de uma biblioteca, na implementação de uma Interface e na utilização de duas *Annotations*. Para instalação basta colocar o arquivo com extensão *JAR* (gerado pelo compilador Java) e as bibliotecas adicionais em uma pasta própria da ferramenta YADMT.

Para a execução dos testes houve certa facilidade em conseguir as bases de dados pelo fato de as mesmas terem sido obtidas do repositório público da *UCI Machine Learning Repository*, que oferece grande quantidade de material, principalmente no que se refere a dados para a tarefa de Classificação. A disponibilização destes dados facilita e incentiva o desenvolvimento e testes de outras técnicas sobre essas bases de dados, permitindo comparações diretas entre as mesmas.

No que se refere à acuidade, as RNAs apresentaram tanto bons quanto maus resultados, porém nenhuma base de dados teve valores próximos para as três RNAs. A Rede MLP foi a RNA que obteve a maior taxa de acuidade entre as três, 100% para a base de dados *Acute Inflammations*, e a Rede LVQ foi a que obteve o pior desempenho, média de 0,19% para a base de dados *Wine Quality*. No entanto, a principal diferença entre as RNAs refere-se ao tempo de treinamento, característica que depende diretamente dos algoritmos de aprendizado utilizados, sendo que a rede MLP é a que apresentou os maiores tempos de treinamento para

uma mesma quantidade de dados e épocas, enquanto a Rede LVQ é a que apresentou os menores tempos. Neste contexto, a Rede RBF apresentou desempenho intermediário na média geral das acurácies e para tempo de treinamento.

Como trabalhos futuros indicam-se:

- Implementação de novos módulos para a ferramenta YADMT, tanto de Classificação para aumentar a abrangência de técnicas da ferramenta para esta tarefa quanto outros módulos, por exemplo, o de pré-processamento que faltaram para a realização dos teste, e se implementados darão suporte a outros algoritmos.
- Um estudo mais profundo para as bases *Ecoli* e *Libras Movement*, a fim de verificar se a causa de *overfitting* nas RNAs foi realmente o fato destas bases de dados estarem com os dados ordenados de acordo com as classes. Como sugestões, há o “embaralhamento” dos dados, o aumento do valor de k do método *k-fold-cross-validation*, a redução do número de épocas de treinamento, o aumento do número de camadas da Rede MLP ou ainda, testar outros métodos de avaliação.
- Um estudo de caso envolvendo as RNAs aqui apresentadas e explorar melhor os parâmetros de configurações possíveis dessas técnicas.

Apêndice A

Descrição das Bases de Dados

Segue-se abaixo uma descrição das dez bases de dados utilizadas para os testes das RNAs MLP, RBF e LVQ. Elas foram obtidas do repositório público UCI *Machine Learning Repository* [54].

A.1 Base de Dados *Abalone*

A base de dados *Abalone* foi criada por Sam Waugh do Departamento de Ciência da Computação da Universidade da Tasmânia, em 1995. Consiste de informações relacionadas a um molusco chamado Haliote, que serve de alimento principalmente nos países asiáticos. Os dados foram inicialmente propostos no estudo [53], que analisava os moluscos a partir das características físicas e determinava qual seria a idade do mesmo para o consumo, e posteriormente foram retiradas as tuplas que continham valores ausentes. O conjunto consiste de:

- 1 de atributo categórico (sexo),
- 6 atributos numéricos (comprimento, diâmetro, altura, peso bruto, peso do molusco descascado, peso das vísceras, peso da concha),
- 1 atributo classificador (anéis), utilizado para determinar a idade do haliote.

A base de dados consiste de 4177 amostras e a distribuição dos dados nas classes segue na Tabela A.1.

Tabela A.1 - Distribuição de classes da Base de Dados *Abalone*.

Classe	Nº de Exemplos
1	1
2	1
3	15
4	57
5	115
6	259
7	391
8	568
9	689
10	634
11	487
12	267
13	203
14	126
15	103
16	67
17	58
18	42
19	32
20	26
21	14
22	6
23	9
24	2
25	1
26	1
27	2
28	1

A.2 Base de Dados *Acute Inflammations*

A base de dados *Acute Inflammations* foi criada por Jacek Czerniak, do Instituto de Pesquisa de Sistemas da Acadêmica Polonesa de Ciências, e está disponível na UCI desde 2009. Apresenta dados criados para testar um sistema especialista, o qual deveria realizar o diagnóstico presumível de doenças do sistema urinário. Originalmente esta base possuía 8 atributos sendo:

- 1 atributo numérico, indicando a temperatura do paciente,
- 7 atributos categóricos (ocorrência de náuseas, dor lombar, necessidade constante de urinar, dores de micção, dores de uretra, ocorrência de inflamação urinária na bexiga e nefrite de origem pelve renal), que podem assumir valores *yes* ou *no*, destes, os dois últimos representam atributos de decisão de classe baseado nos seis primeiros atributos.

Para este trabalho eliminou-se o último atributo classe (nefrite de origem pelve renal), ficando a base de dados com um total de 7 atributos, incluso o atributo que define a classe “*inflamação urinária na bexiga*”. Esta base de dados contém um total de 120 amostras com a distribuição das classes de acordo com a Tabela A.2.

Tabela A.2 - Distribuição de classes da Base de Dados *Acute Inflammations*.

Classe	Nº de Exemplos
yes	59
no	61

A.3 Base de Dados *Ecoli*

A base de dados *Ecoli* foi criada por Kenta Nakai cientista do Instituto de Biologia Celular e Molecular da Universidade de Osaka em 1996. Contêm dados sobre a localização de proteínas em células. Originalmente consistia de:

- 1 atributo categórico (*Sequence Name*, em inglês), o qual foi retirado pois serve apenas para identificação do registro dentro do banco de dados,
- 7 atributos numéricos de medições de características de células;
- um último atributo que define a classe do objeto, neste caso a localização da célula que tem a maior concentração de proteína.

Esta base contém 336 amostras e as classes estão distribuídas de acordo com a Tabela A.3.

Tabela A.3 - Distribuição de classes da Base de Dados *Ecoli*.

Classe	Nº de Exemplos
cp (citoplasma)	143
im (membrana interna sem sinal de seqüência)	77
pp (periplasma)	52
imU (membrana interna, com sinal de seqüência não clivável)	35
om (membrana externa)	20
omL (membrana externa de lipoproteína)	5
imL (membrana interna de lipoproteína)	2
imS (membrana interna, com sinal de seqüência clivável)	2

A.4 Base de Dados *Iris*

A base de dados *Iris*, bastante conhecida, foi criada por Ronald Aylmer Fisher em 1988. Contém dados de três tipos de flores conhecidas por *Iris*: *Setosa*, *Vesicolour* e *Virginica*. Além do atributo de classificação possui outros quatro atributos que trazem informações de tamanho e espessura da sépala e tamanho e espessura de pétala, com todas as dimensões dadas em centímetros.

Sabe-se que classe *Setosa* é linearmente independente das outras, que não o são entre si. A base de dados é composta por 150 amostras e a distribuição igualitária dos dados é dada pela Tabela A.4.

Tabela A.4 - Distribuição de classes da Base de Dados *Iris*.

Classes	Nº de Exemplos
<i>Iris Setosa</i>	50
<i>Iris Versicolour</i>	50
<i>Iris Virginica</i>	50

A.5 Base de Dados *LIBRAS Movement*

A base de dados *LIBRAS Movement* foi criada por Daniel Baptista Dias, Sarajane Marques Peres e Helton Hideraldo Bísvaro, da Universidade de São Paulo - USP. Esta base de dados contém informações de movimentos da mão que representam o tipo de sinal *LIBRAS* (Língua Brasileira de Sinais). Esta base de dados é composta por 360 amostras com a igualitária distribuição das classes de acordo com a Tabela A.5. É composta de 90 atributos numéricos que

representam a posição da mão em cada instante de tempo em um vídeo de 45 frames, além de um atributo categórico que define a classe do objeto, neste caso um dos 15 tipos de movimento LIBRAS.

Tabela A.5 - Distribuição de classes da Base de Dados *LIBRAS Movement*.

Classe	Nº Exemplos
<i>curved swing</i>	24
<i>horizontal swing</i>	24
<i>vertical swing</i>	24
<i>anti-clockwise arc</i>	24
<i>clockwise arc</i>	24
<i>circle</i>	24
<i>horizontal straight-line</i>	24
<i>vertical straight-line</i>	24
<i>horizontal zigzag</i>	24
<i>vertical zigzag</i>	24
<i>horizontal wavy</i>	24
<i>vertical wavy</i>	24
<i>face-up curve</i>	24
<i>face-down curve</i>	24
<i>tremble</i>	24

A.6 Base de Dados *Spambase*

A base de dados *Spambase* foi criada por Mark Hopkins, Erik Reeber, George Forman e Jaap Suermondt do Laboratório da Hewlett-Packard (HP) em 1999, contém 4601 amostras e as classes estão distribuídas de acordo com a Tabela A.6.

Esta base de dados representa dados de vários e-mails que foram classificados como spam ou não-spam. Considera-se spam e-mails não solicitados que fazem propagandas de produtos e de sites, esquemas de ganho rápido de dinheiro, e-mails de correntes, pornografia, entre outros, e é composta de:

- 48 atributos numéricos contínuos que representam a porcentagem de palavras do e-mail que correspondente a uma determinada palavra;
- 6 atributos numéricos contínuos que representam a porcentagem de caracteres do e-mail que correspondem determinado caractere;

- 1 atributos numérico contínuo que representa o tamanho médio de letras maiúscula em seqüência;
- 1 atributo numérico inteiro que representa o tamanho da maior seqüência ininterrupta de letras maiúsculas;
- 1 atributo numérico inteiro que representa a soma de todas as seqüência ininterruptas de letras em maiúsculas;
- além de um atributos numérico inteiro que define a classe do objeto como sendo ou não um spam.

Tabela A.6 - Distribuição de classes da Base de Dados *Spambase*.

Classe	Nº de Exemplos
1 (spam)	1813
0 (não-spam)	2788

A.7 Base de Dados *Teaching Assistant Evaluation*

A base de dados *Teaching Assistant Evaluation* foi criada por Wei-Yin Loh do Departamento de Estatística da Universidade de Wisconsin - Madison em 1997. Consiste de dados da avaliação de desempenho de assistentes de ensino (TA - *Teaching Assistant*) designados no Departamento de Estatística da Universidade de Wisconsin-Madison. É composta por 6 atributos numéricos, o primeiro atributo indica se o TA tem ou não o idioma Inglês como seu idioma nativo, o segundo atributo indica qual o instrutor do curso, o terceiro atributo indica o curso, o quarto atributo indica se é um semestre regular ou um semestre de verão, o quinto atributo indica o tamanho da turma e, por fim, o sexto atributo indica a pontuação do assistente como pequena, média ou grande.

Esta base de dados contém 151 amostras e as classes estão distribuídas nas três classes de acordo com a Tabela A.7.

Tabela A.7 - Distribuição de classes da Base de Dados *Teaching Assistant Evaluation*.

Classe	Nº de Exemplos
1 (pequena)	49
2 (média)	50
3 (grande)	52

A.8 Base de Dados *Statlog (Image Segmentation)*

A base de dados *Statlog (Image Segmentation)* foi criada pelo Vision Group da Universidade de Massachusetts em 1990. Contém 19 atributos numéricos contínuos (Tabela A.8) que representam várias informações sobre segmentos de imagem de tamanho 3x3 pixels, mais um atributo numérico que define a classe do objeto, cuja informação representa um tipo de imagem (tijolo, céu, folhagem, cimento, janela, caminho e grama).

Tabela A.8 - Atributos da Base de Dados *Statlog (Image Segmentation)*.

1	<i>region-centroid-col</i> : coluna da região central do pixel.
2	<i>region-centroid-row</i> : linha da região central do pixel.
3	<i>region-pixel-count</i> : número de pixels na região (sempre 9).
4	<i>short-line-density-5</i> : determina quantas linhas de comprimento 5 (qualquer orientação) de baixo contraste, inferior ou igual a 5, percorrendo a região.
5	<i>short-line-density-2</i> : mesmo que <i>short-line-density-5</i> , mas conta linhas de alto contraste, superiores ou iguais a 5.
6	<i>vedge-mean</i> : medida média de contraste de pixels adjacentes horizontais. Este atributo é usado como um detector vertical de arestas.
7	<i>vedge-sd</i> : desvio padrão do atributo 6
8	<i>hedge-mean</i> : medida média de contraste de pixels adjacentes verticais. Usado como detector de linhas horizontais.
9	<i>hedge-sd</i> : desvio padrão do atributo 8
10	<i>intensity-mean</i> : média de intensidade sobre a região $(R + G + B)/3$
11	<i>rawred-mean</i> : média de intensidade da região no canal R.
12	<i>rawblue-mean</i> : média de intensidade da região no canal B.
13	<i>rawgreen-mean</i> : média de intensidade da região no canal G.
14	<i>exred-mean</i> : medida de excesso de vermelho $(2R - (G + B))$
15	<i>exblue-mean</i> : medida de excesso de azul $(2B - (G + R))$
16	<i>exgreen-mean</i> : medida de excesso de verde $(2G - (R + B))$
17	<i>value-mean</i> : valor da transformação 3-d não-linear.
18	<i>saturatoin-mean</i> : saturação da transformação 3-d não-linear.
19	<i>hue-mean</i> : matiz da transformação 3-d não-linear.

Esta base de dados contém 2310 amostras distribuídas de acordo com a Tabela A.9.

Tabela A.9 - Distribuição de classes da Base de Dados Statlog (*Image Segmentation*).

Classe	Nº de Exemplos
1	330
2	330
3	330
4	330
5	330
6	330
7	330

A.9 Base de Dados Statlog (*Landsat Satellite*)

A base de dados Statlog (*Landsat Satellite*) foi criada por Ashwin Srinivasan do Departamento de Estatística e Modelagem de Dados da Universidade de Strathclyde em 1993. Contém valores multi-espectrais de segmentos de tamanho 3x3 pixels de uma imagem de satélite e a respectiva classificação associada com o pixel central de cada imagem. É composta por 36 atributos numéricos na faixa de 0 a 255, que representam as 4 bandas espectrais dos 9 pixels dos segmentos de imagem, e 1 atributo numérico classificador do pixel central.

Esta base de dados é composta por 6435 amostras e as classes estão distribuídas de acordo com a Tabela A.10.

Tabela A.10 - Distribuição de classes da Base de Dados Statlog (*Landsat Satellite*).

Classe	Nº de Exemplos
1 - latossolo vermelho (<i>red soil</i>)	1533
2 - cultura de algodão (<i>cotton crop</i>)	703
3 - latossolo acinzentado (<i>grey soil</i>)	1358
4 - latossolo acinzentado com humidade (<i>damp grey soil</i>)	626
5 - solo com vegetação seca (<i>soil with vegetation stubble</i>)	707
6 - mistura de classes (<i>mixture class</i>)	0
7 - latossolo acinzentado com muita humidade (<i>very damp grey soil</i>)	1508

A. 10 Base de Dados Wine Quality

A base de dados Wine Quality foi criada por Paulo Cortez, Antonio Cerdeira, Fernando Almeida, Telmo Matos e Jose Reis da Universidade de Minho, Guimarães - Portugal em 2009. Esta base de dados contém informações físico-químicas da variação do vinho Português

de cor vermelha. É composta por 11 atributos numéricos contínuos que representam as características físico-químicas do vinho (*fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol*) e um atributo quantitativo que representa a qualidade do vinho.

Esta base apresenta 4898 amostras cuja distribuição das classes está definida na Tabela A.11.

Tabela A.11 - Distribuição de classes da Base de Dados *Wine Quality*.

Classe	Nº de Exemplos
3	20
4	163
5	1457
6	2198
7	880
8	175
9	5

Referências

- [1] ADRISANS, P., ZANTINGE, D. *Data Mining*. Addison-Wesley, 1997.
- [2] AL-NOUKARI, M., AL-HUSSAN, W. *Using Data Mining Techniques for Predicting Future Car market Demand; DCX Case Study*, Information and Communication Technologies: From Theory to Applications. ICTTA, p. 1-5, Abril, 2008.
- [3] ALTMAN, D. G., BLAND, J.M. *Diagnostic tests 1: sensitivity and specificity*. British Medical Journal – BMJ, p. 1552, 1994.
- [4] AMO, S. A. DE. *Curso de Data Mining*. Universidade Federal de Uberlândia, Faculdade de Computação, Agosto 2003. Disponível em: <http://www.deamo.prof.ufu.br/CursoDM2008.html>
- [5] BARLETT, E. B., UHRIG, R. E. *Nuclear power plant status diagnostics using artificial neural networks*. International conference on frontiers in innovative computing for the nuclear industry, Jackson-WY-United States, v. 97, p. 272–281, Janeiro, 1991.
- [6] BARRETO, G. A. *Rede de Funções de Base Radial (RBF)*, Apostila de Aula. Universidade Federal do Ceará (UFC), Departamento de Engenharia de Teleinformática, Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Fortaleza-CE, Maio, 2007. Disponível em: <http://www.deti.ufc.br/~guilherme/TIP705/rbf.pdf>. Último acesso em 04/11/2010.
- [7] BAXT, W. G. *Use of an artificial neural network for data analysis in clinical decision-making: The diagnosis of acute coronary occlusion*. Neural Computaion, v. 2, p. 480–489, 1990.
- [8] BAXT, W. G. *Use of an artificial neural network for the diagnosis of myocardial infarction*. Annals of Internal Medicine, v. 115, p. 843–848, 1991.
- [9] BENFATTI, E. W., BOSCARIOLI, C., GIRARDELLO A. D., BONIFÁCIO F. N. *YADMT - Yet Another Data Mining Tool*. Universidade Estadual do Estado do Paraná - UNIOESTE, Centro de Ciências Exatas e Tecnológicas – CCET, Curso de Ciência da Computação, Novembro, 2010. Relatório Técnico número 1.
- [10] BIGUS, J. P. *Data Mining with Neural Networks: Solving Business Problems — From Application Development to Decision Support*, McGraw-Hill, New York, 1996.

- [11] BOURLARD, H. E MORGAN, N. *Continuous speech recognition by connectionist statistical methods*. IEEE Transactions on Neural Networks, v. 4, p. 893–909, Novembro, 1993.
- [12] BRAGA, A., CARVALHO, A., LUDERMIR, T. *Redes Neurais Artificiais: Teoria e Aplicações*, Livro Técnico e Científico, Rio de Janeiro, 2000.
- [13] BURGESS, J. *A tutorial on support vector machines for pattern recognition*. Data Mining Knowledge Discovering, p. 121–167, 1998.
- [14] CARPENTER, G.A., GROSSBERG, S. *The ART of adaptive pattern recognition by a self-organizing neural network*, Computer, v.21, n.3, p.77-88, Março, 1988.
- [15] CHANG, C., FU, S. *Image Classification using a Module RBF Neural Network*. First International Conference on Innovative Computing, Information and Control, Beijing, p. 270 - 273, Setembro, 2006.
- [16] CORMEN, T.H., LEISERSON, C.E., RIVEST, R. L., STEIN, C. *Introduction to Algorithms*, 2. Ed., MIT Press & McGraw-Hill, 2001.
- [17] COVER, T. M. *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*, Electronic Computers, IEEE Transactions on, v. 14, n. 3, p. 326-334, 1965.
- [18] CRISTIANINI, N. SHAW-TAYLOR, J. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.
- [19] CYBENKO, G. *Continuous valued neural networks with two hidden layers are sufficient*. Technical report, Department of Computer Science, Tufts University, 1988.
- [20] CYBENKO, G. *Approximation by superpositions of a sigmoid function*. Mathematics of Control, Signals and Systems, 2:303-314, 1989.
- [21] FAYYAD, U. M., UTHURUSAMY, R. *Proceedings of KDD-95: The First International Conference on Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, California, 1995.
- [22] FAYYAD, U. M., PIATETSKY-SHAPIRO, G., SMYTH, P. *From data mining to knowledge discovery: an overview*, Advances in knowledge discovery and data mining, American Association for Artificial Intelligence, Menlo Park, CA, 1996.
- [23] FAYYAD, U. M., PIATETSKY-SHAPIRO, G., SMYTH, P. *From Data Mining to Knowledge Discovery in Databases*, AI Magazine v. 17, n. 3, p. 37-54, 1996.
- [24] FAYYAD, U. M., PIATETSKY-SHAPIRO, G., SMYTH, P. *The KDD process for extracting useful knowledge from volumes of data*, Communications of the ACM, v. 39, n.11, p.27-34, Novembro, 1996.

- [25] FAYYAD U. M., PIATETSKY-SHAPIRO G., SMYTH, P. *Knowledge Discovery and Data Mining: Towards a Unifying Framework*, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, p. 82-88, 1996.
- [26] FUNUHASHI, K. I. *On the approximate realization of continuous mappings by neural networks*. Neural Networks, n. 2, p. 183-192, 1989.
- [27] GESSAROLI, J. *Data Mining: A powerful technology for database marketing*, Tele-marketing, v. 13, n. 11, p. 64-68, Maio, 1995.
- [28] GOLDBERG, D. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
- [29] GOVINDARAJAN, M., CHANDRASEKARAN, R. M. *Performance optimization of data mining application using radial based function classifier*, PWASET, v. 50, p. 557, 2009.
- [30] HAN, J., KAMBER, M. *Data Mining: Concepts and Techniques*. Editora Morgan Kaufmann. 2005. Segunda Edição.
- [31] HAYKIN, S. *Adaptive Filter Theory*, 3. ed., Prentice Hall, Upper Saddle River, New Jersey, 1996.
- [32] HAYKIN, S. *Redes neurais: princípios e prática*. 2.ed. Porto Alegre, Bookman, 2001.
- [33] HERTZ, J., KROGH, A., R. G. Palmer. *Introduction to the Theory of Neural Computation*, Lecture Notes Volume I of Santa Fe Institute Studies in The Science of Complexity. Addison-Wesley, 1991.
- [34] HORNIK, K., STINCHCOMBE, M., WHITE, H. *Multilayer feedforward networks are universal approximators*. Neural Networks, n. 2, p. 359-366, 1989.
- [35] HOSKINS, J. C., KALIYUR, K. M., E HIMMELBLAU, D. M. *Incipient fault detection and diagnosis using artificial neural networks*. International Joint Conference on Neural Networks, San Diego - CA, v. 1, 81-86, Junho, 1990.
- [36] HU, Y., H., HWANG, J. *Handbook of Neural Network Signal Processing*, CRC Press, New York 2002.
- [37] IBM Intelligent Miner. Disponível em: http://www-947.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_Intelligent_Miner_Modeling, Última consulta em: 30/03/2010.
- [38] JANGHEL, R. R. *Breast cancer diagnosis using Artificial Neural Network models*, Information Sciences and Interaction Sciences (ICIS), 3rd International Conference on, p. 89-94, Junho, 2010.

- [39] JAIN, A. K., DUBES, R. C. *Algorithms for Clustering Data*. Englewood Cliffs, Prentice-Hall, 1988.
- [40] KOHAVI, R. *A study of cross-validation and bootstrap for accuracy estimation and model selection*, Proceedings of the 14th international joint conference on Artificial intelligence, p. 1137-1143, Montreal, Quebec, Canada, Agosto, 1995.
- [41] KOHONEN, T. K. *Self-Organizing Maps*. Springer-Verlag, Berlin, Heidelberg, 2nd extended edition, 1997.
- [42] KOHONEN, T. K. *The Self-Organizing Map*, Proceedings of the IEEE, v. 78, n. 9, p. 1464-1480, Setembro, 1990.
- [43] KORTH, H., SILBERSCHARTZ, A. *Sistemas de Banco de Dados*. 1. ed. São Paulo - SP: Campus Ltda, 2006.
- [44] KOSAKA, T., TAKETANI, N., OMATU, S., RYO, K. *Discussion of reliability criterion for US dollar classification by LVQ*. Proceedings of the 1999 IEEE Midnight-Sun Workshop on Soft Computing Methods in Industrial Applications, Kuusamo, 28 - 33, Junho, 1999.
- [45] KOVACS, K. L. *Redes Neurais Artificiais - Fundamentos e Aplicações*. Editora Acadêmica. São Paulo, 1996.
- [46] LIND, Y., BUZO, A., GRAY, R. *An Algorithm for Vector Quantizer Design*, IEEE Trans. on Communications, v. 28, Janeiro, 1980.
- [47] MARCANO-CEDENO, A., ALVAREZ-VELLISCO, A., ANDINA, D. *Artificial metaplasticity MLP applied to image classification*. 7th IEEE International Conference on Industrial Informatics. Cardiff, Wales, p. 650 - 653, Junho 2009.
- [48] MARGARITA, S., BELTRATTI, A. *Credit Risk and lending in an artificial adaptive banking system*, Adaptive Intelligent Systems - Proceedings of the BANKAI Workshop, p. 161 - 176, 1992.
- [49] MCCULLOCH, W.S. E PITTS, S. *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics, v. 9, p. 127-147, 1943.
- [50] MENDEL, J. M., MACLAREN, R. W. *Reinforcement learning control and pattern recognition systems*, in Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications, J. M. Mendel and K. S. Fu, Eds. New York: Academic Press, p. 287-318, 1970.
- [51] MENZIES, T., HU, Y. *Data Mining For Busy People*. IEEE Computer, p. 18-25, Outubro, 2003.
- [52] MOODY, J. E., DARKEN, C. *Fast learning in networks of locally-tuned processing units*. Neural Computation, v. 1, n.1, p. 281-294, 1989.

- [53] NASH, W. J., SELLERS, T. L., TALBOT, S. R., CAWTHORN, A. J., FORD, W. B. *The Population Biology of Abalone (_Haliotis_species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait*, Sea Fisheries Division, Technical Report No. 48, 1994.
- [54] NATIONAL SCIENCE FOUNDATION. UCI Machine Learning Repository. Disponível em: <http://archive.ics.uci.edu/ml/>. Último acesso: 30 de Março de 2010.
- [55] NORTON M. *Close to the nerve (credit card fraud)*, Banking Technology (UK), v. 10, p. 28-31, Dezembro, 1994.
- [56] ORACLE Data Mining. Disponível em: <http://www.oracle.com/technology/documentation/darwin.html>, Consultado na Internet em: 30/03/2010.
- [57] ORANGE. Disponível em: <http://www.ailab.si/orange/>, Última consulta em: 30/03/2010.
- [58] OZEKES S., OSMAN O. *Classification and Prediction in Data Mining with Neural Networks*, Istanbul University Journal of Electrical & Electronics, v. 3, n. 1, p. 707-712, 2003.
- [59] PACHECO, M. A. C., VELLASCO, M., LOPES, C. H. *Descoberta de Conhecimento e Mineração de Dados*, Notas de Aula em Inteligência Artificial. Rio de Janeiro, ICA – Laboratório de Inteligência Computacional Aplicada, Engenharia Elétrica PUC-RIO. Disponível em :<http://www.ica.ele.puc-rio.br>. Último acesso em: 30/03/2010.
- [60] PACHECO, M. A. C. *Algoritmos Genéticos: Princípios e Aplicações*. Apostila de Aula. Versão 1. Pontifícia Universidade Católica do Rio de Janeiro, ICA: Laboratório de Inteligência Computacional Aplicada - Julho, 1999. Disponível em <http://www.ica.ele.puc-rio.br/Downloads/38/CE-Apostila-Comp-Ev.pdf>. Último acesso em: 04/11/2010.
- [61] PEREIRA, B. DE B., RAO, C. R. *Data Mining Using Neural Networks: A Guide for Statisticians*. State College, Pennsylvania, 2009.
- [62] PERES S. M. *LVQ - Learning Vector Quantization*. Notas de Aula. Disponível em: http://www.ic.unicamp.br/~luciano/ACH2016/notasdeaula/08_LearningVectorQuantization.pdf. Último acesso em: 27/04/2010.
- [63] RUMELHART, D.E., HINTON, G.E., WILLIAMS, R.J. *Learning internal representation by error propagation*. In *Parallel Distributed Processing*, p. 318-362, Cambridge, MA, MIT Press, 1986.
- [64] RUMELHART, D. E., MCCLELLAND, J. L. *Parallel distributed processing: explorations in the microstructure of cognition*, Psychological and biological models, MIT Press, Cambridge, MA, 1986.

- [65] RUSSEL, S. J. AND NORVIG, P. *Artificial intelligence: a modern approach*. Upper Saddle River: Prentice-Hall, 1995.
- [66] SAS Enterprise Miner. Disponível em: <http://www.sas.com/technologies/analytics/datamining/miner/>, Última consulta em: 30/03/2010.
- [67] SABBATINI, R. M. E. *Applications of Connectionist System in Biomedicine*. MEDINFO, p. 428-425, 1992.
- [68] SHEN, X., CHEN, J. *Study on Prediction of Traffic Congestion Based on LVQ Neural Network*, Measuring Technology and Mechatronics Automation. ICMTMA '09. International Conference on , v.3, p.318-321, Abril, 2009.
- [69] SAPNA, S., TAMILARASI, A. *Fuzzy Relational Equation in Preventing Diabetic Heart Attack*, Advances in Recent Technologies in Communication and Computing. International Conference on, p.635-637, Outubro, 2009.
- [70] SERPEN, G., JIANG, H., ALLRED, L. *Performance analysis of probabilistic potential function neural network classifier*, In Proceedings of artificial neural networks in engineering conference, St. Louis, MO, v.7, p. 471-476, 1997.
- [71] SHEBLE, G. *Artificial life techniques for market data mining and emulation*, IEE Power Engineering Society Winter Meeting, v.3, p. 1696, Janeiro, 2000.
- [72] SILVA, L. S., SANTOS, A. C. F., MONTES, A., SIMOES, J. D. S. *Hamming Net and LVQ Neural Networks for Classification of Computer Network Attacks: A Comparative Analysis*. Ninth Brazilian Symposium on Neural Networks, Ribeirão Preto, Brazil, p. 72 - 77, Outubro, 2006.
- [73] SOUSA, M. S. R. *Mineração de Dados: Uma Implementação Fortemente Acoplada a um Sistema Gerenciador de Banco de Dados Paralelo*. COPPE - Universidade Federal do Rio de Janeiro, 1998. Dissertação.
- [74] SUG, H. *Performance comparison of RBF networks and MLPs for classification*. In Proceedings of the 9th WSEAS international Conference on Applied informatics and Communications. N. E. Mastorakis, M. Demiralp, V. Mladenov, and Z. Bojkovic, Eds. Recent Advances In Computer Engineering. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, p. 450-454, Agosto, 2009.
- [75] TANAGARA. Disponível em: <http://eric.univ-lyon2.fr/~ricco/tanagra/>. Última consulta em: 30/03/2010.
- [76] XIAO, Y., LEI, L. *Research on comparison of credit risk evaluation models based on SOM and LVQ neural network*, Intelligent Control and Automation, 7th World Congress on, p. 2230-2235, Junho, 2008.

- [77] XIONG, X., KIM, Y., BAEK, Y., RHEE, D. W., KIM, S. *Analysis of breast cancer using data mining & statistical techniques*, Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN. Sixth International Conference on, p. 82- 87, Maio, 2005
- [78] XUAN, K., ZHAO, G., TANIAR, D., SRINIVASAN, B., SAFAR, M., AND GAVRILOVA, M. *Network Voronoi Diagram Based Range Search*. In Proceedings of the International Conference on Advanced information Networking and Applications. IEEE Computer Society, Washington, DC, p. 741-748, 2009.
- [79] WEISS, S. I., KULIKOWSKI, C. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Networks, Machine Learning, and Expert Systems*. San Francisco, Morgan Kaufmann, 1991.
- [80] WEKA. Disponível em: <http://www.cs.waikato.ac.nz/~ml/weka/>. Último acesso em: 30/03/2010.
- [81] WESTPHAL, C., BLAXTON, T. *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*. John Wiley & Sons, Inc., 1998.
- [82] WIDROW B., STEARNS, S. D. *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliff, 1985.
- [83] WIDROW, B., HOFF, M.E., JR. *Adaptive Switching Circuits*, IRE WESCON Convention Record, v. 4, p. 96-104, Agosto, 1960.
- [84] ZHANG, G. P. *Neural networks for classification: a survey*. IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews, v. 30 n. 4, p. 451-462, 2000.
- [85] ZHANG, R., SUNDARARAJAN, N. SARATCHANDRAN, P. *An efficient sequential RBF network for bio-medical classification problems*. IEEE International Joint Conference on Neural Networks, v. 3, p. 2477-2482, Janeiro, 2005.
- [86] ZHENG, Z. *A benchmark for classifier learning*. Basser Department of Computer Science, N.S.W Australia, Technical Report TR474, 2006.