Laboratório de Computação Evolucionária
Departamento de Engenharia Elétrica – UFMG
Av. Pres. Antônio Carlos, 6627 – CEP 31.270-010
Fone: (5531) 3409 34 26 – (5531) 3409-4826

Aplicação de Técnicas de Mineração de Dados em Problemas de Classificação de Padrões

Luciana Gomes Castanheira

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da UFMG como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. João Antônio de Vasconcelos

UFMG – Belo Horizonte

09 / 2008

Agradecimentos

Aos meus pais, pelo estudo que me proporcionaram nesta jornada até aqui e pelo exemplo de vida.

Ao David e minhas irmãs, pela compreensão e incentivo todos os dias.

Ao meu Professor orientador, João Antônio de Vasconcelos, pelos ensinamentos, atenção e paciência que foram essenciais ao desenvolvimento deste trabalho.

A todas às pessoas que de alguma forma fizeram este sonho se tornar uma realidade.

Resumo

O processo de descoberta de conhecimento em bases de dados (*Knowledge Discovery in Databases* – KDD), incluindo a fase de mineração de dados, vem sendo amplamente utilizado como ferramenta para auxiliar na tomada de decisão em áreas como crédito bancário e predições médicas. Neste trabalho este processo de KDD é estudado tendo como objetivo avaliar a utilização de métodos de mineração de dados aplicados em áreas da engenharia elétrica, sendo a abordagem feita sobre uma base de dados oriunda de testes de cromatografia de transformadores de potência. A mineração de dados é aplicada para obter uma classificação de tipos de defeitos dos transformadores. As técnicas abordadas no trabalho são redes neurais e árvores de decisão. As estruturas de algoritmos escolhidas nestas técnicas foram, respectivamente, a rede MLP com treinamento através do algoritmo de retropropagação resiliente, simulada no MatLab, e a árvore gerada pelo algoritmo J4.8, simulada no aplicativo weka. O capítulo 2 traz um estudo sobre o processo de KDD, com as fases de todo o processo e suas respectivas atividades. No capítulo 3 é apresentada a fase de mineração de dados, realçando suas diversas aplicações. Nos capítulos 4 e 5 são feitos estudos das técnicas e seus respectivos algoritmos. Em seguida são apresentadas duas bases de dados consideradas benchmark para a validação do estudo e finalmente os algoritmos são aplicados à base de dados da cromatografia. Nos capítulos 7 e 8 são apresentados os resultados e as conclusões do trabalho, onde é visto que o processo de mineração de dados pode ser aplicado em problemas na área da engenharia elétrica, porém devem ser feitos estudos sobre o domínio de cada base de dados a ser tratada.

Palavras Chave: Redes Neurais, Árvores de Decisão, Mineração de Dados, KDD, Tomada de Decisão.

Abstract

The Knowledge Discovery in Databases (KDD) process, which includes the data mining phase, has been widely used as a tool to assist decision-making in areas such as banking credit and medical predictions. In this work this process is studied with the objective of evaluating the use of data mining methods in areas of electrical engineering, considering data obtained from chromatography tests of power transformers. The data mining is applied for a classification of the types of transformers´s defects. The techniques that were studied in this work are neural networks and decision trees. The algorithms chosen in these techniques are, respectively, MLP´s network with resilient backpropagation algorithm for the training, simulated in Matlab, and the tree generated by the J4.8 algorithm, simulated in Weka. Chapter 2 presents a study about the KDD´s process, with the phases of the whole process and their activities. In Chapter 3 is given the data mining phases, highlighting its various aplications. In Chapters 4 and 5 the studies of the techniques and your algorithms are presented. Then are given two databases considered benchmark for the study validation and finally the algorithms are applied in the database of chromatography. In the following sections the results and conclusions are presented, where it is seen that the data mining can be applied to problems in the electrical engineering area, but must be made studies on the area of each database to be treated.

Key-Words: Neural Networks, Decision Tree, Data Mining, KDD, Decision Making.

Lista de Figuras

Índice

1 Introdução

Durante os últimos anos tem se verificado um crescimento substancial da quantidade de dados armazenados em meios magnéticos. Segundo *FAYYAD et al.* [12], estes dados, produzidos e armazenados em larga escala, são inviáveis de serem lidos ou analisados por especialistas através de métodos tradicionais tais como planilhas de dados e relatórios informativos operacionais, onde o especialista testa sua hipótese contra a base de dados. Ou seja, as informações contidas nos dados não estão caracterizadas explicitamente, uma vez que sendo dados operacionais não interessam quando estudados individualmente. Logo, não bastava armazená-los, era preciso transformá-los em informações.

Estas informações tornaram-se essenciais para as empresas, já que as bases de dados deixaram de ser apenas repositórios de informações, passando a ser tratadas como patrimônio das mesmas.

O dado é um elemento puro, quantificável sobre um determinado evento. A informação é o dado analisado e contextualizado. Envolve a interpretação de um conjunto de dados, ou seja, a informação é constituída por padrões, associações ou relações que todos aqueles dados acumulados podem proporcionar.

A informação pode gerar conhecimento que ajuda na análise de padrões históricos para se conseguir uma previsão dos fatos futuros (pelo menos no contexto das variáveis que estão sendo envolvidas na análise).

O processo capaz de descobrir conhecimento (informação) em bancos de dados chama-se *Knowledge Discovery in Databases* - KDD. Ainda segundo *FAYYAD et al.* [12], este processo foi proposto em 1989 para referir-se às etapas que produzem conhecimento a partir dos dados. Dentro deste processo a etapa de mineração de dados é a fase que transforma dados em informação.

Como o maior equipamento em sistemas de potência, o transformador de potência é vital para a operação dos sistemas e as técnicas para diagnóstico e detecção incipiente de falhas são valiosas para melhorar a manutenção. A análise de gás dissolvido no óleo do

transformador é uma ferramenta poderosa. Neste trabalho será utilizada esta análise, baseada na pesquisa de DUVAL [11], onde é proposto um método para identificação da falha baseado nos teores de formação dos gases etileno (C_2H_4), metano (CH_4), acetileno (C_2H_2), hidrogênio (H_2) e etano (C_2H_6).

1.1 Objetivos

O objetivo da mineração de dados é extrair informações, implicitamente contidas nos banco de dados.

O objetivo principal deste trabalho é estudar, compreender e utilizar ferramentas de mineração de dados, eficientes para extração do conhecimento implícito, em auxílio à tomada de decisões em áreas da engenharia elétrica.

Os objetivos específicos são compreender, analisar e comparar as técnicas de redes neurais e árvores de decisão, aplicadas a problemas de mineração de dados oriundos de testes de cromatografia de transformadores de potência.

1.2 Estado da Arte

Neste item é feita uma revisão bibliográfica, com um breve relato de outros trabalhos desenvolvidos com a utilização de diferentes técnicas em mineração de dados.

BALA *et al.* [6] publicaram um trabalho onde estudam uma forma de aprendizado híbrido, usando algoritmo genético e árvores de decisão para classificação. A idéia foi a integração do algoritmo AG GENESIS, com população constante e taxas de cruzamento e mutação respectivamente iguais a 0,6 e 0,001, com o algoritmo C4.5 para o procedimento de evolução. Os resultados experimentais foram apresentados para ilustrar a eficácia da pesquisa em problemas complexos. Foram estudadas duas bases de dados. Uma delas composta de dados para reconhecimento de imagens faciais, apresentando erro de 27,5%, e a outra de reconhecimento visual de satélite, apresentando erro de 6,97%. Os resultados

mostram bons desempenhos de classificação quando comparados com métodos clássicos para classificação que apresentaram erros de 38,4% e 18,5% respectivamente.

LU *et al.* [24] publicaram um trabalho onde abordaram a aplicação de redes neurais para classificação em mineração de dados, dando ênfase às regras de extração. A base de dados trabalhada era composta de características de pessoas que seriam classificadas em grupos, tais como: idade, salário e possuir casa própria. Neste trabalho foi proposta a rede neural MLP (rede de múltiplas camadas compostas por neurônios do tipo perceptron) com algoritmo de retropropagação para o aprendizado. Os resultados mostraram um erro menor utilizando estas redes quando comparados ao algoritmo C4.5 de árvore de decisão. Porém, a rede neural precisou de um tempo maior de aprendizado.

ALMEIDA e DUMONTIER [3] publicaram um trabalho no qual apresentam uma abordagem estruturada da exploração de redes neurais, utilizando a rede MLP, com algoritmo de aprendizado de retropropagação. O método foi utilizado para avaliação de riscos de inadimplência, avaliando 2412 empresas do setor de transporte de carga rodoviário francês. O desempenho foi comparado com o método da regressão logística (LOGIT). Foi concluído que o desempenho da rede neural implementada não foi significativamente superior ao desempenho do método estatístico, porém possui uma maior capacidade de generalização.

FAYYAD *et al.* [12] publicaram o trabalho "From Data Mining to Knowledge Discovery in Databases" no qual descrevem como são relacionadas a mineração de dados e o KDD em um banco de dados, como em seus campos relacionados – estatística e aprendizagem de máquina. Neste trabalho é conceituado que KDD é todo o processo de descoberta de conhecimento e a mineração de dados refere-se a apenas uma fase deste processo. No trabalho são relatadas técnicas específicas para mineração de dados tais como árvore de decisão, regressão não linear e modelos de aprendizagem relacional. É discutido que não existe um método mais eficiente que sirva para todas as aplicações. A escolha do método vai variar de acordo com o objetivo da mineração de dados.

ALMEIDA e SIQUEIRA em [4] fazem uma comparação entre regressão logística, com o algoritmo LOGIT e redes neurais, aplicando o algoritmo de retropropagação em uma rede

MLP. As técnicas foram aplicadas a uma base de dados balanceada de 54 bancos brasileiros para a avaliação do risco de insolvência. A técnica de rede neural não apresentou resultado muito superior ao obtido pela regressão logística, mas apresentou um fator diferencial que foi o de poder considerar a base de dados com campos vazios. A regressão logística necessita da base de dados com todos os campos não vazios.

ZHANG *et al.* [44] publicaram um trabalho propondo uma rede neural artificial para diagnóstico e detecção de falha em transformadores, considerando as concentrações de gases no óleo do transformador. Os dados são classificados de acordo com quatro diagnósticos. A rede neural utilizada foi a perceptron de múltiplas camadas (MLP) com o aprendizado feito pelo algoritmo de retropropagação. As simulações foram feitas variando os parâmetros de entrada, o número de camadas escondidas e o número de nós na saída. A validação foi realizada com a técnica de validação cruzada. Os autores chegaram à conclusão de que quanto mais complexa a relação mais dados de treinamento são necessários e que aumentando a quantidade destes dados a precisão do modelo pode ser melhorada.

WANG *et al.* [41] publicaram um trabalho para diagnóstico de falhas em transformadores. Foi proposta uma classificação dos estados dos transformadores baseado em três formas: em sistemas especialistas, em redes neurais e em redes neurais conjugada com sistemas especialistas, chamadas pelos autores de redes neurais especialistas. As simulações foram feitas com uma base de 210 dados. A rede neural utilizada foi a perceptron de múltiplas camadas (MLP) com o algoritmo de retropropagação para o treinamento. Os resultados do trabalho mostram que o sistema conjugado tem melhor performance quando comparado com os resultados da classificação feita por cada sistema separadamente.

BRAMEIER e BANZHAF [9] publicaram um trabalho onde apresentam uma comparação entre programação genética linear e a técnica de redes neurais, utilizando a rede MLP com o algoritmo de retropropagação resiliente para aprendizado, para mineração de dados médicos. O desempenho dos dois métodos foi compatível, sendo a programação genética linear considerada satisfatória na classificação e generalização dos dados.

BOSIGNOLI e INFANTOSI em [8] estabelecem uma classificação automática do estado de sono ativo neonatal utilizando uma base de dados oriunda de exames feitos em recém nascidos. A técnica utilizada para esta classificação foi a de redes neurais, com uma rede MLP aplicando o algoritmo de retropropagação para o aprendizado, utilizando a função de ativação tangente hiperbólica. A classificação resultou em 95% de classificação correta.

LEMOS [23] apresentou um trabalho no qual fez uma análise de crédito bancário com o uso de mineração de dados, utilizando técnicas de redes neurais e árvores de decisão. Em redes neurais foi aplicada uma rede MLP, com o algoritmo de aprendizado sendo o de retropropagação e em árvores de decisão foi utilizado o algoritmo J4.8. O objetivo do trabalho era auxiliar na tomada de decisão sobre conceder ou não crédito bancário a um novo cliente. A base de dados foi cedida por uma agência bancária composta de 339 clientes de micro e pequenas empresas, sendo 266 adimplentes e 73 inadimplentes. Os resultados foram considerados satisfatórios, apresentando um erro no conjunto de validação de 28,13% para árvores de decisão e 9,96% para redes neurais. Já no conjunto de treinamento os erros foram de 11,49% e 4,09%, respectivamente.

1.3 Motivação

Diante das diversas aplicações da classificação na mineração de dados foi proposta uma forma de utilizá-las para auxiliar em áreas da engenharia elétrica. A escolha do uso de mineração de dados para auxiliar na tomada de decisão, através da tarefa de classificação, utilizando as técnicas que envolvem redes neurais e árvores de decisão, deve-se a algumas vantagens que a mineração de dados nos proporciona, dentre elas as principais levam em consideração o fato de serem de fácil compreensão e das variáveis envolvidas poderem ser usadas na forma original como aparecem nas bases de dados, não necessitando de uma normalização. O fato dos modelos obtidos com a técnica de árvores de decisão serem de fácil compreensão possibilita às pessoas sem conhecimento estatístico de interpretar tais modelos.

A utilização das técnicas de redes neurais com a rede MLP e algoritmo de retropropagação resiliente, e de árvores de decisão, com o algoritmo J4.8, estão embasadas nos bons resultados que vêm apresentando em trabalhos anteriores.

A aplicação dos métodos para elaboração de classificadores de falhas baseadas em concentrações de gases no óleo dos transformadores teve uma motivação no fato do problema não possuir uma função matemática que descreva o comportamento da taxa de evolução destas concentrações em função das falhas. Assim é justificado o uso de dados históricos aplicados em métodos heurísticos como redes neurais e árvores de decisão. Além disso, os métodos de monitoramento em tempo real dos transformadores, que fornecem as concentrações dos gases, estão cada vez mais confiáveis.

1.4 Limitações do Trabalho

O trabalho tem algumas limitações inerentes à situação. As mais claras são as atividades de pré-processamento que exigem a participação de especialistas do domínio de aplicação das bases de dados. Estas atividades foram escolhidas, então, de forma a não precisarem deste requisito, ou seja, foram realizados os pré-processamentos que não dependiam do domínio de aplicação das bases de dados.

Os trabalhos âncoras para o desenvolvimento da pesquisa foram muitos, destacando-se os das referências BRAMEIER e BANZHAF [9], LEMOS [23] e LU *et al.* [24].

2 Descoberta de conhecimento em bases de dados

2.1 Introdução

Neste capítulo é estudado o processo de descoberta de conhecimento em bases de dados (KDD). O estudo é feito desde o conceito até cada uma de suas fases. Na fase de pré-processamento e limpeza dos dados são abordadas formas de serem realizadas tais tarefas, chamando atenção às conseqüências de serem executadas sem a presença de um especialista no domínio dos dados.

2.2 Descoberta de conhecimento em bases de dados

KDD é um processo de descoberta de conhecimento em bases de dados que tem como objetivo principal extrair conhecimento a partir de grandes bases de dados. Para isto ele envolve diversas áreas do conhecimento, tais como: estatística, matemática, bancos de dados, inteligência artificial, visualização de dados e reconhecimento de padrões. São utilizadas técnicas, em seus diversos algoritmos, oriundas dessas áreas.

O processo de KDD é um conjunto de atividades contínuas que compartilham o conhecimento descoberto a partir de bases de dados. Para iniciar um processo de KDD é preciso ter o entendimento do domínio da aplicação e dos objetivos finais a serem atingidos. Para tal tem-se três pessoas envolvidas no processo de KDD, que são elas:

- analista de Dados: responsável pelos algoritmos e ferramentas utilizadas no processo. Entende das técnicas envolvidas no processo de KDD, mas não necessariamente conhece o domínio ao qual os dados pertencem.
- especialista no Domínio: conhece o domínio ao qual os dados pertencem e onde são aplicados.

- usuário: quem vai utilizar o resultado do processo. Esta parte (pessoa ou empresa) deve sempre fazer parte da equipe envolvida em um projeto de KDD.

Segundo FAYYAD *et al.* em [12], esse conjunto é composto basicamente por 5 (cinco) etapas, relacionadas na Figura 2.1.

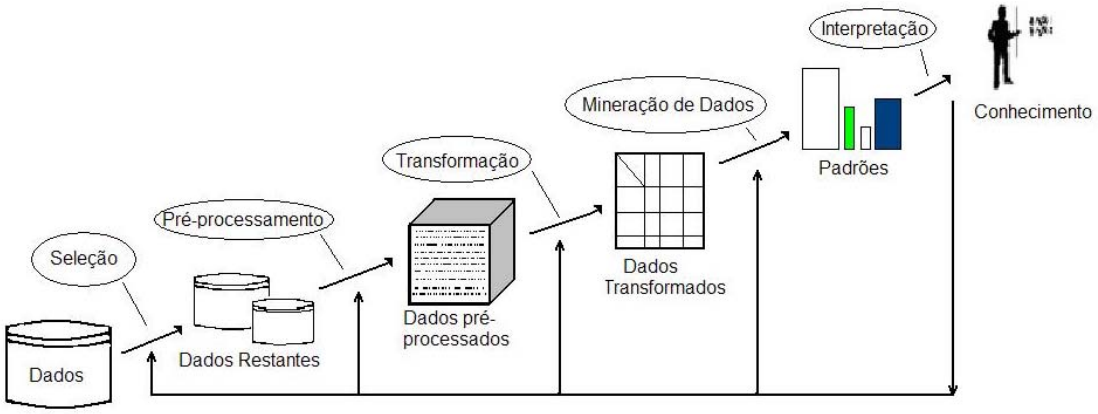


Figura 2.1: Fases de um processo de descoberta de conhecimento em bases de dados.
Fonte: FAYYAD *et al.* [12]

Seguindo a Figura 2.1, iniciando um processo de KDD, a primeira etapa é um agrupamento de forma organizada dos dados (seleção). A etapa da limpeza dos dados vem a seguir, através de um pré-processamento dos dados, visando adequá-los aos algoritmos que serão utilizados. Para facilitar o uso das técnicas de mineração de dados, os dados ainda podem passar por uma transformação que os armazena adequadamente em arquivos para serem lidos pelos algoritmos. É a partir deste momento que se chega à fase de mineração de dados especificamente, que começa com a escolha das ferramentas (algoritmos) a serem utilizadas. Essa escolha depende fundamentalmente do objetivo do processo de KDD: classificação, agrupamento, regras associativas, ou desvio. De acordo com o algoritmo utilizado será gerado um arquivo de descobertas (que pode ser um relatório ou um gráfico, por exemplo). Este arquivo deve ser interpretado, gerando as conclusões que fornecem o conhecimento da base de dados estudada.

2.3 Fases da descoberta de conhecimento em bases de dados (KDD)

2.3.1 Seleção dos dados

De acordo com ALMEIDA e DUMONTIER [3], a tarefa de seleção dos dados é crítica porque os dados podem não estar disponíveis em um formato apropriado para serem utilizados no processo de KDD. Ou, mesmo se disponíveis, os dados podem precisar ser rotulados com o auxílio de um especialista do domínio.

Um dos principais problemas em coletar dados é descobrir onde encontrá-los. A maioria dos sistemas de gerenciamento de dados que estão funcionando hoje são proprietários, mas existe uma tendência mais forte das empresas implementarem novos bancos de dados, que são direcionados para dar suporte às pessoas responsáveis por tomar decisões – chamados de repositórios de dados. O objetivo do repositório de dados é integrá-los, de diversos sistemas transacionais, da forma mais confiável possível, ajudando na seleção dos dados.

2.3.2 Pré-processamento dos dados e limpeza

De acordo com MANNILA [25], a fase de pré-processamento é a mais complexa, podendo tomar até 80% de todo o tempo do processo e precisa ser feita com especialistas que conhecem bem o domínio de aplicação dos dados, já que suas atividades são, por exemplo, a integração de dados heterogêneos e a eliminação de incompletude dos dados. Além disso, podem aparecer problemas que são específicos para cada aplicação e que, dessa forma, precisam ser resolvidos com soluções específicas.

A limpeza dos dados envolve uma verificação da consistência das informações, e o preenchimento ou a eliminação de valores nulos e redundantes. Nessa fase são identificados e removidos os dados duplicados e/ou corrompidos. Uma boa limpeza dos dados é

essencial, podendo inclusive diminuir o tempo de processamento, eliminando consultas desnecessárias à base de dados.

A limpeza dos dados é dependente do domínio da aplicação, tornando a participação do analista de dados essencial também nessa fase.

As atividades do pré-processamento e limpeza dos dados podem ser dividas em dois grupos. Um grupo é composto por atividades que só devem ser executadas por especialistas no domínio dos dados – atividades muito dependentes de conhecimento do domínio, e o outro grupo é composto pelas atividades que são independentes do domínio dos dados, podendo ser executadas por qualquer pessoa.

2.3.2.1 Atividades de Pré-Processamento muito dependentes de conhecimento do domínio

Estas atividades só são efetivamente realizadas com o uso de conhecimento específico do domínio. Até poderiam ser realizadas através de um método automático, desde que para criá-lo seja fornecido conhecimento específico do domínio. São exemplos de dificuldades encontradas no pré-processamento, que são fortemente dependentes de conhecimento do domínio:

2.3.2.1.1 Inconsistências

Este erro é muito freqüente quando um atributo assume diferentes valores, mas que representam na essência a mesma informação. Por exemplo, um atributo nome, que armazena nomes de instituições, assume os valores UFMG e Universidade Federal de Minas Gerais, que são vistos pelo programa como diferentes mas representam a mesma coisa.

2.3.2.1.2 Poluição

A poluição ou ruído são os dados distorcidos, que foram muitas vezes improvisados. Uma poluição muito freqüente acontece quando o sistema é desenvolvido para uma especificação e passa a ser usado para outra. Por exemplo, uma empresa de cartão de crédito que originalmente só cadastrava clientes pessoa física, possui em seu banco de dados um campo sexo. Neste campo deveria aparecer o F (feminino) ou M (Masculino), entretanto, alguns registros assumem o valor E para esse atributo, que corresponde a uma empresa.

2.3.2.1.3 Atributos duplicados e redundantes

Ocorre quando uma informação essencialmente idêntica é armazenada em diversos atributos. Um exemplo é possuir atributos em uma mesma Tabela tais como preço por unidade, quantidade comprada e preço total. O maior dano causado por este tipo de erro é uma leitura desnecessária de dados, aumentando o tempo de processamento.

2.3.2.2 Atividades de Pré-Processamento que não dependem de conhecimento de domínio

Estas atividades são aquelas que utilizam métodos que retiram dos próprios dados as informações necessárias para tratar o problema. Alguns exemplos:

2.3.2.2.1 Valores em branco

Um método muito utilizado para resolver este problema é a substituição dos valores desconhecidos pela média ou moda do atributo correspondente. Outra opção é usar um algoritmo de aprendizado para substituir o vazio por um valor predito, mas neste caso precisa da participação de um especialista no domínio.

2.3.2.2.2 Dados com classes desbalanceadas

Quando em uma base de dados uma classe aparece em maior quantidade que outra, esta base é dita desbalanceada. Em KUBAT e MATWIN [22] foi utilizado o método de seleção unilateral para balancear um conjunto de dados contendo informações colhidas de fotos de satélites. Uma forma mais direta é a replicação dos dados em menor número.

2.3.3 Transformação dos dados

Esta fase é realizada dependendo do algoritmo que será aplicado na mineração de dados, pois é o algoritmo que possui as limitações que precisam ser impostas à base de dados. Algumas das transformações: normalização de atributos quantitativos e transformação de atributos qualitativos em quantitativos.

De acordo com o trabalho de BATISTA [7], quando a base de dados for constituída de dados qualitativos, para aplicá-la em redes neurais é aconselhável criar um nó para cada atributo, sendo estes dados desmembrados em n atributos binários, para n valores diferentes na base qualitativa.

Em resumo essa fase converte os dados para a forma mais adequada à construção e interpretação do modelo.

2.3.4 Mineração de dados

A mineração de dados é a etapa mais importante do processo de KDD. Segundo POSSA *et al.* [30], o cérebro humano, comprovadamente, consegue fazer até 8 (oito) comparações ao mesmo tempo. A função da mineração de dados é justamente ampliar esta comparação para "infinito" e tornar isso visível ao olho humano. Aqui não será dada ênfase a esta fase, sendo tratada especificamente no próximo capítulo.

2.3.5 Avaliação e interpretação de resultados

Os resultados do processo de descoberta do conhecimento podem ser mostrados de diversas formas. Esta fase envolve todos os participantes que avaliam de forma criteriosa os resultados proporcionando uma interpretação para o modelo, de onde se extrai o conhecimento.

2.4 Conclusão

Pode-se concluir que a descoberta de conhecimento em bases de dados é um processo complexo, composto por diversas etapas. Deve ser seguido um critério para realização de cada uma destas etapas, levando em consideração a necessidade ou não da participação de um especialista no domínio da base de dados.

Se não for possível a participação dos especialistas, devem ser feitas as transformações que não dependem do conhecimento do domínio, para não correr o risco de ocorrer uma distorção da base de dados, modificando o relatório gerado pelo algoritmo de mineração de dados.

3 Mineração de Dados

3.1 Introdução

Neste capítulo são fornecidos conceitos da etapa de mineração de dados dentro do processo de descoberta de conhecimento em bases de dados. São relatadas as tarefas mais usadas em mineração de dados, citando aplicações e características.

3.2 Mineração de Dados

São muitas as definições de mineração de dados encontradas na literatura, algumas delas:

"Mineração de dados é uma ferramenta utilizada para descobrir novas correlações, padrões e tendências entre as informações de uma empresa, através da análise de grandes quantidades de dados armazenados em *Data Warehouse* usando técnicas de reconhecimento de padrões, estatística e matemática", NIMER e SPANDRI [28].

"Mineração de dados é um processo que encontra relações e modelos dentro de um grande volume de dados armazenados em um banco de dados", RODRIGUES [33].

"Mineração de dados é uma técnica para determinar padrões de comportamento, em grandes bases de dados, auxiliando na tomada de decisão", SILVA [37].

"Mineração de dados é um conjunto de técnicas que envolvem métodos matemáticos, algoritmos e heurísticas para descobrir padrões e regularidades em grandes conjuntos de dados", POSSA *et al.* [30].

"Mineração de dados é a extração de informações potencialmente úteis e previamente desconhecidas de grandes bancos de dados, serve para descobrir perfis de consumidores e outros comportamentos que não seriam identificados nem por especialistas", GUIZZO [13].

“Extração de conhecimento de base de dados (mineração de dados) é o processo de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis embutidos nos dados”, FAYYAD *et al.* [12].

Resumindo, pode-se concluir que a mineração de dados caracteriza-se pela existência de um algoritmo que diante da tarefa proposta será eficiente em extrair conhecimento implícito e útil de um banco de dados. Pode-se dizer que mineração de dados é a fase que transforma dados puros em informações úteis.

Na fase de mineração de dados, dentro do processo de KDD, necessita-se definir a técnica e o algoritmo a ser utilizado em função da tarefa proposta.

Uma vez escolhido o algoritmo a ser utilizado, deve-se implementá-lo e adaptá-lo ao problema proposto. Para finalizar essa etapa deve-se executar o algoritmo a fim de obter resultados que serão analisados na fase de interpretação e avaliação do resultado.

A mineração de dados difere de técnicas estatísticas porque ao invés de verificar padrões hipotéticos utiliza os próprios dados para descobrir tais padrões. De acordo com THEARLING *et al.* [39], aproximadamente 5% de todas as relações podem ser encontradas por esses métodos estatísticos. A mineração de dados pode descobrir outras relações anteriormente desconhecidas: os 95% restantes.

Diversas ferramentas distintas, como redes neurais, árvores de decisão, algoritmos genéticos, sistemas baseados em regras e programas estatísticos, tanto isoladamente, quanto em combinação, podem ser aplicadas ao problema.

No trabalho de KOHAVI *et al.* [21] é mostrado experimentalmente que não a escolha por um único bom algoritmo para todas as tarefas de mineração de dados não é trivial. Por isso, a escolha de vários algoritmos para realizar a tarefa desejada pode ser feita, levando à obtenção de diversos modelos.

3.3 Principais tarefas de mineração de dados

Como a mineração de dados vem sendo desenvolvida para os mais diferentes domínios, suas tarefas também vêm diversificando cada vez mais.

Essas tarefas podem extrair diferentes tipos de conhecimento, sendo necessário decidir, já no início do processo de mineração de dados, qual o tipo de conhecimento que o algoritmo deve extrair.

3.3.1 Classificação

A tarefa de classificação é uma função de aprendizado que mapeia dados de entrada, ou conjuntos de dados de entrada, em um número finito de classes. Nela, cada exemplo pertence a uma classe, entre um conjunto pré-definido de classes. O objetivo de um algoritmo de classificação é encontrar alguma correlação entre os atributos e uma classe, de modo que o processo de classificação possa usá-la para predizer a classe de um exemplo novo e desconhecido.

A classificação consiste em obter um modelo baseado em um conjunto de exemplos que descrevem uma função desconhecida. Esse modelo é utilizado posteriormente para fornecer o valor de atributos de novos exemplos.

São exemplos de tarefas de classificação: separar pedidos de créditos em baixo, médio e alto risco ou identificar a forma de tratamento mais adequado para um paciente, baseando-se em classes de pacientes que respondem bem a determinado tipo de tratamento médico.

3.3.2 Regressão

A tarefa de regressão é conceitualmente similar à de classificação. A principal diferença é que o atributo a ser predito é contínuo em vez de discreto.

Os métodos de regressão já são estudados pela comunidade estatística há bastante tempo. Porém, segundo INDURKHYA e WEISS [19], nas áreas de aprendizado de máquina e mineração de dados, a maioria das pesquisas é voltada para problemas de classificação, que são mais comumente encontrados na vida real do que problemas de regressão.

O objetivo da tarefa de regressão é encontrar uma relação entre um conjunto de atributos de entrada e um atributo-meta contínuo. Por exemplo, seja X = {x1, ..., xd} o atributo de entrada e y o atributo-meta, o objetivo é encontrar um mapeamento da seguinte forma $y = f(x1,x2,...,xd)$. A regressão também é conhecida por predição funcional, predição de valor real, função de aproximação, ou ainda, aprendizado de classes contínuas, segundo UYSAL e GUVENIR [40].

3.3.3 Regras de associação

Uma regra de associação caracteriza o quanto a presença de um conjunto de itens nos registros de uma base de dados implica na presença de algum outro conjunto distinto de itens nos mesmos registros, conforme AGRAWAL e SRIKANT [2]. Desse modo, o objetivo das regras de associação é encontrar tendências que possam ser usadas para entender e explorar padrões de comportamento dos dados. Por exemplo, observando os dados de vendas de um supermercado sabe-se que 80% dos clientes que compram o produto Q também adquirem, na mesma compra, o produto W. Nessa regra, 80% corresponde a sua confiabilidade.

As cadeias de varejo usam associação para planejar a disposição dos produtos nas prateleiras das lojas ou em um catálogo, de modo que os itens geralmente adquiridos na mesma compra sejam vistos próximos entre si e chamem a atenção do cliente.

3.3.4 Agrupamento

O agrupamento é um processo de partição de uma população heterogênea em vários

subgrupos mais homogêneos. No agrupamento, não existem classes pré-definidas, os registros são agrupados de acordo com a semelhança, o que a diferencia da tarefa de classificação.

Normalmente, a tarefa de agrupamento é realizada antes de alguma outra forma de mineração. Por exemplo, em uma aplicação de segmentação de mercado, pode-se primeiro dividir os clientes em grupos que tenham comportamento de compra similar ou que pertençam a uma região do país, para depois aplicar uma classificação.

3.3.5 Estimativa

A estimativa é usada para definir um valor para alguma variável contínua desconhecida como, por exemplo, altura de uma pessoa ou saldo de cartão de crédito. Ela trabalha com resultados contínuos. Pode ser usada para executar uma tarefa de classificação, convencionando-se que diferentes faixas de valores contínuos correspondem a diferentes classes. "Estimativa é aprender uma função que mapeia um item dado para uma variável de predição real estimada", FAYYAD *et al.* [12].

Como exemplos de tarefas de estimativa têm-se: estimar o número de filhos em uma família, estimar a renda total de uma família, estimar o valor em tempo de vida de um cliente, estimar a probabilidade de que um paciente morrerá baseando-se nos resultados de um conjunto de diagnósticos médicos ou prever a demanda de um consumidor para um novo produto, ainda segundo FAYYAD *et al.* [12].

3.3.6 Desvio

A tarefa de desvio tem por objetivo descobrir um conjunto de valores que não seguem padrões definidos. Para esta tarefa é necessário adotar padrões antecipadamente.

Pode-se usar esta tarefa para identificar fraudes baseadas em elementos que estão fora dos padrões ou são exceções às regras.

3.4 Conclusão

Com o estudo da fase de mineração de dados, vê-se que seu objetivo deve ser bem conhecido para uma escolha correta da tarefa a ser usada. Como o objetivo do trabalho é fazer um mapeamento de um conjunto de dados em um número finito de classes, para auxiliar em tomadas de decisão, a tarefa mais adequada é a classificação.

De acordo com esta escolha serão estudados métodos apropriados para atingir o objetivo. Foi visto que não existe apenas um método eficiente para cada aplicação. De acordo com os bons resultados que vêm apresentando, serão estudadas as técnicas em redes neurais e árvores de decisão para a classificação em mineração de dados. Os estudos das técnicas e seus algoritmos estão nos capítulos seguintes.

4 Redes Neurais

4.1 Introdução

Neste capítulo é feito um estudo das redes neurais. Após um histórico de como as redes neurais surgiram e suas evoluções é abordada a rede perceptron de múltiplas camadas (MLP). Também é discutida a forma de aprendizado utilizada nesta rede, sendo estudados profundamente os algoritmos de retropropagação e retropropagação resiliente. São apresentadas as equações utilizadas na propagação dos sinais, no sentido direto e reverso, em uma rede MLP, que utiliza uma função do tipo sigmóide para os neurônios da camada oculta e uma função linear para os neurônios da camada de saída. Por fim são apresentadas as equações para o algoritmo de retropropagação resiliente.

4.2 Redes neurais

Uma rede neural artificial (RNA) é uma técnica que constrói um modelo matemático, de um sistema neural biológico simplificado, com capacidade de aprendizado, generalização, associação e abstração. Assim como no cérebro humano, as redes neurais apresentam uma estrutura altamente paralelizada, composta por processadores simples (neurônios artificiais) conectados entre si.

De acordo com HAYKIN [15], uma propriedade importante das redes neurais é a sua habilidade para aprender a partir do ambiente na qual estão inseridas, ou ambiente de aprendizado, e melhorar seu desempenho através da aprendizagem. As RNA's tentam aprender por experiência, ou seja, diretamente dos dados, através de um processo de repetidas apresentações dos dados à rede.

Uma rede neural artificial é composta por várias unidades de processamento, cujo funcionamento é bastante simples. Essas unidades geralmente são conectadas por canais de

comunicação que estão associados a determinados pesos. Os pesos do neurônio artificial nada mais são do que um modelo para simular os dendritos, que são os responsáveis pelas sinapses no cérebro humano. São os pesos que alterando os seus valores representativos durante os estímulos, influenciam o resultado do sinal de saída, segundo TAFNER [38].

As entradas, simulando uma área de captação de estímulos, podem ser conectadas em muitos neurônios, resultando em uma série de saídas, onde cada neurônio representa uma saída. Essas conexões, em comparação com o sistema biológico, representam o contato dos dendritos com outros neurônios, formando assim as sinapses. A função da conexão em si é tornar o sinal de saída de um neurônio em um sinal de entrada de outro, ou ainda, orientar o sinal de saída para o mundo externo (mundo real). Ainda segundo TAFNER [38], as diferentes possibilidades de conexões entre as camadas de neurônios podem ter, em geral, n números de estruturas diferentes. Usualmente, trabalha-se com três camadas, que são classificadas em:

- Camada de entrada: onde os padrões são apresentados à rede;
- Camadas intermediárias ou ocultas: onde é feita a maior parte do processamento, através das conexões ponderadas. Estas podem ser consideradas como extratoras de características;
- Camada de saída: onde o resultado final é concluído e apresentado.

Por serem as configurações utilizadas neste trabalho, dar-se-á um destaque nas redes *perceptrons* de múltiplas camadas, nos algoritmos de aprendizado de *retropropagação* e retropropagação resiliente.

4.3 Alguns conceitos utilizados

4.3.1 Aprendizado supervisionado

Este método é chamado aprendizado supervisionado porque a entrada e saída desejadas para a rede são fornecidos por um supervisor (professor) externo. A figura do professor busca um equilíbrio de forma que seja criada uma ligação entre os pares de entrada e saída

fornecidos. Este método está representado na Figura 4.1. O professor avalia o comportamento da rede neural, indicando se está no caminho certo ou errado para direcionar o processo de treinamento. A cada padrão de entrada fornecido, o professor avalia a saída atual e a compara com a saída desejada. Dessa forma ajustam-se os pesos para minimizar os erros.

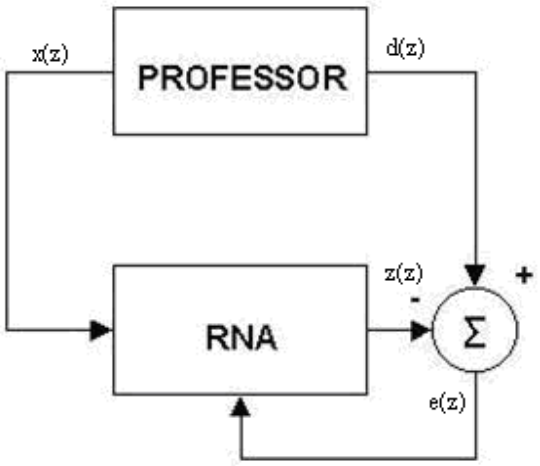


Figura 4.1: Representação do processo de aprendizado supervisionado.

Na fase de treinamento, o erro da rede na n-ésima iteração (i.e., na apresentação do n-ésimo exemplo de treinamento) é calculado tomando a diferença entre o valor desejado $d_k(n)$ (*i.e., valor de saída conhecido para o k-ésimo neurônio*) e o valor de saída da rede $z_k(n)$ *(i.e., valor de saída da rede para o k-ésimo neurônio)*, conforme (4.1):

$$e_k(n) = d_k(n) - z_k(n) \tag{4.1}$$

O valor instantâneo da energia do erro para a k-ésima saída é definida como sendo $e_k^2(n)/2$. Para se avaliar a energia instantânea total do erro, soma-se as contribuições de todas as saídas, conforme (4.2).

$$E(n) = \frac{1}{2}\sum_{k=1}^{Ns} e_k^2(n) \tag{4.2}$$

No caso particular da rede possuir apenas uma saída, a equação (4.2) se resume a (4.3).

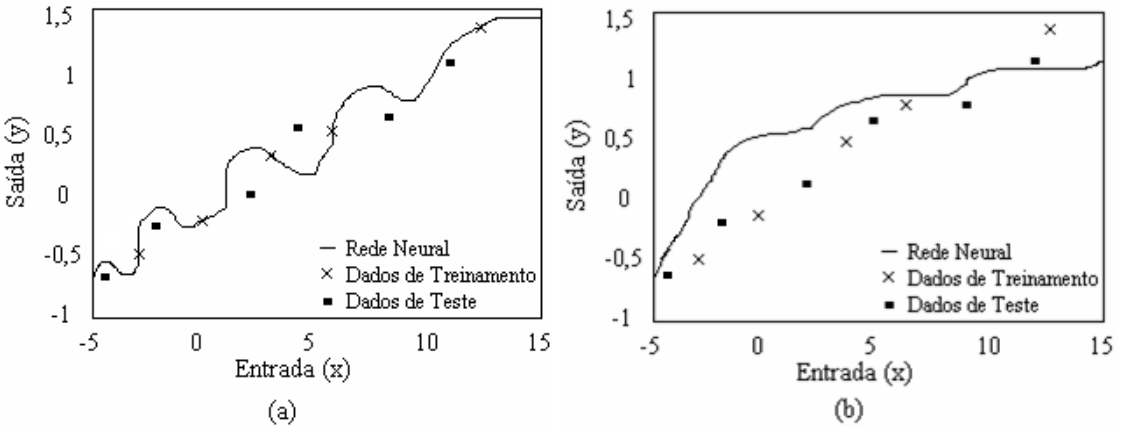
$$E(n) = \frac{1}{2}[e(n)]^2 = \frac{1}{2}[d(n) - z(n)]^2 \tag{4.3}$$

A média dos erros quadrados de todo o conjunto de treinamento Z, é utilizada para uma análise geral do treinamento. Ela é avaliada conforme (4.4).

$$E_{med} = \frac{1}{Z}\sum_{n=1}^{Z} E(n) \tag{4.4}$$

O treinamento é todo realizado com o objetivo de ajustar os pesos da rede, tal que a média dos erros quadrados seja minimizada.

De acordo com PASSOS [29], como é desejável que o erro quadrado seja minimizado, deve-se levar em consideração a influência de dois fenômenos: o sobre aprendizado e o sub-aprendizado. O primeiro é caracterizado quando a rede neural memoriza os dados de treinamento, mas apresenta uma generalização "pobre". Em outras palavras, o erro de treinamento E_{Tr} é pequeno, mas o erro de teste é grande ($E_{Te} >> E_{Tr}$). Razões possíveis para o sobre aprendizado incluem a presença de muitos neurônios ocultos ou a insuficiência dos dados de treinamento. Por outro lado, o sub aprendizado acontece quando a rede tem dificuldade de aprender os próprios dados de treinamento, ou seja, $E_{Tr} >> 0$. Ainda segundo PASSOS [29], isto acontece, geralmente, devido a um número insuficiente de neurônios, treinamento insuficiente ou pela estabilização do algoritmo de treino em um mínimo local da superfície de erro. A Figura 4.2 apresenta exemplos de sobre aprendizado, sub aprendizado e um modelo com boa aprendizagem, para dados sem a presença de ruídos.



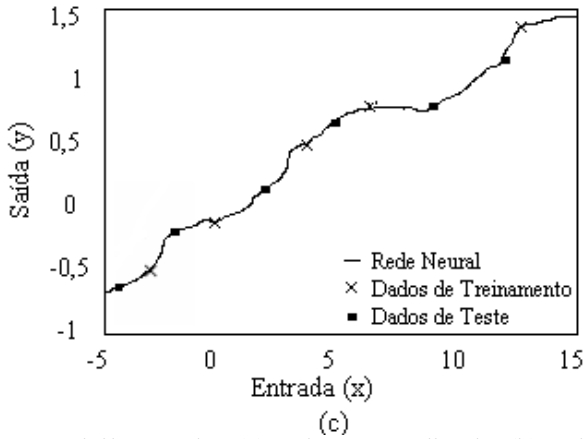


Figura 4.2: Um modelo neural ilustrando: (a) sobre aprendizado (b) sub aprendizado e (c) boa aprendizagem.
Fonte: PASSOS [29]

4.3.2 Método do gradiente

Seja uma função $f(x(n))$ contínua, numa dada iteração n. O método do gradiente é uma técnica numérica para a minimização de funções como esta através de suas derivadas. A direção de pesquisa em busca do mínimo da função será a direção negativa do gradiente. Ou seja:

$$x(n+1) = x(n) - \eta \nabla f(x(n)) \tag{4.5}$$

em que η é uma constante que determina a amplitude do passo na direção de descida da função, e ∇ é o operador matemático que representa o gradiente de uma função escalar multivariável. A convergência será acelerada se for utilizado um valor de η grande, porém isto dificultará o encontro do mínimo apropriado. Por outro lado ocorre uma lentidão considerável na convergência quando o valor de η for muito pequeno. O ideal é que para cada iteração se conheça o η ótimo.

4.4 Evolução das redes neurais

O primeiro modelo artificial de um neurônio biológico foi fruto do trabalho pioneiro de McCulloch e Pitts, quando foi publicado "*A Logical Calculus of the Ideas Immament in Nervous Activity*". Neste trabalho é apresentada uma discussão sofisticada de redes lógicas de neurônios artificiais (chamados de neurônios MCP). O trabalho de MCCULLOCH e PITTS [26] concentrou-se muito mais em descrever um modelo artificial de um neurônio e apresentar suas capacidades computacionais do que em apresentar técnicas de aprendizado.

O primeiro trabalho a ter ligação direta com o aprendizado de redes artificiais foi apresentado por Donald Hebb, em 1949. Hebb mostrou como a plasticidade da aprendizagem de redes neurais é conseguida através da variação dos pesos de entrada dos neurônios. Ele propôs uma teoria para explicar o aprendizado em neurônios biológicos baseada no reforço das ligações sinápticas entre neurônios excitados. Mais tarde, WIDROW e HOFF [42] sugeriram uma regra de aprendizado, conhecida como regra de delta, que ainda hoje é bastante utilizada. Esta, por sua vez, é baseada no método do gradiente descendente para minimização do erro na saída de um neurônio com resposta linear.

Em 1958, ROSENBLATT [35] demonstrou, com seu novo modelo, o *perceptron*, que se fossem acrescidas de sinapses ajustáveis, as redes com neurônios MCP poderiam ser treinadas para classificar certos tipos de padrões. Rosenblatt descreveu uma topologia de rede, estruturas de ligação entre os neurônios e propôs um algoritmo para treinar a rede para executar determinados tipos de funções.

E em 1986, Rumelhart, Hinton e Willians publicaram um trabalho onde foi desenvolvido o algoritmo de retropropagação para treinamento de redes MLP *(multi layer perceptron*), que são redes *perceptron* de múltiplas camadas.

4.5 Redes perceptron de múltiplas camadas – MLP

As redes *perceptron* de múltiplas camadas têm como unidade básica o *perceptron* descrito por MCCULLOCH e PITTS [26]. Segundo PASSOS [29], estas unidades são distribuídas em camadas onde cada uma está conectada a todas as unidades da camada anterior. Neste modelo, é calculado o produto interno das entradas aplicadas, x_i, com os pesos, w_{ji} e também é incorporada uma polarização, x_0, aplicada externamente. O efeito desta polarização é importante quando a soma ponderada dos neurônios da camada anterior for igual a zero. Ainda de acordo com PASSOS [29], a soma resultante, considerada como nível de atividade interna ou potencial de ativação é aplicada então a uma função de ativação, φ(.), que pode ser a saída final da rede, ou a entrada de outros *perceptrons* da camada seguinte. A Figura 4.4 apresenta a configuração do *perceptron*.

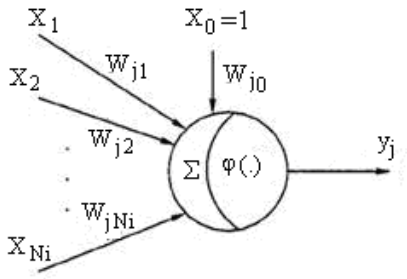


Figura 4.3: Modelo de neurônio artificial (*perceptron*) utilizado nas redes MLPs.

Em redes MLP´s as funções de ativação mais utilizadas são:

Função linear:

$$\varphi(net) = net \tag{4.6}$$

Função sigmóide:

$$\varphi(net) = \frac{1}{1+\exp(-net)} \tag{4.7}$$

Função tangente hiperbólica:

$$\varphi(net) = tanh(net) \tag{4.8}$$

HORNIK *et al.* [17] provaram o teorema da aproximação universal para as redes MLP. O teorema não demonstra como escolher o número de neurônios necessários para alcançar a aproximação de uma função, mas afirma que sempre existirá uma rede de três camadas capaz de aproximar qualquer função não linear e contínua. O que se sabe é que quanto mais complexo o problema (depende do grau de não linearidade e dimensionalidade) mais unidades ocultas serão necessárias. De acordo com HORNIK *et al.* [17], as falhas mais comuns em modelos neurais estão relacionadas a escolha do número de neurônios.

Uma solução para o problema do tamanho da rede é o teste por tentativa e erro, até conseguir um nível arbitrário de aproximação. Deve ser considerado, no entanto, que o desempenho de uma rede neural deve ser medido não em função do seu número de neurônios, e sim pela sua capacidade de mapeamento e generalização.

4.6 Algoritmos de retropropagação e retropropagação resiliente

O princípio do algoritmo de retropropagação é, utilizando-se o médodo do gradiente descendente, minimizar o erro das camadas intermediárias por meio de uma estimativa do efeito que estas causam no erro da camada de saída. Assim, o erro de saída da rede é calculado e este é retro-alimentado para as camadas intermediárias, possibilitando o ajuste dos pesos proporcionalmente aos valores das conexões entre camadas. A utilização do gradiente descendente requer o uso de função de ativação contínua e diferenciável.

Mas este algoritmo apresenta uma convergência lenta, causada pelo tamanho das derivadas parciais nos pesos. JACOBS [20] identificou duas causas fundamentais para este fato:

- Segundo ele, quando a superfície de erro (E) apresentar uma variação pequena (região *flat*) em relação a um dado peso, sua derivada terá uma magnitude pequena e consequentemente o ajuste será pequeno, requerendo muitas iterações para a convergência. Se a variação for elevada (região *sharp*), o gradiente e o ajuste também serão elevados acarretando uma passagem pelo mínimo da superfície de erro.

- Ainda segundo JACOBS [20], o vetor oposto ao vetor gradiente pode apontar para longe do mínimo da superfície de erro fazendo com que os ajustes ocorram em uma direção ruim.

Para uma boa convergência no modelo de retropropagação deve-se ter uma boa escolha da taxa de aprendizado η. Uma técnica aplicada para esta escolha é o uso do algoritmo de retropropagação resiliente, utilizado neste trabalho e que foi proposto por RIEDMILLER [32].

A idéia básica do algoritmo de retropropagação resiliente é eliminar a influência do valor das derivadas parciais na atualização dos pesos. Como conseqüência, é considerada somente a indicação do sinal da derivada parcial. A atualização dos pesos é determinada, de acordo com RIEDMILLER [32], exclusivamente por um valor de atualização $\Delta_{ji}(n)$, conforme (4.9) e (4.10).

$$\Delta w_{ji}(n) = \begin{cases} -\Delta_{ji}^{(n)}, & se \quad \dfrac{\partial E^{(n)}}{\partial w_{ji}} > 0 \\ +\Delta_{ji}^{(n)}, & se \quad \dfrac{\partial E^{(n)}}{\partial w_{ji}} < 0 \\ 0, & demais \quad casos \end{cases} \tag{4.9}$$

onde $\Delta_{ji}(n)$ é aumentado ou diminuído segundo o procedimento dado em (4.10).

$$\Delta ji(n) = \begin{cases} \eta^{+}\Delta_{ji}^{(n-1)}, & se \quad \dfrac{\partial E^{(n)}}{\partial w_{ji}} \dfrac{\partial E^{(n-1)}}{\partial w_{ji}} > 0 \\ \eta^{-}\Delta_{ji}^{(n-1)}, & se \quad \dfrac{\partial E^{(n)}}{\partial w_{ji}} \dfrac{\partial E^{(n-1)}}{\partial w_{ji}} < 0 \\ \Delta_{ji}^{(n-1)}, & demais \quad casos \end{cases} \tag{4.10}$$

Em (4.9) e (4.10), $E^{(n)}$ é a função erro quadrática (4.3), $\eta^{+} = 1{,}2$ e $\eta^{-} = 0{,}5$ são constantes escolhidas empiricamente.

Segundo RIEDMILLER [32], a regra de adaptação dos pesos trabalha do seguinte modo: cada vez que a derivada parcial do erro correspondente muda de sinal indica que a última atualização foi muito grande, ou seja, o algoritmo saltou o mínimo local. Neste caso o valor de adaptação é diminuído pelo fator η^-. Se o sinal da derivada permanece o mesmo, indica que o valor de adaptação deve ser aumentado, acelerando a convergência mesmo em regiões suaves da superfície de erro.

Uma vez que os valores de atualização para cada peso são adaptados, a atualização dos pesos segue uma regra muito simples:

• se a derivada trocar de sinal (erro de incremento), o peso é diminuído;

• se a derivada mantiver o sinal, o peso é aumentado.

4.7 Algoritmo de retropropagação aplicado à rede MLP

Segundo PASSOS [29], as equações de ajuste de uma rede neural são realizadas no sentido de minimizar o erro entre a resposta desejada e a saída da rede. Para o desenvolvimento desta teoria, considerou-se a estrutura neural da Figura 4.5, a qual possui um neurônio na camada de saída e uma camada oculta. Além disto, tem-se a função de ativação sigmóide para a camada oculta e a linear para a camada de saída. Esta estrutura será base para dedução de algumas fórmulas utilizadas no algoritmo de retropropagação aplicado à rede MLP e para justificar as equações propostas por RIEDMILLER em seu trabalho [32].

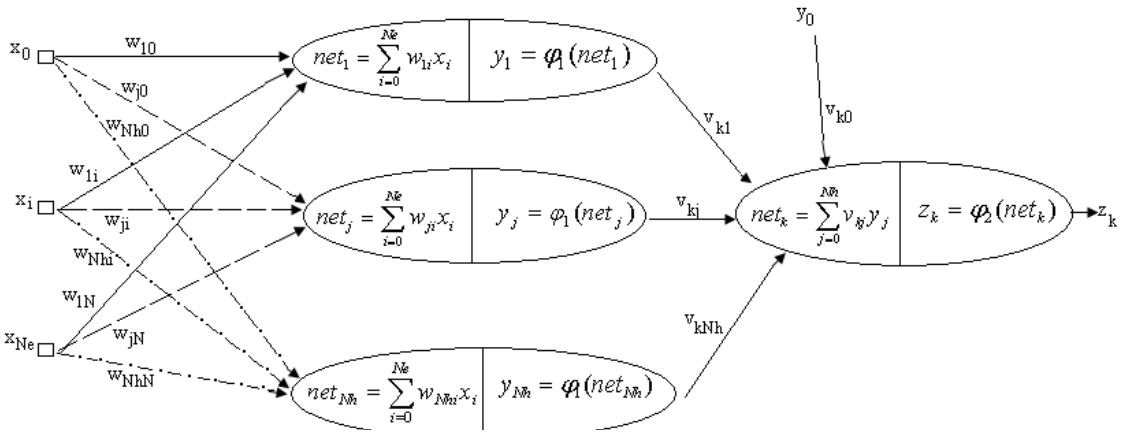


Figura 4.4: Configuração de uma rede MLP com uma camada de neurônios ocultos e um neurônio na camada de saída.
Fonte: PASSOS [29]

Para um dado conjunto de amostras para treinamento, em uma dada iteração n, tem-se que:

$$y_j(n) = \varphi_1(net_j(n)) \tag{4.11}$$

e

$$z_k(n) = \varphi_2(net_k(n)) \tag{4.12}$$

onde:

$$net_j(n) = \sum_{i=0}^{N_e} w_{ji}(n)x_i(n) \tag{4.13}$$

$$net_k(n) = \sum_{j=0}^{Nh} v_{kj}(n)y_j(n) \tag{4.14}$$

em que N_e representa o número de entradas da rede, Nh o número de neurônios ocultos; w_{ji} refere-se aos pesos da camada oculta e v_{kj} os pesos da camada de saída.

Para o desenvolvimento que se segue, considera-se para a função de ativação da camada oculta a função sigmóide dada em (4.7) e para a camada de saída a função linear (4.6), respectivamente conforme (4.15) e (4.16):

$$\varphi_1(net_j(n)) = \frac{1}{1+\exp(-net_j(n))} \tag{4.15}$$

$$\varphi_2(net_k(n)) = net_k(n) \tag{4.16}$$

4.7.1 Processamento no sentido direto do algoritmo de retropropagação

Considerando (4.15) e (4.16), pode-se escrever para um dado conjunto de amostras para treinamento, em uma dada iteração n e usando k=1:

$$z_k(n) = \varphi_2(net_k(n)) = net_k(n) = \sum_{j=0}^{Nh} v_{kj}(n) y_j(n), \tag{4.17}$$

$$y_j(n) = \varphi_1(net_j(n)) = \frac{1}{1+exp(-net_j(n))}, \quad \text{j = 1,2,...Nh} \tag{4.18}$$

$$net_j(n) = \sum_{i=0}^{Ne} w_{ji}(n) x_i(n), \qquad \text{j = 1,2,...Nh} \tag{4.19}$$

4.7.2 Processamento no sentido inverso do algoritmo de retropropagação

Em uma rede MLP, as variáveis a serem otimizadas são os pesos w_{ji}, da camada oculta, e v_{kj}, da camada de saída. O processo é iterativo, e quando realizado em cada apresentação de uma amostra para treinamento, os novos pesos são obtidos através da seguinte iteração:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{4.20}$$

$$v_{kj}(n+1) = v_{kj}(n) + \Delta v_{kj}(n) \tag{4.21}$$

em que $\Delta w_{ji}(n)$ e $\Delta v_{kj}(n)$ são as correções.

Para calcular expressões para as atualizações dos pesos, considere o gradiente do erro quadrado (4.3) em relação aos pesos:

$$\nabla E(n)\Big|_w = \begin{Bmatrix} \dfrac{\partial E(n)}{\partial w_{j0}(n)} \\ \dfrac{\partial E(n)}{\partial w_{j2}(n)} \\ \vdots \\ \dfrac{\partial E(n)}{\partial w_{jNe}(n)} \end{Bmatrix} = -e(n) \begin{Bmatrix} \dfrac{\partial z(n)}{\partial w_{jo}} \\ \vdots \\ \dfrac{\partial z(n)}{\partial w_{ji}} \\ \vdots \\ \dfrac{\partial z(n)}{\partial w_{jNe}} \end{Bmatrix}, \qquad \forall j = 1,2,...,Nh \tag{4.22}$$

Considerando o i-ésimo componente de (4.22) e utilizando o método do gradiente para minimização do erro, obtém-se para a atualização dos pesos da camada escondida a seguinte expressão:

$$w_{ji}(n+1) = w_{ji}(n) + \eta \frac{\partial z_k(n)}{\partial w_{ji}(n)} e(n) \tag{4.23}$$

Como $z_k(n) = \varphi_2(net_k(n)) = net_k(n)$, tem-se que:

$$\frac{\partial z_k(n)}{\partial w_{ji}(n)} = \frac{\partial \sum_{j=0}^{Nh} v_{kj}(n) y_j(n)}{\partial w_{ji}(n)} = \frac{\partial \sum_{j=0}^{Nh} v_{kj}(n) \varphi_1(net_j(n))}{\partial w_{ji}(n)} \tag{4.24}$$

ou

$$\frac{\partial z_k(n)}{\partial w_{ji}(n)} = v_{kj}(n) \frac{\partial \varphi_1(net_j(n))}{\partial net_j(n)} \frac{\partial net_j(n)}{\partial w_{ji}(n)} = v_{kj}(n) \frac{\partial \varphi_1(net_j(n))}{\partial net_j(n)} \frac{\partial \sum_{i=0}^{Ne} w_{ji}(n) x_i(n)}{\partial w_{ji}(n)} \tag{4.25}$$

ou

$$\frac{\partial z_k(n)}{\partial w_{ji}(n)} = v_{kj}(n)\frac{\partial \varphi_1(net_j(n))}{\partial net_j(n)} x_i(n) \quad . \tag{4.26}$$

Mas,

$$\frac{\partial \varphi_1(net_j(n))}{\partial net_j(n)} = \frac{\partial}{\partial net_j(n)}\left[\frac{1}{1+exp(-net_j(n))}\right] = \frac{net_j(n)exp(-net_j(n))}{\left(1+exp(-net_j(n))\right)^2} = y_j(n)\left[1-y_j(n)\right]$$

ou simplesmente,

$$\frac{\partial \varphi_1(net_j(n))}{\partial net_j(n)} = y_j(n)\left[1-y_j(n)\right] \tag{4.27}$$

Substituindo (4.27) em (4.26), tem-se que

$$\frac{\partial z_k(n)}{\partial w_{ji}(n)} = v_{kj}(n)y_j(n)[1-y_j(n)]x_i(n)\,. \tag{4.28}$$

Portanto, a expressão para a correção dos pesos da camada escondida, após substituição desta última expressão em (4.23), torna-se

$$w_{ji}(n+1) = w_{ji}(n) + \eta e(n)v_{kj}(n)y_j(n)[1-y_j(n)]x_i(n)\,. \tag{4.29}$$

De modo semelhante aos pesos da camada escondida (Eq. (4.23)), se se utiliza o método do gradiente para atualizar os pesos da camada de saída, tem-se (4.30).

$$v_{kj}(n+1) = v_{kj}(n) + \eta\frac{\partial z_k(n)}{\partial v_{kj}(n)}e(n) \tag{4.30}$$

Desenvolvendo (4.23), tem-se:

$$v_{kj}(n+1) = v_{kj}(n) + \eta.e(n)\frac{\partial z_k(n)}{\partial v_{kj}(n)} = v_{kj}(n) + \eta.e(n)\frac{\partial \sum_{j=0}^{Nh} v_{kj}(n)y_j(n)}{\partial v_{kj}(n)} \quad .$$

Logo, a expressão para a correção dos pesos da camada de saída se resume a:

$$v_{kj}(n+1) = v_{kj}(n) + \eta.e(n).y_j(n) \quad . \tag{4.31}$$

As expressões (4.29) e (4.31) são utilizadas para a atualização na iteração (n+1) dos valores $w_{ji}(n+1)$ e $v_{kj}(n+1)$, uma vez que se conheça o tamanho do passo η.

4.8 Algoritmo de retropropagação resiliente aplicado à rede MLP

Foi visto que o algoritmo de retropropagação resiliente é uma modificação do algoritmo de retropropagação, na tentativa de se acelerar a convergência, uma vez que o tamanho ótimo do passo η^* na direção de busca não é fácil de se estimar, e é problema dependente. Para isso a taxa de aprendizado é adaptativa, individual para cada peso, e a influência indesejada da magnitude das derivadas é ignorada. O procedimento adaptativo, proposto por RIEDMILLER [32], foi apresentado nas equações (4.9) e (4.10), e por questão de melhor compreensão do texto é reapresentado a seguir, onde a correção nos pesos da camada escondida é dado por

$$\Delta w_{ji}(n) = \begin{cases} -\Delta_{ji}^{(n)}, & se \quad \dfrac{\partial E^{(n)}}{\partial w_{ji}} > 0 \\ +\Delta_{ji}^{(n)}, & se \quad \dfrac{\partial E^{(n)}}{\partial w_{ji}} < 0 \\ 0, & demais \quad casos \end{cases}$$

e $\Delta_{ji}(n)$ é aumentado ou diminuído segundo o seguinte procedimento:

$$\Delta ji(n) = \begin{cases} \eta^{+}\Delta_{ji}^{(n-1)}, & se \quad \dfrac{\partial E^{(n)}}{\partial w_{ji}} \dfrac{\partial E^{(n-1)}}{\partial w_{ji}} > 0 \\ \eta^{-}\Delta_{ji}^{(n-1)}, & se \quad \dfrac{\partial E^{(n)}}{\partial w_{ji}} \dfrac{\partial E^{(n-1)}}{\partial w_{ji}} < 0 . \\ \Delta_{ji}^{(n-1)}, & demais \quad casos \end{cases}$$

A demonstração de que o procedimento acima tem base matemática é desenvolvida a seguir, fazendo-se a estimativa do valor ótimo do passo (busca unidirecional), empregando

o método do gradiente. Assim, sejam as correções nos valores dos pesos das camadas escondida de saída, dados por (4.25) e (4.26), respectivamente.

$$w_{ji}(n+1) = w_{ji}(n) - \eta^*_{w_{ji}} \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{4.32}$$

$$v_{kj}(n+1) = v_{kj}(n) - \eta^*_{v_{kj}} \frac{\partial E(n)}{\partial v_{kj}(n)} \tag{4.33}$$

onde o * no tamanho do passo (η*) designa o passo ótimo. Como não se conhece este passo ótimo, é preciso determinar um procedimento iterativo para determiná-lo. Assim, considere uma iteração p genérica, independente da iteração n. Na iteração p+1, pode-se estimar as seguintes correções:

$$\eta_{w_{ji}}(p+1) = \eta_{w_{ji}}(p) - \delta \frac{\partial E(p)}{\partial \eta_{w_{ji}}(p)} \tag{4.34}$$

$$\eta_{v_{kj}}(p+1) = \eta_{v_{kj}}(p) - \delta \frac{\partial E(p)}{\partial \eta_{v_{kj}}(p)} \tag{4.35}$$

Onde η_{Wji} e η_{Vkj} estão indicando as taxas de aprendizado da camada oculta e de saída, respectivamente, e δ é a taxa de aprendizado do método gradiente.

Desenvolvendo a derivada em (4.35), tem-se:

$$\frac{\partial E(p)}{\partial \eta_{v_{kj}}(p)} = \frac{\partial E(p)}{\partial z_k(p)} \cdot \frac{\partial z_k(p)}{\partial net_k(p)} \cdot \frac{\partial net_k(p)}{\partial \eta_{vkj}(p)} \tag{4.36}$$

$$\frac{\partial E(p)}{\partial z_k(p)} = \frac{\partial}{\partial z_k(p)} \left[\frac{(d_k(p) - z_k(p))^2}{2} \right] = -e(p) \tag{4.37}$$

$$\frac{\partial z_k(p)}{\partial net_k(p)} = \frac{\partial \varphi_2(net_k(p))}{\partial net_k(p)} = 1 \tag{4.38}$$

$$\frac{\partial net_k(p)}{\partial \eta_{v_{kj}}(p)} = \frac{\partial}{\partial \eta_{v_{kj}}(p)} \left(\sum_{j=0}^{Nh} v_{kj}(p) y_j(p) \right) = \sum_{j=0}^{Nh} \left[\frac{\partial v_{kj}(p)}{\partial \eta_{v_{kj}}(p)} \right] . y_j(p) = -\frac{\partial E(p-1)}{\partial v_{kj}(p-1)} . y_j(p) \tag{4.39}$$

Substituindo (4.37), (4.38) e (4.39) em (4.36), tem-se:

$$\frac{\partial E(p)}{\partial \eta_{v_{kj}}(p)} = e(p)\frac{\partial E(p-1)}{\partial v_{kj}(p-1)} \cdot y_j(p) \tag{4.40}$$

Substituindo (4.40) em (4.35), tem-se:

$$\eta_{v_{kj}}(p+1) = \eta_{v_{kj}}(p) - \delta e(p)\frac{\partial E(p-1)}{\partial v_{kj}(p-1)}.y_j(p) \tag{4.41}$$

Por outro lado,

$$\frac{\partial E(p)}{\partial v_{kj}(p)} = -e(p).\frac{\partial(\varphi_2(net_k(p))}{\partial v_{kj}(p)} = -e(p)\frac{\partial\left[\sum_{j=0}^{Nh} v_{kj}(p)y_j(p)\right]}{\partial v_{kj}(p)} = -e(p)y_j(p) \tag{4.42}$$

De (4.42), tem-se que

$$y_j(p) = -\frac{1}{e(p)}\frac{\partial E(p)}{\partial v_{kj}(p)} \tag{4.43}$$

Substituindo (4.43) em (4.42) tem-se a correção do passo para a atualização do peso da camada de saída, dado por (4.44).

$$\eta_{v_{kj}}(p+1) = \eta_{v_{kj}}(p) + \delta.\frac{\partial E(p-1)}{\partial v_{kj}(p-1)}.\frac{\partial E(p)}{\partial v_{kj}(p)} \tag{4.44}$$

De forma análoga, para a camada oculta pode-se desenvolver a expressão de correção desejada. Lembre-se que a correção do passo para a atualização do peso da camada escondida é conforme (4.34), repetida aqui para facilitar a compreensão:

$$\eta_{w_{ji}}(p+1) = \eta_{w_{ji}}(p) - \delta\frac{\partial E(p)}{\partial \eta_{w_{ji}}(p)} \tag{4.45}$$

$$\frac{\partial E(p)}{\partial \eta_{w_{ji}}(p)} = \frac{\partial E(p)}{\partial y_j(p)}\frac{\partial y_j(p)}{\partial net_j(p)}\frac{\partial net_j(p)}{\partial \eta_{w_{ji}}(p)} \tag{4.46}$$

$$\frac{\partial E(p)}{\partial y_j(p)} = \frac{\partial}{\partial y_j(p)}\left[\frac{1}{2}\left(e(p)\right)^2\right] = -e(p)\frac{\partial z_j(p)}{\partial y_j(p)} = -e(p)v_{kj}(p) \tag{4.47}$$

$$\frac{\partial y_j(p)}{\partial net_j(p)} = \frac{\partial \varphi_1(net_j(p))}{\partial net_j(p)} \tag{4.48}$$

$$\frac{\partial net_j(p)}{\partial \eta_{w_{ji}}(p)} = \frac{\partial}{\partial \eta_{w_{ji}}(p)}\left(\sum_{j=0}^{Ne} w_{ji}(p)x_i(p)\right) = \frac{\partial}{\partial \eta_{w_{ji}}(p)}\left[w_{ji}(p)x_i(p)\right] \tag{4.49}$$

$$\frac{\partial net_j(p)}{\partial \eta_{w_{ji}}(p)} = x_i(p).\frac{\partial}{\partial \eta_{w_{ji}}(p)}\left[w_{ji}(p-1) - \eta_{w_{ji}}\frac{\partial E(p-1)}{\partial w_{ji}(p-1)}\right] = -x_i(p)\frac{\partial E(p-1)}{\partial w_{ji}(p-1)} \tag{4.50}$$

Substituindo (4.47), (4.48) e (4.50) em (4.46), tem-se:

$$\frac{\partial E(p)}{\partial \eta_{w_{ji}}(p)} = e(p)v_{kj}(p)\frac{\partial \varphi_1(net_j(p))}{\partial net_j(p)}\frac{\partial E(p-1)}{\partial w_{ji}(p-1)}x_i(p) \tag{4.51}$$

A expessão (4.51) pode ser simplificada se observa-se que

$$\frac{\partial E(p)}{\partial w_{ji}(p)} = \frac{\partial}{\partial w_{ji}(p)}\left[\frac{(d(p)-z(p))^2}{2}\right] = -e(p)\frac{\partial z_k(p)}{\partial w_{ji}(p)} = -e(p)\frac{\partial}{\partial w_{ji}(p)}\left[\sum_{j=0}^{Nh} v_{kj}(p)y_j(p)\right]$$

$$= -e(p)\frac{\partial}{\partial w_{ji}(p)}\left[\sum_{j=0}^{Nh} v_{kj}(p)\varphi_1(net_j(p))\right] = -e(p).v_{kj}(p).\frac{\partial \varphi_1(net_j(p))}{\partial net_j(p)}.\frac{\partial net_j(p)}{\partial w_{ji}(p)}$$

$$= -e(p).v_{kj}(p).\frac{\partial \varphi_1(net_j(p))}{\partial net_j(p)}.\frac{\partial \sum_{i=0}^{Ne} w_{ji}(p).x_i(p)}{\partial w_{ji}(p)} = -e(p).v_{kj}(p).\frac{\partial \varphi_1(net_j(p))}{\partial net_j(p)}.x_i(p) \tag{4.52}$$

Logo, (4.51) pode ser reescrita como

$$\frac{\partial E(p)}{\partial \eta_{w_{ji}}(p)} = -\frac{\partial E(p-1)}{\partial w_{ji}(p-1)}\frac{\partial E(p)}{\partial w_{ji}(p)} \tag{4.53}$$

Substituindo (4.53) em (4.45), finalmente temse a expressão de atualização desejada:

$$\eta_{w_{ji}}(p+1) = \eta_{w_{ji}}(p) + \delta \frac{\partial E(p-1)}{\partial w_{ji}(p-1)} \frac{\partial E(p)}{\partial w_{ji}(p)} \tag{4.54}$$

As equações (4.44) e (4.54) fornecem as expressões para a atualização da taxa de aprendizado da camada de saída e da camada escondida. Elas não são usadas diretamente nos ajustes dos pesos. Foram deduzidas para demonstrar a utilização das mesmas nas equações propostas por RIEDMILLER [32] em (4.9) e (4.10).

4.9 Conclusão

No capítulo é visto que a utilização da técnica de redes neurais é bastante interessante considerando suas características de versatilidade, estabilidade e boa capacidade de aprendizado.

Agora, a adaptação de qual configuração deve ser usada é uma tarefa peculiar. Não existe uma definição de melhor configuração. Esta deve ser obtida por experiência e de acordo com os dados a serem tratados.

Também foi concluída a vantagem do algoritmo de retropropagação resiliente em relação ao de retropropagação, o qual melhora a convergência . Neste método, apenas o sinal do gradiente é levado em conta no momento do ajuste dos parâmetros livres.

5 Árvores de Decisão

5.1 Introdução

As árvores de decisão são representações simples do conhecimento e um meio eficiente de construir classificadores que predizem classes baseadas nos valores de atributos de um conjunto de dados.

Neste capítulo é apresentado um método de classificação utilizando árvores de decisão. É dada ênfase ao algoritmo J4.8, que deriva dos algoritmos ID3 e C4.5. São apresentados os conceitos e fórmulas para cálculos de algumas medidas utilizadas nestes algoritmos. Para finalizar é construída uma árvore de decisão, utilizando os conceitos de entropia e ganho de informação, seguindo os conceitos do algoritmo J4.8.

5.2 Árvores de Decisão

Uma árvore de decisão tem a função de particionar recursivamente um conjunto de treinamento, até que cada subconjunto obtido contenha casos de uma única classe. Para atingir esta meta, o algoritmo escolhido para a árvore de decisão examina e compara a distribuição de classes durante a construção da árvore. Segundo QUINLAN [31], os resultados obtidos, após a construção de uma árvore de decisão, são dados organizados de maneira compacta, com a árvore podendo ser utilizada para classificar novos casos.

As árvores de decisão são construídas baseadas no modelo *Top-Down*, ou seja, do nó raiz em direção às folhas. Na verdade, embora contendo diferenças importantes na forma de efetuar os passos, qualquer algoritmo desta categoria utiliza da técnica de dividir para conquistar. De uma forma geral, esta filosofia baseia-se na sucessiva divisão do problema em vários subproblemas de menores dimensões, até que uma solução para cada um dos problemas mais simples seja encontrada. Fundamentados neste princípio, os classificadores

baseados em árvores de decisão procuram encontrar formas de dividir sucessivamente o universo em vários subconjuntos, até que cada um deles contemple apenas uma classe ou até que uma das classes demonstre uma clara maioria, não justificando posteriores divisões.

O aprendizado de uma árvore de decisão é do tipo supervisionado. O método faz aproximações de funções nas quais a função aprendida é representada por uma árvore de decisão.

A Figura 5.1 a seguir representa uma árvore de decisão hipotética. Em uma árvore de decisão são usados os seguintes conceitos:

- Nó: são todos os itens que aparecem na árvore;
- Folhas: são nós que não têm filhos, os últimos itens da árvore;
- Filhos: são os itens logo abaixo da raiz
- Raiz: é o item topo da árvore.

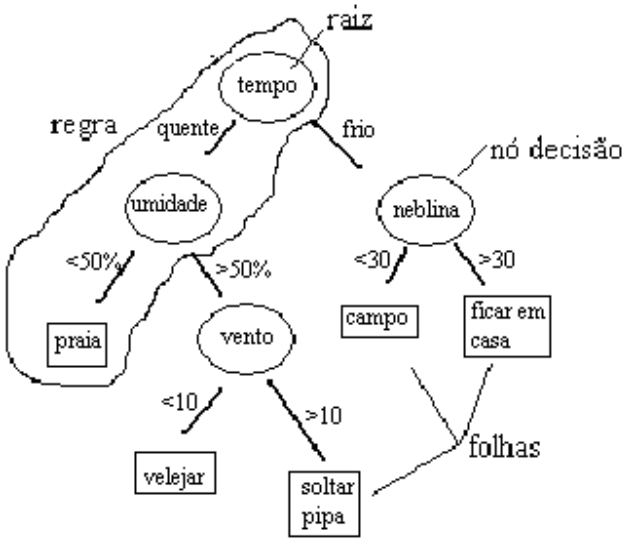


Figura 5.1: Exemplo de uma árvore de decisão.

Em um conjunto contendo N amostras para treinamento, cada amostra n, com n = 1,2,...N, é caracterizada por dois tipos de atributos: o atributo classe, que indica a classe a qual a

amostra n pertence; e os atributos preditivos, cujos valores serão analisados para que seja descoberto o modo como eles se relacionam com o atributo classe.

Assim, em uma árvore de decisão, tem-se:

- cada nó interno é um teste em um atributo preditivo;.
- uma ramificação partindo de um nó interno representa um resultado para o teste (por exemplo, tempo = “Quente”);
- uma folha da árvore representa um rótulo de classe (por exemplo, praia ou velejar);
- em cada nó da árvore, um atributo deve ser escolhido para dividir as observações do conjunto de treinamento em classes, na medida do possível;
- uma nova observação é classificada seguindo um caminho na árvore, da raiz até a folha;

Analisando a estrutura da árvore de decisão da Figura 5.1, é fácil perceber que é possível derivar regras do tipo “se-então” para melhoria da compreensão e interpretação dos resultados. As regras são escritas considerando o trajeto do nó raiz até uma folha da árvore. Os antecedentes da regra são formados pelos atributos preditivos que aparecem ao longo do caminho percorrido, testando os valores que os definem e os conseqüentes são formados pelo atributo classe.

Considerando ainda a árvore da Figura 5.1, podem ser derivadas as regras, do tipo “se-então”:

- se tempo é quente e umidade menor que 50% então praia.
- se tempo é quente e umidade maior que 50% e vento maior que 10m/s então soltar pipa.

Estes dois métodos – as árvores de decisão e as regras - são geralmente utilizados em conjunto. As regras geradas podem ser facilmente manipuladas. Uma regra pode ser compreendida sem que haja a necessidade de se referenciar outras regras. Quando uma aplicação gera uma árvore muito extensa, é mais fácil trabalhar com as regras.

Uma questão chave para a construção de uma árvore de decisão consiste na estratégia para a escolha dos atributos que estarão mais próximos da raiz da árvore (ou seja, os atributos que são inicialmente avaliados para determinar a classe a qual uma observação pertence).

5.3 Conceitos utilizados em árvores de decisão

5.3.1 Entropia

O conceito de entropia é uma medida de informação calculada pelas probabilidades de ocorrência de eventos individuais ou combinados. Pode-se dizer que a entropia é dada como uma medida da impureza em um conjunto arbitrário de amostras de treinamento. Pode ser considerada a medida da quantidade de desordem de um conjunto de amostras.

Dado um atributo classe A, de um conjunto de amostras S, em que A pode assumir v_i valores de classes diferentes, então a entropia de A relativa a esta classificação é definida como:

$$Entropia(A) = -\sum_{i=1}^{m} p_i \log_2 p_i \qquad (5.1)$$

onde m é o número total de classes e $p_i = p(A=v_i)$ é a probabilidade do atributo classe A ser igual à classe cujo índice é *i* (*i.e., é a proporção do número de amostras com valor* v_i *em relação ao número total de amostras de S*).

Quanto maior a entropia de um atributo, mais uniforme é a distribuição dos seus valores. A entropia será igual a zero quando ocorrer apenas uma classe no conjunto de dados, e igual a

um se o número de amostras de cada classe for igual. Se a entropia é pequena, próxima de zero, significa que as classes são pouco uniformes.

5.3.2 Ganho de Informação

O ganho de informação é definido como uma soma das entropias individuais menos a entropia conjunta, sendo uma medida de correlação entre duas variáveis. É uma propriedade estatística que mede como um determinado atributo separa as amostras de treinamento de acordo com sua classificação. Ele mede a eficácia de um atributo em classificar os dados de treinamento.

Um dos objetivos da construção de árvores de decisão é diminuir o valor da entropia. A medida do ganho de informação representa a redução esperada na entropia de um atributo preditivo, considerando que um atributo classe já tenha sido determinado. Ou seja, o valor do ganho de informação fornece uma redução esperada na entropia causada pela partição das amostras de acordo com este atributo classe conhecido previamente. No processo de construção da árvore de decisão, o atributo preditivo que possuir o maior ganho de informação deve ser colocado como raiz da árvore, pois é este atributo que fornecerá a maior redução na entropia, classificando os dados de forma mais rápida.

Para conhecer o valor do ganho de informação devem ser feitos dois cálculos:

- A entropia conjunta, ou seja, para todo o conjunto de dados – neste caso levando-se em consideração os subconjuntos referentes às classificações existentes;
- A entropia individual para cada atributo do conjunto de dados.

Considere um conjunto de amostras, contendo um atributo classe definido como A e um dos atributos preditivos definido como B. O ganho de informação (GI) do atributo preditivo B é definido como sendo a diferença entre a entropia do atributo classe A ($Entropia\ (A)$) menos a entropia condicional do atributo preditivo B, tendo sido definido o valor do atributo classe A ($Entropia\ (B \mid A)$). Matematicamente, o ganho de informação é dado por (5.2).

$$GI(B,A) = Entropia(A) - Entropia(B \mid A) \quad (5.2)$$

A entropia condicional, definida como a entropia de um atributo preditivo B, sendo conehcido o atributo classe A, é dada por:

$$Entropia(B \mid A) = \sum_{i=1}^{m} p_i . Entropia(B \mid A = v_i) \quad (5.3)$$

onde m é o número total de classes do conjunto de amostras, B é o atributo preditivo que está sendo considerado. A é o atributo classe assumindo o valor v_i. Além disso, p_i é como definido antes, i.e., $p_i = p(A = v_i)$, é a proporção dada pela razão entre o número de amostras com valor v_i e o número total de amostras de S.

O termo $Entropia(B \mid A = v_i)$ é a entropia do atributo preditivo B sendo dado o valor do atributo classe $A = v_i$.

$$Entropia(B \mid A = v_i) = -\sum_{i=1}^{m} p(B \mid A = v_i) \log_2 p(B \mid A = v_i) \quad (5.4)$$

onde m é o número de classes que o atributo classe A pode assumir, $p(B \mid A = v_i)$ é a probabilidade condicional do atributo B, i.e., é a proporção dada pela razão entre o número de exemplos de B com $A = v_i$ e o número total de amostras na classe $A = v_i$.

5.4 Contribuições de J. Ross Quinlan

Os estudos que permitiram o aparecimento das árvores de decisão tiveram início com o professor Ross Quinlan da Universidade de Sidney. A sua contribuição foi a elaboração de um algoritmo chamado ID3 – Iterative Dichotomiser 3 - desenvolvido em 1983. Logo em seguida surgiu o C4.5, que deu origem ao algoritmo utilizado neste trabalho – o J4.8.

J. Ross Quinlan originalmente desenvolveu o algoritmo ID3 na Universidade de Sidney. O algoritmo ID3 faz uma avaliação da informação contida nos atributos segundo o seu ganho de informação. O atributo mais importante (com o maior valor de ganho de informação) é colocado na raiz e, de forma top-down, a árvore é construída recursivamente. Se um

atributo classifica perfeitamente um conjunto de treinamento o algoritmo pára; senão ele opera recursivamente nos "N" subconjuntos particionados para conseguir o melhor atributo.

Considere no algoritmo ID3 a seguir, que Amostras são as amostras do conjunto de treinamento, Classe é o atributo classe cujo valor será predito pela árvore, e Atributos é o conjunto de atributos preditivos, que podem ser testados pela árvore de decisão. Neste algoritmo, o resultado é o retorno de uma árvore de decisão que classifica corretamente o conjunto de Amostras fornecidas como parâmetros de entrada.

```
ID3(Amostras, Classe, Atributos)
Início
    Crie um nó raiz para a árvore
    Se A = (+) ∀A ∈ Amostras,
        Então retorne um nó marcado como positivo;
    Fim-se
    Se A = (-) ∀A ∈ Amostras,
        Então retorne um nó marcado como negativo;
    Fim-se
    Se Atributos preditivos for vazio
        Então retorne a árvore de nó único marcado com o valor mais comum da Classe no
        conjunto Amostras
    Fim-se
    Senão
        Para cada atributo preditivo (valor do conjunto Atributos) calcular a Entropia e GI
        Faça N igual ao atributo preditivo cujo valor de ganho de informação é o maior (max
        GI)
        Faça Raiz igual a N
        i=1;
        Para cada valor v_i ∈ N (i.e. para cada valor de N) faça
            Adicionar um novo ramo abaixo da Raiz, correspondendo a r_i = v_i;
            Faça R(i)={v_i} (armazenamento na lista de ramos do ramo r_i);
            Se R(i) for vazio
                Então para este ramo adicionar uma nova folha igual ao valor mais
                comum da Classe em Amostras
            Fim-se
            Senão (para este ramo)
                i=i+1;
                ID3(Amostras, Classe, Atributos);
            Fim-senão
        Fim-faça
    Fim-senão
Final
```

Figura 5.2: Algoritmo ID3.

O atributo preditivo selecionado para o nó raiz deve minimizar o trabalho de classificação do conjunto de treinamento, reduzindo a aleatoriedade no processo de particionamento e objetivando a simplificação da árvore de decisão. Este atributo é o que apresentar o maior valor de ganho de informação.

Uma forte limitação do algoritmo ID3 é que ele não trabalha com atributos do tipo contínuo.

5.5 Algoritmo *J4.8*

O algoritmo J4.8 é a última revisão do algoritmo C4.5, que por sua vez é uma versão estendida do ID3. Este algoritmo acrescenta algumas características como manipular atributos contínuos, lidar com valores nulos de atributos, evitar o sobre aprendizado, ou seja, evitar que a árvore de decisão represente com muita fidelidade os dados utilizados na sua geração, perdendo a capacidade de generalização para dados não conhecidos.

Sua filosofia segue o mesmo princípio do algoritmo ID3. Ele particiona recursivamente um conjunto de treinamento até que cada subconjunto deste particionamento contenha amostras de uma única classe.

Um avanço do algoritmo J4.8 é sua aplicação em dados com valores contínuos. Neste caso será produzida uma partição binária do conjunto de amostras, seguindo a ordem:

- Amostras onde valor do atributo < ponto de referência;
- Amostras onde valor do atributo > ponto de referência.

A escolha deste ponto de referência pode ser feita:

- Ordenando as amostras por ordem crescente dos valores do atributo;
- Qualquer ponto intermediário entre dois valores diferentes e consecutivos dos valores observados no conjunto de treinamento pode ser utilizado como possível ponto de referência;

- É usual considerar o valor médio entre dois valores diferentes e consecutivos.

Para a definição do nó raiz e os nós subseqüentes são utilizados o cálculo de entropia e ganho de informação GI. Mas, segundo QUINLAN [31], o critério de ganho de informação tem uma séria deficiência, favorecendo o atributo preditivo que apresenta grande variação de valores no conjunto de amostras. Assim foi desenvolvida a fórmula da razão do ganho RG, que é outra evolução, utilizada no algoritmo J4.8.

5.5.1 Razão do Ganho (RG) e Informação Dividida (ID)

A razão do ganho e a informação dividida foram desenvolvidas por QUINLAN [31] com o propósito de reduzir o peso da quantidade de classes como determinante para o cálculo do melhor atributo particionador, aquele atributo que será escolhido para ser a raiz da árvore. O critério da entropia tem a desvantagem de favorecer claramente os atributos que efetuem a partição do conjunto de amostras em maior número de intervalos. A medida da razão do ganho ameniza este favorecimento. A razão do ganho RG é definida como sendo a razão entre o ganho de informação (GI) e a informação dividida (ID).

A informação dividida é a técnica apresentada para normalizar o cálculo no caso de conjuntos de amostras com atributos apresentando grandes variações. Segundo SARMENTO [36], Quinlan diagnosticou que o ganho de informação de um atributo preditivo com muitos valores diferentes, sejam eles contínuos ou não, favorece a sua escolha como nó raiz, fazendo um cálculo sobre avaliado. Ainda segundo SARMENTO [36], a explicação encontra-se considerando, por absurdo, um atributo classe A detentor de tantos valores diferentes quanto amostras, sendo que para cada valor existe um atributo classe. Assim sendo, se considerarmos que para cada valor corresponde um subconjunto do atributo classe, pode-se afirmar que o ganho de informação do atributo A para estabelecer partições da classe é máximo. Esta sobre avaliação é retificada ponderando o ganho de informação pela quantidade média de informação (informação dividida) que é necessária gerar para identificar a que subconjunto do atributo pertence uma determinada amostra por ele caracterizada. Quanto maior for o número de valores distintos de um atributo, maior

será a quantidade de informação a transmitir para identificar que uma determinada amostra detém um determinado valor no atributo classe A.

Para tal foi desenvolvida uma fórmula que passa a dividir o valor do ganho de informação pelo cálculo da informação dividida.

$$ID = -\sum_{j=1}^{n} p_j \cdot \log_2 p_j \tag{5.5}$$

onde n é o número de classes do subconjunto de atributos preditivos que está sendo considerado, $p_j = p(B=v_j)$ é a probabilidade do atributo preditivo B ser igual ao valor do subconjunto cujo índice é j (*i.e., é a proporção do número de amostras com valor* v_j *em relação ao número total de amostras do subconjunto de amostras S*).

Finalizando, a razão do ganho então é dada por:

$$RG = \frac{GI}{ID} \tag{5.6}$$

5.6 Aplicando o Algoritmo J4.8

5.6.1 Construção de uma árvore

Considerando a base de dados da Tabela 5.1 e seguindo os passos para a construção de uma árvore de decisão, tem-se:

A base de dados apresentada é de um grupo de 12 alunos contendo informações sobre o curso de cada aluno, o esporte e a comida preferidos. Para ilustrar como se desenvolve uma árvore de decisão, considere que o objetivo seja definir se existem regras sobre preferência de alimentação dos alunos baseadas em outros atributos.

Tabela 5.1: Base de dados hipotética para construção de uma árvore de decisão.

	Atributos		
Alunos	Curso	Esporte	Comida
1	Computação	Futebol	Japonesa
2	Computação	Natação	Fast-Food
3	Computação	Natação	Fast-Food
4	Computação	Natação	Fast-Food
5	Matemática	Voleibol	Italiana
6	Matemática	Natação	Vegetariana
7	Matemática	Voleibol	Fast-Food
8	Biologia	Futebol	Fast-Food
9	Biologia	Futebol	Italiana
10	Biologia	Futebol	Vegetariana
11	Biologia	Futebol	Italiana
12	Biologia	Natação	Fast-Food

Neste caso, tem-se como atributos preditivos o curso e o esporte, e como atributo classe a comida. A comida é definida como atributo classe porque é ela que é a referência da pesquisa, i.e., se procura regras para descobrir a preferência de alimentação com base nos outros atributos. A base de dados possui 12 amostras para o atributo classe comida, com 4 diferentes valores: Fast-food, Japonesa, Italiana e Vegetariana. As amostras estão distribuídas sendo: 6 amostras para o valor Fast-food, 3 amostras para o valor Italiana, 2 amostras para o valor Vegetariana e apenas 1 amostra para o valor Japonesa.

1º. Passo:

O primeiro passo para a construção de uma árvore de decisão é o cálculo da entropia. Para fazê-lo precisa-se da proporção (probabilidade) de ocorrência de cada atributo. Este valor é calculado como o número de ocorrências de um valor de atributo dividido pelo número de amostras na base de dados. Este valor é dado por p_i, onde $p_i = p(A=v_i)$ é a probabilidade do atributo classe A ser igual à classe cujo índice é *i* (*i.e., é a proporção do número de amostras com valor* v_i *em relação ao número total de amostras de S*). Assim, cada item dos atributos Curso, Esporte e Comida apresenta a seguinte proporção:

Atributo Curso:

- p(Curso = Computação) = 4/12 = 0.333
- p(Curso = Matemática) = 3/12 = 0.250

- p(Curso = Biologia) = 5/12 = 0.417

Atributo Esporte:

- p(Esporte = Futebol) = 5/12 = 0.417
- p(Esporte = Natação) = 5/12 = 0.417
- p(Esporte = Voleibol) = 2/12 = 0.167

Atributo Comida:

- p(Comida = FastFood) = 6/12 = 0.500
- p(Comida = Japonesa) = 1/12 = 0.083
- p(Comida = Italiana) = 3/12 = 0.250
- p(Comida = Vegetariana) = 2/12 = 0.167

Na Tabela 5.2 tem-se os valores das probabilidades e da entropia parcial para os atributos. A entropia parcial é o valor da entropia de cada valor possível de um atributo A, dada pela equação (5.6).

$$\text{Entropia parcial} = -p_i \log_2 p_i \qquad (5.6)$$

Tabela 5.2: Valores das probabilidades e entropias parciais.

Atributo	Probabilidade (p_i)	Entropia parcial (- $p_i \log_2 p_i$)
Curso = Computação	0.333	0.5283
Curso = Matemática	0.250	0.5000
Curso = Biologia	0.417	0.5263
Esporte = Futebol	0.417	0.5263
Esporte = Natação	0.417	0.5263
Esporte = Voleibol	0.167	0.4308
Comida = Fast-Food	0.500	0.5000
Comida = Japonesa	0.083	0.2987
Comida = Italiana	0.250	0.5000
Comida = Vegetariana	0.167	0.4308

Usando a Tabela 5.2 como base, calcula-se a entropia de cada atributo, que é dada pela fórmula:

$$Entropia(A) = -\sum_{i=1}^{m} p_i \log_2 p_i \quad (5.1)$$

Para A = Curso e i assumindo valores de Computação, Matemática e Biologia:

Entropia(Curso) = (-0.333log$_2$0.333 - 0.250log$_2$0.250 - 0.417log$_2$0.417)

Entropia(Curso) = (0.5283 + 0.500 + 0.5263) = 1.5546

Para A = Esporte e i assumindo valores de Futebol, Natação e Vôlei:

Entropia(Esporte) = (-0.417log$_2$0.417 - 0.417log$_2$0.417 - 0.167log$_2$0.167)

Entropia (Esporte) = (0.5263 + 0.5263 + 0.4308) = 1.4834

Para A = Comida e i assumindo valores de Fast-food, Japonesa, Italiana e Vegetariana:

Entropia(Comida) = (-0.500log$_2$0.500 - 0.083log$_2$0.083 - 0250log$_2$0.250 - 0.417log$_2$0.417)

Entropia(Comida) = (0.500 + 0.2987 + 0.500 + 0.4308) = 1.7296

O próximo passo para o algoritmo J4.8 é fazer o cálculo do Ganho de Informação (GI). Esse cálculo é dado pela fórmula:

$$Ganho(B, A) = Entropia(A) - Entropia(B \mid A) \quad (5.2)$$

Para simplificar é realizado separadamente o cálculo da entropia condicional, dada por:

$$Entropia(B \mid A) = \sum_{i=1}^{m} p_i . Entropia(B \mid A = v_i) \quad (5.3)$$

O cálculo da entropia condicional utiliza a probabilidade condicional. A probabilidade condicional é a medida da probabilidade de um atributo B assumir um valor, considerando que um atributo classe A já possui um valor pré-definido. Como no exemplo a classificação precisa ser feita em relação ao atributo comida, ela é a variável A, ou seja, o atributo classe.

Assim, os cálculos das probabilidades condicionais do curso ser computação, dado que a comida é Fast-food, Japonesa, Italiana e Vegetariana é dada, respectivamente, por:

- p(curso = computação | comida = fast-food) = 3/6 = 0.500
- p(curso = computação | comida = japonesa) = 1/1 = 1.000
- p(curso = computação | comida = italiana) = 0/3 = 0.000
- p(curso = computação | comida = vegetariana) = 0/2 = 0.000

Os cálculos das probabilidades condicionais do curso ser matemática, dado que a comida é Fast-food, Japonesa, Italiana e Vegetariana é dada, respectivamente, por:

- p(curso = matemática | comida = fast-food) = 1/6 = 0.166
- p(curso = matemática | comida = japonesa) = 0/1 = 0.000
- p(curso = matemática | comida = italiana) = 1/3 = 0.333
- p(curso = matemática | comida = vegetariana) = 1/2 = 0.500

E, finalmente, os cálculos das probabilidades condicionais do curso ser biologia, dado que a comida é Fast-food, Japonesa, Italiana e Vegetariana é dada, respectivamente, por:

- p(curso = biologia | comida = fast-food) = 2/6 = 0.333
- p(curso = biologia | comida = japonesa) = 0/1 = 0.000
- p(curso = biologia | comida = italiana) = 2/3 = 0.667
- p(curso = biologia | comida = vegetariana) = 1/2 = 0.500

Com estes valores se chega à Tabela 5.3 abaixo, que são os valores das probabilidades condicionais de cada atributo.

Tabela 5.3: Probabilidades Condicionais para o curso dado a comida.

	Computação	Matemática	Biologia
Fast-Food	0.500	0.166	0.333
Japonesa	1.000	0.000	0.000
Italiana	0.000	0.333	0.667
Vegetariana	0.000	0.500	0.500

O cálculo da entropia condicional parcial para cada classe de comida é realizado utilizando a fórmula 5.1, considerando que o p_i agora é dado pelas probabilidades condicionais, utilizando os valores da Tabela 5.3. Assim:

- Entropia(Curso | Comida=Fast-Food)) = (-0.500$\log_2$0.500 - 0.166$\log_2$0.166 - 0.333$\log_2$0.333)
- Entropia (Curso | Comida=Fast-Food) = (0.500 + 0.4308 + 0.5283) = 1.4591

- Entropia (Curso | Comida=Japonesa) = (-1.000$\log_2$1.000 - 0.000$\log_2$0.000 - 0.000$\log_2$0.000)
- Entropia (Curso | Comida=Japonesa) = (0.000 + 0.000 + 0.000) = 0.000

- Entropia (Curso | Comida=Italiana) = (-0.000$\log_2$0.000 - 0.333$\log_2$0.333 - 0.667$\log_2$0.667)
- Entropia (Curso | Comida=Italiana) = (0.000 + 0.5283 + 0.3900) = 0.9183

- Entropia (Curso | Comida=Vegetariana) = (-0.000$\log_2$0.000 - 0.500$\log_2$0.500 - 0.500$\log_2$0.500)
- Entropia (Curso | Comida=Vegetariana) = (0.000 + 0.500 + 0.500) = 1.000

Por fim, com estes valores e a Tabela 5.3, calcula-se a entropia condicional (Veja Tabela 5.4).

Tabela 5.4: Valores da Entropia Condicional para o curso dado a comida.

Comida	Probabilidade	Entropia Parcial	Entropia(Curso|Comida)
Fast-Food	0.500	1.4591	0.7296
Japonesa	0.083	0.0000	0.0000
Italiana	0.250	0.9183	0.2296
Vegetariana	0.166	1.0000	0.1667
Soma			1.1258

De forma semelhante, tem-se a Tabela 5.5 para a entropia condicional do esporte dado a comida.

Tabela 5.5: Valores da Entropia Condicional para o esporte dado a comida.

Comida	Probabilidade	Entropia Parcial	Entropia(Esporte|Comida)
Fast-Food	0.500	1.2516	0.6258
Japonesa	0.083	0.0000	0.0000
Italiana	0.250	0.9183	0.2296
Vegetariana	0.166	1.0000	0.1667
Soma			1.0221

Agora, o ganho de informação GI é dado por:

$$GI(B,A) = Entropia(A) - Entropia(B \mid A) \qquad (5.2)$$

O problema continua sendo a classificação dos dados de acordo com a comida, ou seja, ela é nossa variável A. A variável B varia entre os atributos preditivos curso e esporte.

GI(Curso , Comida) = Entropia(Comida) – Entropia (Curso | Comida)

GI(Curso , Comida) = 1.7296 – 1.1258 = 0.6038

GI(Esporte , Comida) = Entropia(Comida) – Entropia(Esporte | Comida)

GI(Esporte , Comida) = 1.7296 – 1.0221 = 0.7075

Como o GI do esporte dado a comida é maior, conclui-se que é mais compacto classificar a comida a partir do esporte e assim o esporte é o nó pai, como na Figura 5.3.

Figura 5.3: Árvore parcial gerada.

Se existissem mais atributos na base de dados o algoritmo calcularia, de forma recursiva, o GI novamente para os atributos, eliminando da base de dados o atributo preditivo já escolhido, para definir qual seria o novo atributo preditivo pai. Isto ocorre no algoritmo até

se chegar a um ponto onde não se tem mais atributos preditivos para serem nós pais e estes se tornam nós folhas, finalizando a árvore.

Neste exemplo não existem mais atributos preditivos para serem escolhidos a não ser o atributo preditivo curso. Desta forma, a árvore de decisão poderia ser representada como na Figura 5.4.

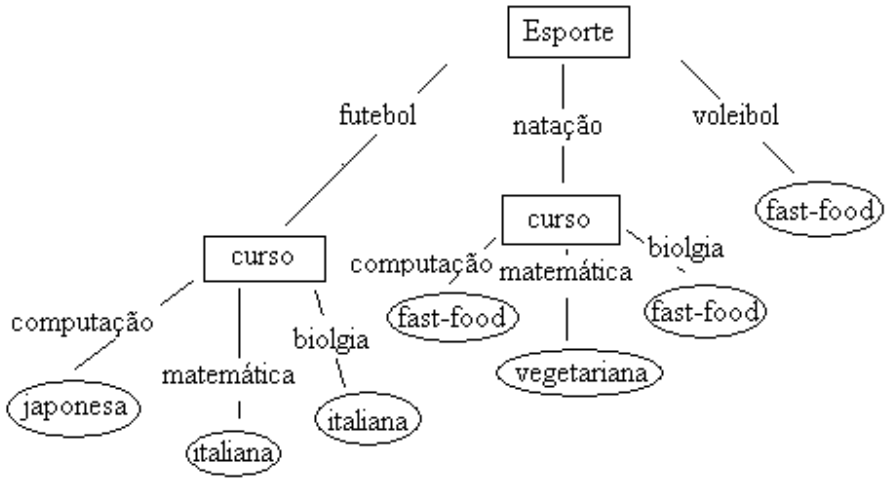


Figura 5.4: Árvore gerada.

Neste exemplo da árvore de decisão, a base de dados utilizada não possui nenhum atributo capaz de efetuar a partição do conjunto de amostras em vários intervalos. Assim, a desvantagem do método do ganho de informação de favorecer claramente os atributos que efetuem a partição do conjunto de amostras em maior número de intervalos não se verifica. Neste caso, não se fez necessário o cálculo da razão do ganho. Uma base de dados para exemplificar a utilização destes cálculos precisaria conter um número maior de amostras, o que acarretaria em inúmeros cálculos, extendendo demasiadamente a exemplificação. Quando é necessário se aplicar a razão do ganho, os cálculos devem ser feitos logo após os cálculos de entropia e ganho de informação. O atributo preditivo que apresentar o maior valor da razão do ganho seria escolhido como nó raiz. O restante do processametno do algoritmo não sofre alterações.

5.7 Poda em Árvores de Decisão

Segundo QUINLAN [31], os procedimentos para o processo de criação das árvores de decisão são executados recursivamente e continuarão dividindo o conjunto de treinamento até que, após um particionamento, exista somente uma classe ou os testes subseqüentes não aperfeiçoem a árvore de decisão, ou seja, a árvore estende a sua profundidade até o ponto de classificar perfeitamente os elementos do conjunto de treinamento.

No trabalho de HAN e KAMBER [14] é afirmado que muitos galhos refletem incorreções provocadas pelo conjunto de treinamento devido a ruídos. Quando o conjunto de treinamento possui ruídos ou não é representativo, o algoritmo pode produzir árvores em que há um ajuste forçado. Os métodos de poda de árvores de decisão abordam os problemas com base em fórmulas estatísticas, de forma que os galhos menos confiáveis sejam removidos. O processo de poda deve produzir árvores de decisão capazes de efetuar as classificações de forma mais rápida e com nível de precisão superior à árvore de decisão original.

No trabalho de CUROTTO e EBECKEN [10] são citadas duas abordagens possíveis para poda em árvores de decisão. Estas abordagens são a poda com redução de erros (*reduced error pruning*) e a poda posterior (*post-pruning*).

O algoritmo J4.8 faz a simplificação (poda) através da abordagem da poda posterior. Apesar do custo computacional ser bem maior, tem-se no final melhores simplificações, pois se pode explorar melhor os custos na árvore.

O algoritmo de simplificação começa pelo nível mais baixo da árvore e vai examinando cada nó. Para cada nó ele verifica se a exclusão da sub-árvore em favor de uma folha ou a substituição pelo galho mais freqüente deste nó irá diminuir o erro de treinamento, caso positivo a operação é realizada.

Um problema na fase de poda é como prever os erros que guiarão o processo. A taxa de erro obtida simplesmente usando o conjunto de treinamento utilizado para construir a árvore não oferece uma estimativa razoável. De acordo com QUINLAN [31], existem duas

famílias de técnicas para predição de erro. A primeira prevê a taxa usando um conjunto separado de dados (às vezes chamado de dados de poda), distinto do conjunto de treinamento. Uma vez que estes dados não foram examinados no momento em que a árvore foi construída, as estimativas obtidas a partir dos mesmos não são tendenciosas e, se existirem casos suficientes, são confiáveis. A abordagem da poda com redução de erro é um exemplo desta aplicação.

As desvantagens associadas a estas técnicas estão relacionadas à necessidade de possuir à disposição um grande volume de dados, de tal forma que se possa separar um porção para construir a árvore e uma outra significativa para podá-la.

Ainda segundo QUINLAN [31], a outra família para predição de erro usa apenas os dados de treinamento a partir do qual a árvore foi construída. Um exemplo é a poda posterior. Neste caso o erro verdadeiro associado a um nó folha é estimado como sendo o limite superior para a distribuição binomial para algum nível de confiança *CF* (ex. 25%). Este limite superior é escrito como U_{CF}(E,N), onde N representa o número de casos cobertos pelo nó folha e E o número de casos erradamente cobertos. Se a taxa de erro pessimista for maior que a da árvore original, então não faz sentido excluir a condição.

Ao invés de se utilizar uma busca exaustiva para encontrar o subconjunto das condições dentre todas as existentes que deve ser excluído, uma abordagem baseada em um algoritmo guloso é utilizada. Cada iteração deste procedimento é repetida até que não haja melhora na precisão do classificador.

5.8 Conclusão

Muitos são os algoritmos de classificação que montam árvores de decisão. A escolha de um melhor algoritmo é uma tarefa peculiar, já que um algoritmo pode ter melhor desempenho em determinada situação e outro pode ser mais eficiente em outros tipos de situações.

A técnica de árvore de decisão é vista como uma boa escolha quando o objetivo é gerar regras que podem ser facilmente entendidas, explicadas e traduzidas para a linguagem

natural. No trabalho proposto este é um dos objetivos quanto à identificação de defeitos incipientes em transformadores.

Uma questão chave para a construção de uma árvore de decisão consiste na estratégia para a escolha dos atributos que estarão mais próximos da raiz da árvore (ou seja, os atributos que são inicialmente avaliados para determinar a classe a qual uma instância pertence).

A construção de uma árvore de decisão tem três objetivos: diminuir a entropia (ou seja, a aleatoriedade da variável objetivo), ser consistente com o conjunto de dados e possuir o menor número de nós.

6 Base de Dados, Pré-Processamento e Metodologia

6.1 Introdução

Neste capítulo apresenta-se a descrição das bases de dados utilizadas. É feita uma análise dos pré-processamentos necessários para que as bases de dados se enquadrem nos algoritmos aplicados e como estes pré-processamentos foram realizados. Cada base de dados é organizada em dois conjuntos, sendo um de treinamento e o outro de validação. Os treinamentos foram feitos com 70% dos dados e as validações com os 30% restantes. As bases de dados de Hepatite e de DNA foram utilizadas como benchmark, uma vez que existem resultados de classificação publicados na literatura utilizando-as para validação. Estas duas bases serviram como referência para comparação dos resultados obtidos pela rede MLP e árvore de decisão utilizadas neste trabalho. Após esta análise, passou-se ao estudo da base de dados relacionada aos testes de cromatografia de transformadores. São três bases de dados de cromatografia publicadas na dissertação de mestrado de MORAIS [27], contendo dados da IEC TC10, CEPEL e de um especialista de uma grande empresa do setor elétrico. Aqui estas bases de dados foram denominadas de IEC, Base 1 e Base 2.

A seguir são apresentadas as bases de dados utilizadas na análise da rede MLP e árvore decisão, baseada no algoritmo J4.8.

6.2 Hepatite

A base de dados foi retirada do endereço <http://mlearn.ics.uci.edu/MLSummary.html>, com acesso em Março de 2007.

É uma base de dados médica, contendo informações clínicas de um grupo de pessoas e o diagnóstico se essas pessoas possuem ou não a patologia de hepatite. Da base de dados, 86 casos não apresentam a patologia e 69 casos a apresentam.

Sua forma original contém 155 instâncias com 20 atributos cada, sendo um deles a classe. Quatorze destes atributos são normalizados (podendo ser considerados apenas como 0 ou 1). Os outros são numéricos, com variações razoavelmente consideráveis.

Do montante dos atributos a quantidade de instâncias vazias é relevante. Tem-se por exemplo, 29 instâncias vazias no atributo fosfato alcalino.

No caso da eliminação dos casos vazios, o procedimento adotado foi completar os dados com a média dos valores mais próximos, sendo que o atributo que apresentava 67 dados vazios (43% dos valores) foi expurgado da base de dados. Com isso a base de dados para aplicação dos algoritmos ficou com 155 instâncias com 19 atributos cada.

Para as simulações o conjunto de dados foi agrupado da seguinte forma:

- um conjunto de treinamento com 52 instâncias classificadas como doente e 65 como sadio.
- um conjunto de validação com 17 instâncias classificadas como doente e 21 como sadio.

6.3 DNA

A base de dados foi retirada do endereço <http://www.cs.utoronto.ca/~delve/data/splice/spliceDetail.html> , com acesso em Junho de 2007.

É uma base de dados com uma cadeia de DNA, na qual o objetivo da classificação é dizer se a seqüência tem característica *intron* (IE), *exon* (EI) ou indiferente (N). A seqüência que for *exon* tem a característica de codificar aminoácidos de proteínas e a seqüência que for *intron* representa um DNA não codificante de qualquer parte da proteína.

Sua forma original contém 3159 instâncias com 61 atributos cada, sendo um deles a classe. Todos os 60 atributos recebem a classificação de A, C, G ou T, que são as iniciais dos nucleotídeos. Da base de dados tem-se 1639 instâncias classificadas como N, 759 instâncias

classificadas como EI e 761 instâncias classificadas como IE. Não é apresentado nenhum atributo com valor vazio.

Neste caso o pré-processamento foi simples. Como a base de dados não apresentava nenhum campo em branco e a faixa de "valores" – A, C, T e G – é pequena, o único pré-processamento foi a reorganização da base de dados. Para um melhor teste dos algoritmos a base de dados foi dividida em duas, sendo uma delas denominada balanceada (classes IE e EI) e a outra, desbalanceada (classes EI e N).

Para as simulações as bases de dados ficaram estruturadas da seguinte forma:

Balanceada: 1520 instâncias com 61 atributos cada, sendo um deles a classe (EI ou IE). Da base de dados tem-se 759 classificadas como EI e 761 classificadas como IE.

Este conjunto de dados foi dividido para compilação dos algoritmos:

- um conjunto de treinamento com 571 instâncias classificadas como IE e 568 como EI
- um conjunto de validação com 190 instâncias classificadas como IE e 191 como EI.

Desbalanceada: 2398 instâncias com 61 atributos cada, sendo um deles a classe (EI ou N). Da base de dados tem-se 759 classificadas como EI e 1639 classificadas como N.

Este conjunto de dados foi dividido para compilação dos algoritmos:

- um conjunto de treinamento com 1239 instâncias classificadas como N e 555 como EI
- um conjunto de validação com 400 instâncias classificadas como N e 204 como EI.

6.4 Transformadores

Para aplicação dos algoritmos estudados, foram utilizadas três bases de dados contendo os diagnósticos de defeitos e as concentrações de gases diluídos no óleo isolante dos transformadores. As bases são descritas abaixo:

- IEC: composta de 53 amostras com diagnósticos determinados através de medições específicas e inspeções visuais feitas por especialistas, com 17 amostras com diagnóstico de normalidade, 22 apresentando falha elétrica e 14 apresentando falha térmica;
- Base 1: composta por amostras com diagnósticos determinados através de medições específicas e inspeções visuais, feitos por especialistas, considerando transformadores de vários níveis de tensão. A base de dados totaliza 224 amostras, divididas em 83 amostras com diagnóstico de normalidade, 61 com falha elétrica e 80 com falha térmica; e
- Base 2: composta por amostras com diagnósticos determinados através de medições específicas e feitos por especialistas num total de 212 amostras, divididas em 180 amostras com diagnóstico de normalidade, 10 com falha elétrica e 22 com falha térmica.

Foram constituídos três grupos para simulações:

- na primeira análise os dados foram treinados com 70 % dos dados da base IEC e validados com os 30% dos dados restantes, isso tanto para a rede neural quanto para árvore de decisão. A rede neural e a árvore de decisão geradas foram utilizadas para classificar as bases de dados Base 1 e Base 2. Realizou-se o mesmo procedimento considerando as bases de dados de geração e utilização das técnicas balanceadas.
- na segunda análise os dados foram treinados com 70% dos dados da Base 1 e validados com os 30% dos dados restantes, isso tanto para rede neural quanto para árvore de decisão. A rede neural e a árvore de decisão geradas foram utilizadas para classificar a base de dados IEC e Base 2. Realizou-se o mesmo procedimento considerando as bases de dados de geração e utilização das técnicas balanceadas.

- na terceira análise os dados das bases IEC e Base 1 foram agrupados, o treinamento feito com 70% deles e a validação com os 30% dos dados restantes (também das duas bases), isso tanto para rede neural quanto para árvore de decisão. A rede neural e a árvore de decisão geradas foram utilizadas para classificar os dados da Base 2. Realizou-se o mesmo procedimento considerando as bases de dados de geração e utilização das técnicas balanceadas.

Os diagnósticos de normalidade, falha elétrica e falha térmica foram codificados da seguinte forma:

- Classificação tipo A: transformador com diagnóstico de normalidade;
- Classificação tipo B: transformador com diagnóstico de falha elétrica;
- Classificação tipo C: transformador com diagnóstico de falha térmica.

6.5 Softwares utilizados

6.5.1 Redes neurais

Para analisar a técnica de redes neurais MLP na mineração de dados foi utilizado o *software* MATLAB e sua ferramenta *Neural Networks*.

6.5.1.1 MatLab

O MATLAB é uma ferramenta computacional que permite a realização de aplicações em diversas áreas da ciência como análise numérica, cálculo matricial, processamento de sinais, otimização, controle de processos, solução de equações diferenciais, entre outras.

É um sistema interativo cujo elemento básico de informação é uma matriz que não requer dimensionamento. Este sistema permite a resolução de problemas numéricos em apenas uma fração do tempo que se gastaria para escrever um programa semelhante numa linguagem de programação clássica.

6.5.1.1.1 Formato de entrada de dados no MatLab

Para execução da rede MLP no *software* MatLab foi preciso alocar cada base de dados em duas matrizes, sendo uma delas contendo os dados de instância x atributo e a outra as classes. Estas duas matrizes estão representadas nas Figuras 6.1 e 6.2.

1	6	17	0,04	16	6
2	10	9	0,04	14	10
3	80	34	0,04	19	15
4	7	75	0,04	16	9
5	49	34	0,04	32	18
6	81	47	0,04	27	16
7	39	41	2	29	16
8	73	45	0,04	29	23
9	52	2	0,04	15	17
10	46	12	0,04	9	2
11	320	30	0,04	19	27
12	610	17	0,04	15	18
13	230	20	0,04	12	17
14	410	19	0,04	19	19

Figura 6.1 - Matriz instância x atributo, para o exemplo dos transformadores.

1	A
2	A
3	A
4	B
5	C
6	C
7	B
8	B
9	B
10	A
11	A

Figura 6.2 - Matriz classe, para o exemplo dos transformadores.

Na Figura 6.1 apresenta-se nas colunas, respectivamente, o número da amostra, as concentrações dos gases hidrogênio (H_2), metano (CH_4), etileno (C_2H_4), etano (C_2H_6) e acetileno (C_2H_2), que são gases produzidos no interior do transformador. Estes são os principais gases encontrados em dados históricos e a produção dos mesmos se dá principalmente por:

- hidrogênio: grandes quantidades associadas com condições de descarga parcial;

- hidrogênio, etano, metano e etileno: resultados da decomposição térmica do óleo, ou seja, contato do óleo isolante com partes quentes;
- acetileno: associado com arco elétrico no óleo.

Na Figura 6.2 tem-se apenas o número do teste e seu diagnóstico.

Como já explicado, os treinamentos em redes neurais foram feitos para uma rede MLP com o algoritmo de retropropagação resiliente, no qual a taxa de aprendizado é adaptativa, individual para cada peso, e a influência indesejável da magnitude das derivadas (que ocorre no algoritmo de retropropagação) é ignorada. Devido a estas características ele tem a convergência acelerada em relação ao de retropropagação.

As simulações foram realizadas com variações da quantidade de neurônios e da função de ativação, que tem o papel de mapear a camada de saída de acordo com as entradas da rede. Foram realizadas simulações com as três funções de ativação mais utilizadas, sendo elas a tangente hiperbólica, a sigmóide e a linear, sendo esta última utilizada apenas para a saída.

Para cada uma destas funções de ativação as bases de dados foram submetidas aos diferentes parâmetros:

- quantidade de iterações (ou ciclos): em cada conjunto de teste, o conjunto utilizado para treinamento da rede foi submetido às seguintes quantidades de iterações: 1.000, 4.000 e 8.000.
- quantidade de neurônios intermediários (ou escondidos) da rede: a rede foi treinada variando-se o número dos neurônios da camada escondida. Foram realizados testes com 4, 6, 8 e 10 neurônios.

6.5.2 Árvores de Decisão

Para a simulação do algoritmo J4.8 foi utilizado o software weka *(Waikato Environment for Knowledge Analysis).*

Este *software* é formado por um conjunto de algoritmos de diversas técnicas para resolver problemas concretos de mineração de dados. O WEKA está implementado em linguagem Java e foi desenvolvido no meio acadêmico da Universidade de Waikato, na Nova Zelândia, em 1999. O WEKA foi escolhido tendo em vista sua praticidade de utilização, bem como o fato de ser um *software* de domínio público estando disponível em: http://www.cs.waikato.ac.nz/ml/weka.

Os algoritmos que compõem o WEKA são alocados em forma de pacotes. Foi seguida a mesma filosofia para a simulação deste trabalho. O algoritmo J4.8 está implementado em Java e foi rodado na plataforma Eclipse 32. Esta estrutura de pacotes define uma hierarquia no algoritmo, como por exemplo, criar uma estrutura com os atributos para depois chamar a estrutura de classificação e geração de regras.

6.5.2.1 Plataforma Eclipse 32

Eclipse é uma plataforma universal para a integração de ferramentas construídas por uma comunidade aberta de desenvolvedores de *softwares*. Esta plataforma foi projetada para construir ambientes de desenvolvimento integrados (IDEs) que podem ser utilizados para criar os mais diversos tipos de aplicação.

Uma grande vantagem da plataforma Eclipse é que ela é independente do sistema operacional. Segue uma filosofia para facilitar a integração de ferramentas, adiciona facilmente novas ferramentas a produtos já instalados.

6.5.2.1.1 Formato do arquivo de entrada do weka

Para a simulação no algoritmo J4.8 foi utilizado o padrão ARFF para os arquivos de entrada. Este padrão é utilizado para representar uma série de dados que consistem em exemplos independentes.

Em um arquivo com extensão ARFF, tem-se:

- opcionalmente, linhas começando com o símbolo %: significa que a linha é um comentário, não tendo validade junto ao processamento realizado pelo algoritmo;
- a primeira linha válida indica o nome da relação a encontrar (@relation nome_da_relação);
- as linhas seguintes devem listar todos os atributos, onde deve-se definir o tipo do atributo e os valores que ele pode representar. Neste último caso os valores devem estar entre "{ }" e separados por vírgulas;
- no próximo bloco de informações, após uma linha de indicação (@data), vêm as instâncias, ou seja, os registros a serem minerados com o valor dos atributos para cada instância separado por vírgula; a ausência de um item em um registro deve ser atribuída pelo símbolo "?".

Na Figura 6.3 abaixo, tem-se a representação de um arquivo .arff para a base de dados dos transformadores IEC TC10.

```
@relation IEC

@attribute H2 real
@attribute CH4 real
@attribute C2H2 real
@attribute C2H4 real
@attribute C2H6 real
@attribute diag {A, B, C}

@data
134,134,0.04,45,157,A
100,200,20,200,200,A
0.04,225,3,110,225,A
```

Figura 6.3 - Arquivo no formato ARFF contendo exemplos dos transformadores.

Os termos de A a C da Figura acima, como já explicado, referem-se a diagnósticos dos transformadores.

As simulações foram feitas variando os parâmetros de poda ou não da árvore e do fator de confiança (CF). O fator de confiança é uma forma simples de avaliar a precisão das regras obtidas nos dados de treinamento. Este fator é calculado pela razão X/Y, onde X é o número de registros que satisfazem o antecedente e o conseqüente da regra e Y é o número total de registros que satisfazem o antecedente da regra.

6.6 Conclusão

Neste capítulo foram descritas as bases de dados utilizadas e os pré-processamentos das mesmas. É explícito que os pré-processamentos realizados são aqueles que não dependem muito do conhecimento do domínio – tratamento de classes desbalanceadas e de valores em branco nas bases de dados.

O capítulo também descreveu a metodologia empregada na realização das simulações, bem como os softwares utilizados. As bases de dados foram divididas em conjunto de treinamento e de validação, com proporções de 70% e 30% dos dados respectivamente.

7 Resultados

7.1 Introdução

Neste capítulo serão apresentados os resultados das simulações, tanto nas bases de dados consideradas *benchmark*, quanto na base de dados dos transformadores. É feita uma comparação entre resultados existentes na classificação das bases hepatite e DNA, validando assim os algoritmos utilizados. Em seguida são apresentados os resultados da classificação de falhas em transformadores de potência relacionadas às concentrações de gases em seus óleos.

7.2 Comparativo entre resultados existentes e os simulados (hepatite e DNA)

As simulações realizadas no desenvolvimento deste trabalho, tendo como base os dados das bases de dados hepatite e DNA, geraram os resultados apresentados em porcentagem de índice de concordância na Tabela 7.1.

Para os dados de hepatite, as simulações foram realizadas utilizando 70% dos dados para treinamento e 30% para validação.

Para os dados de DNA, as simulações foram realizadas considerando duas bases de dados: uma com as classes distribuídas de forma balanceada, e a outra, desbalanceada. Também nos dois casos foi utilizada a proporção de 70% dos dados para treinamento e 30% dos dados para validação.

Tabela 7.1: Índice de Concordância Percentual - Resultados para Hepatite e DNA.

	Rede Neural		Árvore de Decisão	
	Índice de Concordância (%)		Índice de Concordância (%)	
	Treinamento	Validação	Treinamento	Validação
Hepatite	92,7	73,1	91,3	80,4
DNA (B)	98,6	94,8	98,6	97,5
DNA (D)	97,2	86,1	97,6	93,1

Para se poder analisar a qualidade destes resultados, é interessante ver alguns resultados encontrados na literatura.

ZENCO *et al.* [43] fez um trabalho de classificação utilizando uma seleção de 21 bases de dados do repositório UCI, dentre elas a base de dados hepatite. Foram feitas simulações com o algoritmo Naive Bayes, no qual os resultados encontrados utilizaram o método de validação 10-fold-cross validation, e o algoritmo K-vizinhos mais próximos, no qual o k foi selecionado com validação cruzada em uma faixa de 1 a 77 e o peso da distância inversa foi usado. Os índices de concordância apresentados foram, respectivamente, 84,65% e 82,71%.

AFTARCZUK [1] também fez um trabalho de classificação utilizando uma seleção de bases de dados do repositório UCI, dentre elas a base de dados hepatite. Foram feitas simulações com vários algoritmos. Destacam-se aqui os resultados conseguidos com o algoritmo Naive Bayes: utilizando a validação cruzada com 10 variações (índice de concordância de 71,3%), com 5 variações (índice de concordância de 69,8%), com 15 variações (índice de concordância de 71,3%); o outro método de validação utilizado foi o split, com as porcentagens de 30% (índice de concordância de 73,6%), de 50% (índice de concordância de 64,6%) e de 66% (índice de concordância de 81,8%).

Em HOFFMAN *et al.* [16] foi feito um estudo sobre a base de dados do DNA. Neste caso foi feita uma classificação da base de dados utilizando o algoritmo de extração de regra clementine. O resultado encontrado foi de um índice de concordância de 71,45%.

ASUNCION, NEWMAN [5] publicaram no trabalho Machine Learning Repository um estudo sobre a base de dados DNA, onde foram utilizados aleatoriamente 1000 instâncias das 3190 presentes na base de dados. As classificações dos dados foram em relação às classes:

- *exon* – denominada no trabalho de EI e que representa uma seqüência de DNA com características dominantes para codificar aminoácidos;
- *intron* – denominada no trabalho de IE e que representa uma seqüência de DNA com características dominantes para não codificar aminoácidos;
- indiferente – denominada no trabalho de N e que representa uma seqüência de DNA sem nenhuma das duas características marcantes.

Os índices de concordância na validação, em porcentagem, apresentados pelos algoritmos utilizados estão no quadro abaixo:

Tabela 7.2: Índice de Concordância Percentual: ASUNCION, NEWMAN [5].

Algoritmo	Base de Dados		
	Índice de Concordância (%)	Índice de Concordância (%)	Índice de Concordância (%)
	N	EI	IE
KBANN	95,4	92,4	91,5
PEBLS	93,1	91,8	92,4
COBWEB	88,2	84,9	90,5
Near Neighboor	68,9	88,3	90,9

Comparando os resultados publicados na literatura, pode-se considerar que os algoritmos utilizados neste trabalho apresentaram resultados iguais ou superiores. Assim, os algoritmos de rede neural e árvore de decisão foram considerados validados e assim poderão ser utilizados na base de dados de cromatografia para detecção de faltas incipientes em tranformadores de potência.

7.3 Resultados para os transformadores

Para a classificação de defeitos incipientes em transformadores de potência em relação às concentrações dos gases foram realizadas quatro análises. Em todas elas foram utilizados 70% dos dados para treinamento e os 30% restantes para a validação. Além disso, as três primeiras análises foram realizadas considerando ambas as bases de dados na forma original, isto é, desbalanceadas, e balanceadas (através da inserção do mesmo número de amostras para cada tipo de diagnóstico). Na quarta análise considerou-se o caso das bases

de dados balanceadas e com a amostra normalizada pelo TGC (Total de Gases Combustíveis). Outra característica das análises é a utilização de uma avaliação cruzada, ou seja, os dados são treinados e validados com uma base de dados e em seguida, a árvore de decisão e a rede neural geradas são aplicadas às outras bases de dados.

Em todas as análises com redes neurais foram feitas 36 configurações para determinar a melhor topologia, variando os parâmetros de acordo com o relatado no capítulo anterior. Foram então realizadas 12 simulações com a função de ativação, para a camada de saída, sendo a logsig, 12 sendo a tansig e 12 com a purelin. Nestas 12 simulações foram variados os números de neurônios e de iterações. Os resultados apresentados a seguir correspondem aos melhores resultados para cada análise.

Na primeira análise foi usada a base de dados IEC para treinamento e validação. Após, a árvore de decisão e a rede neural geradas foram aplicadas no diagnóstico dos dados de cromatografia das Bases 1 e 2. Os resultados globais (índices de concordância considerando toda a base de dados) são apresentados na Tabela 7.3, e os resultados separados por diagnósticos dos transformadores são apresentados na Tabela 7.4. Da mesma forma, os resultados para a base de dados balanceada estão apresentados nas Tabelas 7.5 e 7.6.

Tabela 7.3: Índice de Concordância Percentual global para a base desbalanceada.

Rede Neural				Árvore de decisão			
Índice de Concordância (%)				Índice de Concordância (%)			
Geração da Rede (IEC)		Diagnóstico		Montagem da Árvore (IEC)		Diagnóstico	
Trein.	Valid.	Base 1	Base 2	Trein.	Valid.	Base 1	Base 2
95,4	68,8	50,9	52,6	97,2	68,8	67,9	66,8

Tabela 7.4: Índice de Concordância Percentual discriminado por tipo de defeito para a base desbalanceada.

Defeitos	Rede Neural				Árvore de decisão			
	Índice de Concordância (%)				Índice de Concordância (%)			
	Geração da Rede (IEC)		Diagnóstico		Montagem da Árvore (IEC)		Diagnóstico	
	Trein.	Valid.	Base 1	Base 2	Trein.	Valid.	Base 1	Base 2
Normal	98,8	98,6	42,8	51,1	100,0	100,0	59,1	46,9
Def Elétrico	90,5	46,8	49,9	71,5	93,3	42,9	67,2	80,0
Def Térmico	99,0	98,0	65,6	61,1	100,0	100,0	77,5	72,7

Tabela 7.5: Índice de Concordância Percentual global para a base balanceada.

Rede Neural				Árvore de decisão			
Índice de Concordância (%)				Índice de Concordância (%)			
Geração da Rede (IEC)		Diagnóstico		Montagem da Árvore (IEC)		Diagnóstico	
Trein.	Valid.	Base 1	Base 2	Trein.	Valid.	Base 1	Base 2
95,4	77,9	59,8	60,5	97,8	83,6	75,9	69,7

Tabela 7.6: Índice de Concordância Percentual discriminado por tipo de defeito para a base balanceada.

Defeitos	Rede Neural				Árvore de decisão			
	Índice de Concordância (%)				Índice de Concordância (%)			
	Geração da Rede (IEC)		Diagnóstico		Montagem da Árvore (IEC)		Diagnóstico	
	Trein.	Valid.	Base 1	Base 2	Trein.	Valid.	Base 1	Base 2
Normal	100,0	100,0	45,8	68,2	100,0	100,0	65,9	88,8
Def Elétrico	86,7	57,2	63,9	52,6	93,4	57,2	86,1	60,7
Def Térmico	100,0	85,8	69,9	60,9	100,0	100,0	77,0	63,3

Neste caso a base de dados para validação contém apenas 7 amostras de cada tipo de diagnóstico, o que pode levar a um índice de concordância baixo como no caso do diagnóstico de defeito elétrico, onde resultou em 3 das 7 amostras classificadas de forma incorreta.

Na segunda análise foram usados os dados da Base 1 para treinamento e validação. A árvore de decisão e a rede neural geradas foram aplicadas no diagnóstico de faltas das bases IEC e Base 2. Os resultados gerais são apresentados na Tabela 7.7, e os resultados separados por diagnósticos dos transformadores são apresentados na Tabela 7.8. Da mesma forma, os resultados para a base de dados balanceada estão apresentados nas Tabelas 7.9 e 7.10.

Tabela 7.7: Índice de Concordância Percentual global para a base desbalanceada.

Rede Neural				Árvore de decisão			
Índice de Concordância (%)				Índice de Concordância (%)			
Geração da Rede (Base 1)		Diagnóstico		Montagem da Árvore (Base 1)		Diagnóstico	
Trein.	Valid.	IEC	Base 2	Trein.	Valid.	IEC	Base 2
91,6	66,3	57,7	52,8	92,8	76,5	72,8	61,3

Tabela 7.8: Índice de Concordância Percentual por tipo de defeito para a base desbalanceada.

Defeitos	Rede Neural				Árvore de decisão			
	Índice de Concordância (%)				Índice de Concordância (%)			
	Geração da Rede (Base 1)		Diagnóstico		Montagem da Árvore (Base 1)		Diagnóstico	
	Trein.	Valid.	IEC	Base 2	Trein.	Valid.	IEC	Base 2
Normal	93,9	54,1	63,1	63,6	94,8	68,0	75,0	68,2
Def Elétrico	91,9	60,3	82,8	78,7	93,1	72,3	90,9	84,6
Def Térmico	87,8	76,1	21,3	12,1	89,5	83,3	50,0	25,9

Tabela 7.9: Índice de Concordância Percentual global para a base balanceada.

Rede Neural				Árvore de decisão			
Índice de Concordância (%)				Índice de Concordância (%)			
Geração da Rede (Base 1)		Diagnóstico		Montagem da Árvore (Base 1)		Diagnóstico	
Trein.	Valid.	IEC	Base 2	Trein.	Valid.	IEC	Base 2
92,7	78,6	75,2	52,8	93,7	87,3	84,9	74,2

Tabela 7.10: Índice de Concordância Percentual por tipo de defeito para a base balanceada.

Defeitos	Rede Neural				Árvore de decisão			
	Índice de Concordância (%)				Índice de Concordância (%)			
	Geração da Rede (Base 1)		Diagnóstico		Montagem da Árvore (Base 1)		Diagnóstico	
	Trein.	Valid.	IEC	Base 2	Trein.	Valid.	IEC	Base 2
Normal	96,6	61,9	81,0	86,6	96,6	82,7	86,4	90,5
Def Elétrico	94,8	60,3	92,8	56,5	93,1	92,9	96,2	66,5
Def Térmico	86,3	76,1	52,4	67,6	91,4	88,1	72,8	68,7

Na terceira análise foram agrupadas os dados das bases IEC e Base 1 para serem utilizados no treinamento e validação da rede MLP e montagem da árvore de decisão. A árvore de decisão e a rede neural geradas foram aplicadas no diagnóstico dos dados da Base 2. Os resultados gerais são apresentados na Tabela 7.11, e os resultados separados por tipo de diagnóstico dos transformadores são apresentados na Tabela 7.12. Da mesma forma, os resultados para a base de dados balanceada estão apresentados nas Tabelas 7.13 e 7.14.

Tabela 7.11: Índice de Concordância Percentual global para a base desbalanceada.

Rede Neural			Árvore de decisão		
Índice de Concordância (%)			Índice de Concordância (%)		
Geração da Rede (IEC + Base 1)		Diagnóstico	Montagem da Árvore (IEC + Base 1)		Diagnóstico
Trein.	Valid.	Base 2	Trein.	Valid.	Base 2
92,2	70,3	69,8	86,5	81,5	72,8

Tabela 7.12: Índice de Concordância Percentual por tipo de defeito para a base desbalanceada.

Defeitos	Rede Neural			Árvore de decisão		
	Índice de Concordância (%)			Índice de Concordância (%)		
	Geração da Rede (IEC + Base 1)		Diagnóstico	Montagem da Árvore (IEC + Base 1)		Diagnóstico
	Trein.	Valid.	Base 2	Trein.	Valid.	Base 2
Normal	93,5	86,2	78,1	91,2	90,3	83,3
Def Elétrico	96,2	87,3	48,3	93,1	92,0	53,9
Def Térmico	84,3	41,1	3,8	75,8	64,3	11,0

Tabela 7.13: Índice de Concordância Percentual global para a base balanceada.

Rede Neural			Árvore de decisão		
Índice de Concordância (%)			Índice de Concordância (%)		
Geração da Rede (IEC + Base 1)		Diagnóstico	Montagem da Árvore (IEC + Base 1)		Diagnóstico
Trein.	Valid.	Base 2	Trein.	Valid.	Base 2
94,8	80,2	78,3	95,7	90,1	84,6

Tabela 7.14: Índice de Concordância Percentual por tipo de defeito para a base balanceada.

Defeitos	Rede Neural			Árvore de decisão		
	Índice de Concordância (%)			Índice de Concordância (%)		
	Geração da Rede (IEC + Base 1)		Diagnóstico	Montagem da Árvore (IEC + Base 1)		Diagnóstico
	Trein.	Valid.	Base 2	Trein.	Valid.	Base 2
Normal	91,2	68,0	89,0	92,7	90,3	92,2
Def Elétrico	97,1	83,6	62,6	97,1	93,6	79,9
Def Térmico	96,6	90,3	83,3	96,9	96,8	82,1

Finalmente, na quarta análise, foram realizadas simulações considerando cada amostra da base de dados (IEC, Bases 1 e 2) normalizada pelo valor do total de gases combustíveis (TGC), isto é, pela soma das concentrações dos gases da referida amostra. Neste caso, as simulações foram realizadas apenas para as bases de dados balanceadas, já que os resultados obtidos com as bases desbalanceadas se mostraram piores. Assim, foram encontrados os resultados das Tabelas 7.15 a 7.20, respectivamente, para a primeira, segunda e terceira análises anteriores.

Tabela 7.15: Índice de Concordância Percentual global para a basedes balanceada, considerando o TGC – primeira análise.

Rede Neural				Árvore de decisão			
Índice de Concordância (%)				Índice de Concordância (%)			
Geração da Rede (IEC)		Diagnóstico		Montagem da Árvore (IEC)		Diagnóstico	
Trein.	Valid.	Base 1	Base 2	Trein.	Valid.	Base 1	Base 2
98,2	81,1	66,9	69,2	98,0	84,5	77,2	73,6

Tabela 7.16: Índice de Concordância Percentual discriminado por tipo de defeito para a base balanceada, considerando o TGC – primeira análise.

Defeitos	Rede Neural				Árvore de decisão			
	Índice de Concordância (%)				Índice de Concordância (%)			
	Geração da Rede (IEC)		Diagnóstico		Montagem da Árvore (IEC)		Diagnóstico	
	Trein.	Valid.	Base 1	Base 2	Trein.	Valid.	Base 1	Base 2
Normal	100,0	100,0	63,8	79,9	100,0	100,0	67,9	81,2
Def Elétrico	95,3	58,9	67,1	56,8	96,3	61,2	87,8	57,1
Def Térmico	98,4	87,2	71,8	72,5	98,2	100,0	77,6	75,1

Tabela 7.17: Índice de Concordância Percentual global para a base balanceada, considerando o TGC – segunda análise.

Rede Neural				Árvore de decisão			
Índice de Concordância (%)				Índice de Concordância (%)			
Geração da Rede (Base 1)		Diagnóstico		Montagem da Árvore (Base 1)		Diagnóstico	
Trein.	Valid.	IEC	Base 2	Trein.	Valid.	IEC	Base 2
95,2	81,9	82,4	60,3	95,7	89,6	87,2	75,1

Tabela 7.18: Índice de Concordância Percentual por tipo de defeito para a base balanceada, considerando o TGC – segunda análise.

Defeitos	Rede Neural				Árvore de decisão			
	Índice de Concordância (%)				Índice de Concordância (%)			
	Geração da Rede (Base 1)		Diagnóstico		Montagem da Árvore (Base 1)		Diagnóstico	
	Trein.	Valid.	IEC	Base 2	Trein.	Valid.	IEC	Base 2
Normal	97,7	84,2	87,1	86,2	97,2	84,8	88,3	90,4
Def Elétrico	95,3	77,4	95,9	56,9	95,3	96,7	98,6	63,1
Def Térmico	91,1	82,6	61,5	65,4	93,9	90,2	77,3	68,7

Tabela 7.19: Índice de Concordância Percentual global para a base balanceada, considerando o TGC – terceira análise.

Rede Neural			Árvore de decisão		
Índice de Concordância (%)			Índice de Concordância (%)		
Geração da Rede (IEC + Base 1)		Diagnóstico	Montagem da Árvore (IEC + Base 1)		Diagnóstico
Trein.	Valid.	Base 2	Trein.	Valid.	Base 2
97,1	85,3	83,7	98,1	92,6	87,8

Tabela 7.20: Índice de Concordância Percentual por tipo de defeito para a base balanceada, considerando o TGC – terceira análise.

Defeitos	Rede Neural			Árvore de decisão		
	Índice de Concordância (%)			Índice de Concordância (%)		
	Geração da Rede (IEC + Base 1)		Diagnóstico	Montagem da Árvore (IEC + Base 1)		Diagnóstico
	Trein.	Valid.	Base 2	Trein.	Valid.	Base 2
Normal	95,2	81,7	91,2	100,0	90,6	93,3
Def Elétrico	98,8	84,3	73,6	96,3	95,7	82,9
Def Térmico	97,7	94,8	84,9	98,9	97,2	84,7

Os melhores resultados de cada análise foram obtidos com a base de dados balanceada. O fato da base de dados ser balanceada evita alguns dos problemas como *overfitting*, que é causado quando a rede neural ou a árvore de decisão tem bons resultados para o treinamento, mas apresentam uma generalização pobre, tendo resultados ruins para a validação. Dentre os resultados das três primeiras análises, o resultado mais eficiente encontrado foi na terceira análise com o algoritmo J4.8. Também foi nesta análise que a rede neural obteve os melhores resultados. A base de dados utilizada para gerar o classificador, ou seja, a rede neural ou a árvore de decisão, possuía uma variação maior nos dados já que foi constituída pelo agrupamento da base IEC com a Base 1.

Na quarta análise, quando a base de dados foi considerada utilizando-se o cálculo do valor da concentração do gás dividido pelo TGC, ambas as técnica de redes neurais e árvore de decisão apresentaram resultados com maior índice de concordância.

As melhores árvores de decisão geradas, tanto para a base de dados normal balanceada, quanto para a base balanceada e normalizada, estão apresentadas nas Figuras 7.1 e 7.2 seguintes.

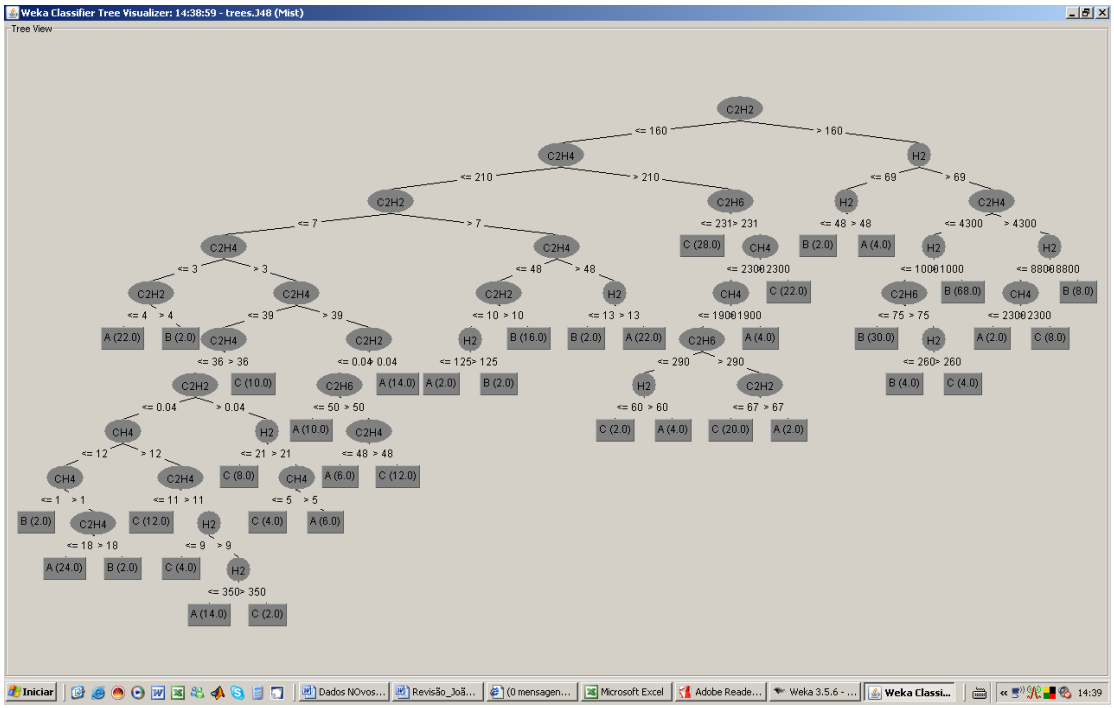


Figura 7.1: Árvore de Decisão gerada pelo algoritmo J4.8, para a terceira análise, com a base de dados normal.

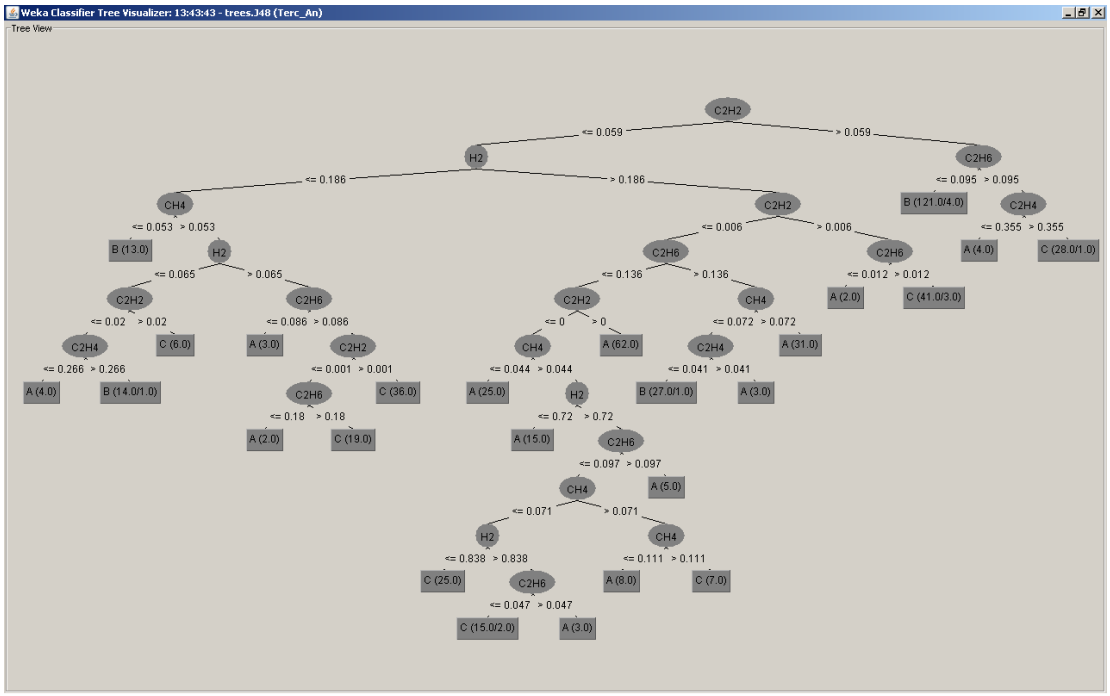


Figura 7.2: Árvore de Decisão gerada pelo algoritmo J4.8, para a terceira análise, com a base de dados normalizada com o valor de TGC.

Os piores resultados obtidos foram aqueles em que a Base 2 foi utilizada para treinamento/validação da MLP e montagem da árvore de decisão. A concentração de alguns dos gases no óleo dos transformadores desta base apresentou variações bruscas, ocorrendo até reduções, mesmo sem a apresentação de uma ação de filtragem do óleo. Uma avaliação da variação dos dados em geral para essa base mostra uma superposição grande dos dados entre 0 e 2000ppm, para todos os gases. Os dados desta base segundo as diferentes falhas são apresentados nos três gráficos a seguir, mostrando a grande superposição dos mesmos.

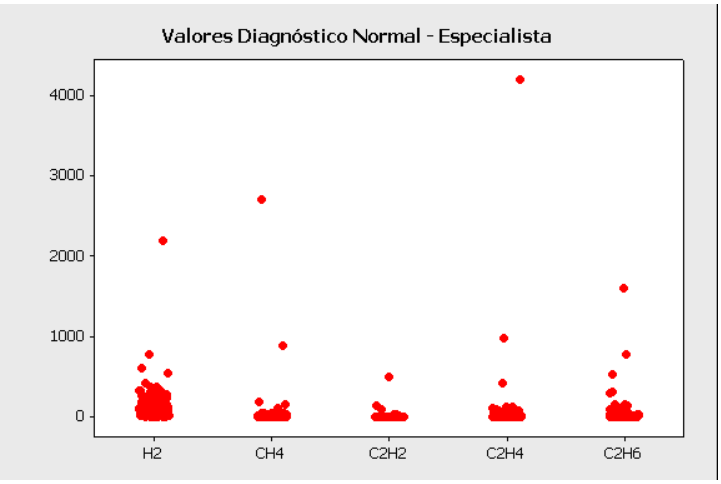


Figura 7.3: Distribuição das concentrações de gases com diagnóstico normal – Base 2.

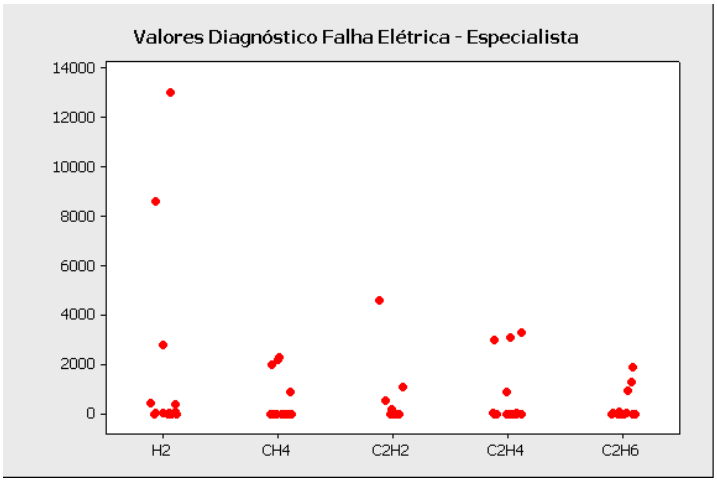


Figura 7.4: Distribuição das concentrações de gases com falha elétrica – Base 2.

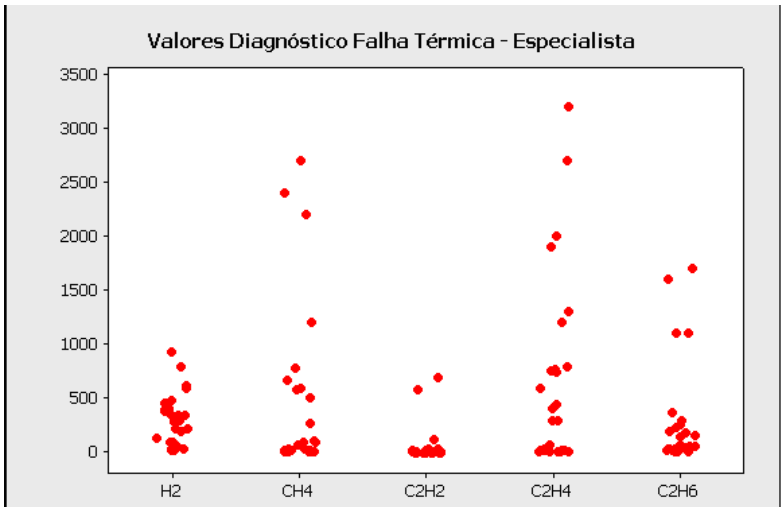


Figura 7.5: Distribuição das concentrações de gases com falha térmica – Base 2.

Outro fator é o desbalanceamento dos dados da Base 2, representado na Figura 8.4.

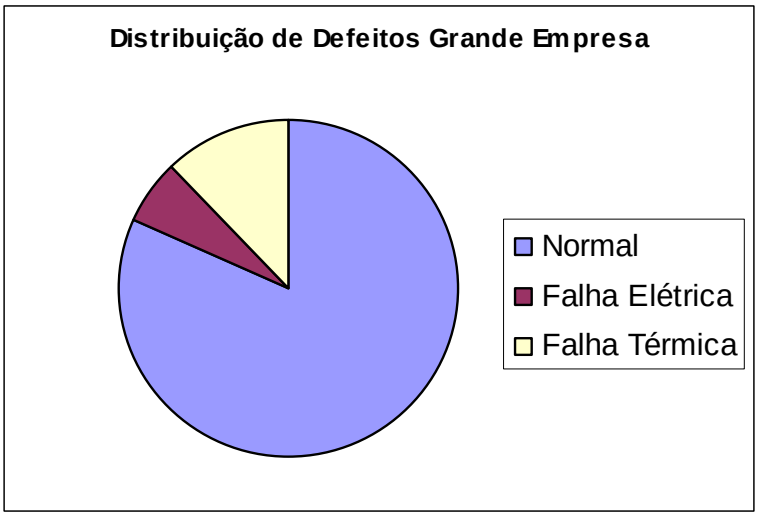


Figura 7.6: Distribuição geral das concentrações dos gases segundo os diagnósticos normal, falha elétrica e falha térmica – Base 2.

7.4 Conclusões

Neste capítulo foram apresentados os resultados das simulações para a base de dados hepatite e DNA e estes foram analisados em relação a outros resultados encontrados na

literatura. Os resultados destas simulações, aplicando redes neurais e árvores de decisão, foram compatíveis com os resultados encontrados, chegando a superá-los no caso da base de dados DNA.

Assim as ferramentas foram consideradas válidas para serem utilizadas como classificadores em outras bases de dados.

A segunda parte do capítulo traz os resultados das ferramentas aplicadas como auxílio para tomada de decisões em relação ao diagnóstico de defeitos em transformadores de potência com base nos dados de concentração de gases diluídos no óleo isolante. Foram apresentados os resultados de índice de concordância global e índice de concordância discriminado por tipo de defeito. A melhor eficiência se deu quando da utilização da ferramenta de árvore de decisão, com o algoritmo J4.8 no software *Weka*.

8 Conclusões

KDD é um processo de descoberta de conhecimento em bases de dados que tem como objetivo principal extrair conhecimento a partir de grandes bases de dados.

A etapa mais importante deste processo é a mineração de dados que se caracteriza pela existência de um algoritmo que diante da tarefa proposta será eficiente em extrair conhecimento implícito e útil de um banco de dados. Pode-se dizer que mineração de dados é a fase que transforma dados puros em informações úteis.

Neste trabalho, dois algoritmos para extração do conhecimento implícito, no auxílio à tomada de decisões em engenharia, foram analisados: o primeiro é uma rede perceptron de camadas paralelas – MLP, cujo treinamento foi efetuado através do algoritmo de retropropagação resiliente; o segundo é o algoritmo J4.8, baseado na técnica de árvores de decisão.

Os algoritmos propostos foram estudados e utilizados para classificação de três bases de dados, oriundas de testes de cromatografia em óleo isolante de transformadores de potência. A classificação foi realizada em função de defeitos de natureza elétrica ou térmica em transformadores de potência, detectados a partir das concentrações dos gases.

Inicialmente, os resultados dos algoritmos, obtidos em problemas de classificação de duas bases de dados denominadas de Hepatite e DNA, foram analisados e confrontados com resultados disponíveis na literatura. As taxas percentuais de acerto na classificação obtidos pelos dois algoritmos (MLP e J4.8) demonstraram ser equivalentes ou melhores do que aqueles encontrados na literatura. Esta análise foi utilizada para confirmar se estas ferramentas foram corretamente programadas e se poderiam ser utilizadas para a finalidade de classificação em base de dados de cromatografia, objetivando o diagnóstico de faltas incipientes (defeitos) em transformadores.

Para a base de dados hepatite os resultados das simulações com a MLP e o J4.8 foram compatíveis com os resultados encontrados em trabalhos disponíveis na literatura internacional.

Na simulação com a base de dados DNA, os resultados encontrados foram melhores que os apresentados na literatura consultada. Diversos fatores levam a este melhor resultado:

- A base de dados foi dividida em duas bases, sendo uma delas balanceada e a outra desbalanceada. Isto coloca os dados de forma mais uniforme para o algoritmo.
- As combinações de camadas de saída, épocas e neurônios utilizadas no caso das redes neurais. Neste trabalho, o resultado ótimo foi encontrado com a configuração: função de transferência logsig, com 8000 iterações e 4 neurônios.

Estes resultados obtidos validam os algoritmos escolhidos (devido a semelhança dos resultados com os já consolidados na literatura) e encorajam a sua utilização como ferramentas auxiliares na tomada de decisões na área de engenharia elétrica.

Em geral foi percebida uma eficiência maior nos resultados de diagnóstico utilizando o algoritmo J4.8.

Um fato pertinente a se discutir é a dificuldade de obtenção de dados cromatográficos organizados e com diagnósticos confirmados por medições específicas. Não se deve levar em consideração apenas os teores de concentrações instantâneos; o mais confiável seria um estudo da taxa de variação destas concentrações, sendo essa taxa essencial para a decisão de diagnóstico ou não de um determinado transformador.

Outros fatores também precisam ser considerados como, por exemplo, a migração de gases entre a celulose e o óleo do transformador de acordo com a temperatura do meio. Este fato proporciona, para o mesmo transformador, valores diferentes de teor de concentrações dos gases, de acordo com a temperatura ambiente.

O processo de mineração de dados para auxiliar na atividade de tomada de decisões na área da engenharia elétrica pode ser aplicado, com a maioria apresentando uma taxa de acerto entre 70 e 85%. Este resultado ainda pode ser melhorado se o pré-processamento nas bases de dados for realizado por especialistas com conhecimento do domínio de aplicação.

As diferenças entre os transformadores, como: volume do óleo isolante, aspectos contrutivos, classes de tensões e fatores ambientais envolvidos, aliados à incerteza nos

processos de cromatografia dos transformadores impossibilitam a obtenção de um classificador com 100% de diagnósticos corretos. Mas a combinação dos resultados obtidos com os métodos apresentados e a experiência dos especialistas aumentam a confiabilidade dos diagnósticos.

8.1 Trabalhos Futuros

Fica a sugestão de desenvolvimento de trabalhos futuros que poderiam abordar principalmente as possibilidades do pré-processamento de dados, utilizando especialistas no domínio da aplicação. Acredita-se que os resultados possam melhorar satisfatoriamente se este pré-processamento for feito em conjunto com pessoas especialistas, que teriam conhecimento para retirar ou inserir dados relacionados à aplicação.

Referências Bibliográficas

[1] AFTARCZUK, K. Evaluating of Selected Data Mining Algorithms implemented in Medical Decision Support Systems. Sweden, 2007. Dissertação (Mestrado em Engenharia de Software) – Blekinge Institute of Technology, School of Engineering.

[2] AGRAWAL, R.; SRIKANT, R. Fast Algorithms for Mining Association Rules. Chile, 2004. *Proceedings of the 20th VLDB Conference*.

[3] ALMEIDA, F.C.; DUMONTIER, P. O Uso de Redes Neurais em Avaliação de Riscos de Inadimplência, Revista de Administração FEA/USP, São Paulo, jan/mar 1996.

[4] ALMEIDA, F.C.; SIQUEIRA, J.O. *Qualidade na Decisão de Crédito: Regressão Logística e Redes Neurais na previsão de Falência de Bancos Brasileiros*. Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo. Disponível em: < http://www.fea.usp.br/Proinfo/artigos/asiq.pdf>. Acesso em: 15 de maio de 2008.

[5] ASUNCION, A.; NEWMAN, D.J. Machine Learnin Repository. Irvine, 2007. Department of Information and Computer Science, University of California. Disponível em: < http://www.ics.uci.edu/~mlearn/MLRepository.html>. Acesso em: 29 de maio de 2008.

[6] BALA, J.; HUANG, J.; VAFAIE, H.; DEJONG, K.; WECHSLER, H. Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification. Montreal, 1995. *IJCAI Conference*.

[7] BATISTA, G.E.A.P.A. Pré-Processamento de Dados em Aprendizado de Máquina Supervisionado. São Carlos, 2003. Tese (Doutorado em Ciências da Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, USP.

[8] BOSIGNOLI, R.; INFANTOSI, A.F.C. Redes Neurais Artificiais na Classificação de Estados do Ciclo sono-vigília em recém-nascidos. *Revista Brasileira de Engenharia Biomédica*. Rio de Janeiro, **18** (2): 75-87, mai/ago 2002.

[9] BRAMEIER, M.; BANZHAF, W. A Comparision of Linear Genetic Programming and Neural Networks in medical Data Mining, 2001. *Evolutionary Computation, IEEE Transactions* **5** (1):17-26.

[10] CUROTTO, C.L. Árvores de Decisão. Rio de Janeiro, 2000. Tese (Mestrado em Engenharia Civil) – Computação de Alto Desempenho e Sistemas Computacionais COPPE, Universidade Federal do Rio de Janeiro.

[11] DUVAL, M. A Review of Faults Detectable by Gas-in-Oil Analysis in Transformers. *IEEE Electrical Insulation Magazine*. Canadá, **18**, (3): 8-17, 2002.

[12] FAYYAD, U.; SHAPIRO, G.P.; SMYTH, P. From Data Mining to Knowledge Discovery in Databases. *American Association for Artificial Intelligence*, 1996.

[13] GUIZZO, Érico. Quem Procura Acha. *Revista Negócios Exame*, 2000. Disponível em: <http://www.uol.com.br/negociosexame/complementos/revista0002.html>. Acesso em: 04 maio 2007.

[14] HAN, J; KAMBER, M. *Data Mining: Concepts and Techniques*. San Diego: Academic Press, 2001.

[15] HAYKIN, S. *Redes Neurais - Princípios e Prática*. 1. ed. Bookman, 2001.

[16] HOFFMAN, P.; GRINSTEIN, G.; MARX, K., GROSSE, I.; STANLEY, E. DNA Visual and Analytic Data Mining. Lowell, [199-]. Department of Computer Science, University of Massachusetts Lowel.

[17] HORNIK, K.; STINCHCOMBE, M.; WHITE, H. *Multilayer Feed forward Networks are Universal Approximators*. California, 1989. Neural Networks **2** (5): 359-366.

[18] HUNT, E.B.; MAIN, I.; STONE, P.J. *Experiments in Induction*. New York: Academic Press, 1993.

[19] INDURKHYA, N.; WEISS, S.M. Estimating Performance Gains for Voted Decision Trees. 1999. *IBM Research Division Technical Repor in Intelligent Data Analysis (IDA).*.

[20] JACOBS, R.A. Increased Rates of Convergence Through Learning Rate Adaptation. 1987. University of Massachusetts, Technical Reprt **1** p: 295-307.

[21] KOHAVI, R.; SOMMERFIELD, D.; ODUGHERTY, J. *Data Mining Using MLC++ A Machine Learning Library in C++*. 1996. Disponível em: <http://citeseer.ist.psu.edu/kohavi97data.html>. Acesso em: Maio/07.

[22] KUBAT, M.; MATWIN, S. *Addressing the Course of Imbalanced Training sets: One Sided Selection.* University of Ottawa. Disponível em: <http://www.miami.edu/eng-electrical/mkubat/Publications/Sampling.ps>. Acesso em: Ago/07.

[23] LEMOS, E.P., Análise de Crédito Bancário com o uso de Data Mining: Redes Neurais e Árvores de Decisão, Curitiba, 2003. Tese (Mestrado em Ciências) – Departamento de Matemática, Universidade Federal do Paraná.

[24] LU, H.; SETIONO, R.; LIU, H. NeuroRule: A Connectionist Approach to Data Mining, *In: Proceedings of the 21st VLDB Conference*, 1995.

[25] MANNILA, H. Data Mining: Machine Learning, Statistics and databases. *In Proceedings of the eigth IEEE International Conference*. 1996, p 2-9.

[26] MCCULLOCH, W.S.; PITTS, W. A Logical Calculus of the Ideas Immanent in nervous activity. 1943. *Bulletin of Mathematical Biophysics*, **5**: 115-133.

[27] MORAIS, D. R. Ferramenta Inteligente para detecção de falhas incipientes em transformadores baseada na análise de gases dissolvidos no óleo isolante. Florianópolis, 2004. Tese (Mestrado em Engenharia Elétrica) – Universidade Federal de Santa Catarina.

[28] NIMER, F.; SPANDRI, L., C. *Data Mining*. Revista Developers. Fev 1998, p 32.

[29] PASSOS, M.G. Modelos de Dispositivos de Microondas e Ópticos Através de Redes Neurais Artificiais de Alimentação Direta. Natal, 2006. Tese (Mestrado em Engenharia Elétrica) – Universidade Federal do Rio Grande do Norte.

[30] POSSA, B. A. V.; CARVALHO, M. L. B. de; REZENDE, R.S.F.; MEIRA JR., W. Data Mining: Técnicas para Exploração de Dados. Universidade Federal de Minas Gerais, 1998.

[31] QUINLAN, J.C. *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann 302p, 1993.

[32] RIEDMILLER, M.; BRAUN, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. *In Proceedings of the IEEE International Conference on Neural Networks*, 1993.

[33] RODRIGUES, A.M. *Escavando Dados no varejo*. Disponível em: <http://www.centraldeestudosemlogistica.com.br/new/fs-busca.htm?fr-varejo.htm>. Acesso em: Ago/07.

[34] RONIS, H.; MONRO, S. A Stochastic Approximation Method. *In: Annals of Mathematical Statistics*, **22**, 1951.

[35] ROSENBLATT, F. The Perceptron: a probabilistic model for information storage and organization in the brain. 1958. *Psychological Review*, **65**: 386-408.

[36] SARMENTO, A. Experimentação e Avaliação de Modelos para um Problema de Atribuição de Crédito. 2005. Tese (Mestrado em Análise de Dados e Sistemas de Apoio a Decisão) – Faculdade de Economia, Universidade do Porto.

[37] SILVA, E.M. Avaliação do Estado da Arte e Produtos. Data Mining. 2000. Tese (Dissertação de Mestrado). Universidade Católica de Brasília.

[38] TAFNER, M.A. *Redes Neurais Artificiais: Aprendizado e Plasticidade*. Revista Cérebro e Mente, Universidade Estadual de Campinas. mar/mai 1998.

[39] THEARLING, Kurt; BERSON, Alex; SMITH, Stephen. *Building Data Mining Application for CRM*. McGrawHill, 488p, 1999.

[40] UYSAL, I.; GUVENIR, H.A. Instance-Based Regression by Partitioning Feature Projections. TurKey, 2004. Applied Intelligence **21**: 57-79.

[41] WANG, Z.; LIU, Y.; GRIFFIN, P. J. A Combined ANN and Expert System Tool for Transformer Fault Diagnosis. 1998. *IEEE Transactions on Power Delivery*, **13** (4): 1224-1229.

[42] WIDROW, B.; HOFF, M.E. Adaptative Switching Circuit. New York, 1960. *IRE WESCON Convention Record* p: 96-104.

[43] ZENCO, B.; TODOROVSKI, L.; DZEROSKI, S. *A comparision of stacking with MDTs to bagging, boosting, and other stacking methods*. Slovenia, Josef Stefan Institute.

[44] ZHANG, Y.; DING, X.; LIU, Y.; GRIFFIN, P.J. 1996. An Artificial Neural Network Approach to Transformer Fault Diagnosis. *IEEE Transactions on Power Delivery*, **11** (4): 1836-1841.