SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito... 19 / 3 / 99

Assinatura:.....

**Investigação de Algoritmos de
Aprendizado de Máquina
Pertencentes ao Paradigma Estatístico
para Aquisição de Conhecimento¹**

Thereza Patrícia Pereira Padilha

Orientação:

Profa. Dra. Solange Oliveira Rezende

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - USP, como parte dos requisitos para a obtenção do título de Mestre em Ciências - Área de Ciências de Computação e Matemática Computacional.

USP - São Carlos

Março de 1999

¹Trabalho realizado com auxílio do CNPq.

Este documento foi preparado com o formatador de textos L^AT_EX. O sistema de citações de referências bibliográficas utiliza o padrão *Chicago* do sistema B_IB_TE_X.

© Copyright 1999 por Thereza Patrícia Pereira Padilha
Todos os Direitos Reservados

Dedicatória

Aos meus pais, José de Barros e Terezinha,
minhas irmãs Christina e Regina e meu irmão José Alberto.

Agradecimentos

À minha orientadora Profa. Dra. Solange Oliveira Rezende, pela competente orientação, incentivo, dedicação e, principalmente, pela sua amizade, propiciando-me um grande crescimento profissional e pessoal.

À Profa Dra. Maria Carolina Monard e ao Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho, pela atenção e valiosas contribuições para a realização deste trabalho.

Ao Anderson Shiota e Marcos Ferreira de Paula, pelo auxílio e colaboração que deram a este trabalho.

Aos meus amigos (Chandler, Fábio, Luis Carlos, Augusto, Gustavo, Robson, Selma, Maju, Sandra, Sílvio, Juliana, Cris, Marco, Walter, Huei, Gedson, Ana Elisa, Marisa, Andréa, Naiara, Antônio, entre tantos outros), pelos bons momentos compartilhados. Em especial, a Jaqueline pelo precioso tempo dispensado no auxílio e revisão desta dissertação.

À todos os amigos do LABIC, pelo companherismo, amizade e união.

À Aletéia, Paulo e Alex, pelo apoio, carinho e alegria em todos os momentos, os quais nunca vou esquecer e, acima de tudo, por serem grandes amigos.

Ao Gentil Veloso Barbosa pelo carinho, incentivo e paciência.

À toda minha família, pelo amor, apoio e compreensão pelo tempo que estive ausente.

Às funcionárias da Pós-Graduação (Beth, Laura e Marília), da biblioteca, ao Paulinho, ao Jacques e à todos os funcionários do ICMC, pela maneira prestativa e educada que sempre me atenderam.

Ao CNPq pelo apoio financeiro.

Àqueles que, de um modo ou de outro, contribuíram para a realização deste trabalho.

À Deus, pela permissão e por estar sempre ao meu lado.

Sumário

1	Introdução	1
2	Aquisição de Conhecimento de Dados	4
2.1	Considerações Iniciais	4
2.2	Aprendizado de Máquina	5
2.3	Tarefas de Aprendizado de Máquina	7
2.4	Classificação dos Algoritmos de Aprendizado	11
2.4.1	Por Tipo de Aprendizado	12
2.4.2	Por Tipo de Paradigma	13
2.5	Considerações Finais	14
3	Paradigma Estatístico	16
3.1	Considerações Iniciais	16
3.2	Teoria da Probabilidade	17
3.3	Redes Bayesianas	22
3.3.1	Independência Condicional	23
3.3.2	Representação	23
3.3.3	Inferência	25
3.3.4	Aprendizado de Redes Bayesianas	26
3.4	Considerações Finais	27
4	Aprendizado Não Supervisionado e <i>Clustering</i>	29
4.1	Considerações Iniciais	29
4.2	Aprendizado Não Supervisionado	29
4.3	<i>Clustering</i>	30

4.3.1	Medidas de Similaridades	32
4.3.2	Análise de <i>Cluster</i>	36
4.3.3	Algoritmos para Análise de <i>Cluster</i>	40
4.4	Considerações Finais	42
5	Ferramentas e Algoritmos Utilizados	44
5.1	Considerações Iniciais	44
5.2	<i>MinesetTM</i>	44
5.3	<i>MLC⁺⁺</i>	49
5.4	Algoritmos Estatísticos	51
5.4.1	<i>Naive Bayes</i>	51
5.4.2	<i>AutoClass</i>	58
5.4.3	<i>AutoClass Pro (ACPro)</i>	64
5.4.4	<i>K-Means</i>	66
5.5	Considerações Finais	69
6	Avaliação Experimental de Alguns Algoritmos Estatísticos	71
6.1	Considerações Iniciais	71
6.2	Descrição dos Conjuntos de Dados Utilizados	72
6.2.1	Conjunto de Dados Iris	72
6.2.2	Conjunto de Dados Seguros	72
6.3	Investigação dos Algoritmos Utilizando o Conjunto de Dados Iris	75
6.3.1	<i>Naive Bayes</i>	75
6.3.2	<i>AutoClass</i>	79
6.3.3	<i>AutoClass Pro</i>	82
6.3.4	<i>K-Means</i>	87
6.3.5	Análise dos Resultados	89
6.4	Investigação dos Algoritmos Utilizando o Conjunto de Dados Seguros	92
6.4.1	<i>AutoClass</i>	92
6.4.2	<i>AutoClass Pro</i>	94
6.4.3	<i>K-Means</i>	97
6.4.4	Análise dos Resultados	100
6.5	Considerações Finais	102

7 Conclusões	104
Referências	107

Lista de Figuras

2.1	Sistema de Aprendizado.	5
2.2	Representação das Tarefas Básicas de Aprendizado.	6
2.3	Tarefas de Aprendizado de Máquina (Meneses & Grinstein, 1998).	8
2.4	Os Exemplos do Conjunto de Dados Empréstimo (Fayyad et al., 1996).	9
2.5	Uma Classificação Linear para o Conjunto de Dados Empréstimo (Fayyad et al., 1996).	9
2.6	Uma Regressão Linear para o Conjunto de Dados Empréstimo (Fayyad et al., 1996).	10
2.7	Um Modelo de <i>Clustering</i> para o Conjunto de Dados Empréstimo (Fayyad et al., 1996).	11
3.1	Rede Bayesiana e a Tabela de Probabilidade Condicional (Russell & Norvig, 1995).	24
3.2	Redes a) Simplesmente Conexa e b) Multiconexa (Castineira, 1996).	26
4.1	Tarefa de <i>Clustering</i>	30
4.2	Exemplos de Partição dos Dados (Mizoguchi & Shimura, 1980).	31
4.3	Uma Ilustração da Distância Euclidiana.	33
4.4	Uma Ilustração da Distância de <i>Manhattan</i>	34
4.5	Distâncias do <i>Threshold</i> d_0 a) grande b) intermediário e c) pequeno (Duda & Hart, 1973).	35
4.6	Três Exemplos de Conjuntos de Pontos no Espaço (Duda & Hart, 1973).	36
4.7	Um Dendrograma para <i>Clustering</i> Hierárquico (Duda & Hart, 1973).	38
4.8	Busca de <i>Clusters</i> com o Algoritmo Vizinho Mais Próximo (Duda & Hart, 1973).	42

5.1	Visão Geral dos Elementos Constituintes do <i>MinesetTM</i> (Félix, 1998).	45
5.2	Interface Gráfica do Software <i>MinesetTM</i>	46
5.3	Interface da Ferramenta <i>Cluster Visualizer</i>	47
5.4	Interface da Ferramenta <i>Evidence Visualizer</i>	48
5.5	Interface da Ferramenta <i>Scatter Visualizer</i>	49
5.6	Interface do Utilitário <i>Dotty</i>	51
5.7	Rede Bayesiana para o Classificador <i>Naive Bayes</i> (Dietterich, 1997).	54
5.8	Rede Bayesiana para o Algoritmo <i>AutoClass</i> (Heckerman, 1997).	60
6.1	<i>Dotty</i> do Sub-conjunto Dados1 gerado pelo <i>Naive Bayes</i>	78
6.2	<i>Evidence Visualizer</i> do Sub-conjunto Dados1 gerado pelo <i>Naive Bayes</i>	79
6.3	Relatório Parcial do Sub-conjunto Dados1 gerado pelo <i>AutoClass</i> com o Atributo-classe.	81
6.4	Relatório do Sub-conjunto Dados1 gerado pelo <i>AutoClass</i> sem o Atributo-classe.	83
6.5	<i>Cluster Viz</i> do Sub-conjunto Dados1 gerado pelo <i>ACPro</i> com o Atributo-classe.	85
6.6	<i>Scatter Visualizer</i> do Sub-conjunto Dados1 gerado pelo <i>ACPro</i> com o Atributo-classe.	86
6.7	<i>Cluster Viz</i> do Sub-conjunto Dados1 gerado pelo <i>ACPro</i> sem o Atributo-classe.	88
6.8	<i>Scatter Visualizer</i> do Sub-conjunto Dados1 gerado pelo <i>ACPro</i> sem o Atributo-classe.	89
6.9	<i>Cluster Viz</i> do Sub-conjunto Dados1 gerado pelo <i>K-Means</i> com o Atributo-classe.	91
6.10	<i>Scatter Visualizer</i> do o Sub-conjunto Dados1 gerado pelo <i>K-Means</i> com o Atributo-classe.	92
6.11	<i>Cluster Viz</i> do Sub-conjunto Dados1 gerado pelo <i>K-Means</i> sem o Atributo-classe.	94
6.12	<i>Scatter Visualizer</i> do Sub-conjunto Dados1 gerado pelo <i>K-Means</i> sem o Atributo-classe.	95
6.13	Relatório do Sub-conjunto Dados1 gerado pelo <i>AutoClass</i>	97
6.14	<i>Cluster Viz</i> do Sub-conjunto Dados1 gerado pelo <i>ACPro</i>	98
6.15	<i>Scatter Visualizer</i> gerado pelo <i>ACPro</i> com o Sub-conjunto Dados1.	99
6.16	<i>Cluster Viz</i> com o Sub-conjunto Dados1 gerado pelo <i>K-Means</i>	100
6.17	<i>Scatter Visualizer</i> do Sub-conjunto Dados1 gerado pelo <i>K-Means</i>	101

Lista de Tabelas

3.1	Distribuição Probabilística Conjunta.	19
5.1	Levantamento de Alguns Algoritmos Estatísticos.	70
6.1	Descrição dos Atributos do Conjunto de Dados Iris.	73
6.2	Número de Exemplos Parcial e Total do Conjunto de Dados Seguros.	74
6.3	Descrição dos Atributos da Relação HHOLD.	75
6.4	Matriz de Confusão com Três Atributos-Classes.	76
6.5	Matriz de Confusão e Taxas de Erro de Classificação.	77
6.6	Matriz de Confusão e Taxas de Erro dos Três Sub-conjuntos de Dados Iris. .	77
6.7	Informações dos <i>Clusters</i> Encontrados pelo <i>AutoClass</i> com o Atributo-classe.	80
6.8	Informações dos <i>Clusters</i> Encontrados pelo <i>AutoClass</i> sem o Atributo-classe.	82
6.9	Informações dos <i>Clusters</i> Encontrados pelo <i>ACPro</i> com o Atributo-classe. . .	84
6.10	Informações dos <i>Clusters</i> gerados pelo <i>ACPro</i> sem o Atributo-classe.	87
6.11	Informações dos <i>Clusters</i> Encontrados pelo <i>K-Means</i> com o Atributo-classe. .	90
6.12	Informações dos <i>Clusters</i> Identificados pelo <i>K-Means</i> sem o Atributo-classe. .	93
6.13	Síntese Geral dos Algoritmos Utilizados com o Conjunto de Dados Iris. . . .	96
6.14	Busca Automática de <i>Clusters</i> Utilizando o <i>AutoClass</i>	96
6.15	Busca Automática com os Três Sub-conjuntos Utilizando o <i>ACPro</i>	97
6.16	Explorando o <i>Cluster</i> 6 do Sub-conjunto Dados1 gerado pelo <i>ACPro</i>	98
6.17	Busca Iterativa com os Três Sub-conjuntos Utilizando o <i>K-Means</i>	99
6.18	Explorando os <i>Clusters</i> Identificados no Sub-conjunto Dados1 Seguros. . . .	100
6.19	Explorando o <i>Cluster</i> 3 do Sub-conjunto Dados1.	101
6.20	Síntese Geral dos Algoritmos Utilizados com o Conjunto de Dados Seguros. .	102

Resumo

Uma grande revolução tecnológica ocorreu nos últimos anos em diversas áreas relacionadas a ciência da computação. Um dos aspectos que mais influenciou esta revolução foi o armazenamento, o processamento e a análise de grandes quantidades de dados geradas por várias empresas e centros de pesquisas. Com isso, a incorporação de métodos e técnicas estatísticas para a aquisição de conhecimento de dados na área de Aprendizado de Máquina tem apresentado um grande crescimento.

O propósito desse trabalho é investigar alguns algoritmos de Aprendizado de Máquina pertencente ao paradigma estatístico para a aquisição de conhecimento a partir de conjuntos de dados. Nessa investigação foram estudados os algoritmos estatísticos *Naive Bayes*, *AutoClass*, *AutoClass Pro* e *K-Means*. Dois estudos de casos (um conjunto de plantas iris e um conjunto de domicílios de clientes) foram realizados verificando, entre outros, o comportamento desses algoritmos, a relevância dos atributos dos conjuntos de dados e apresentando os *clusters* encontrados nas ferramentas de visualização.

Abstract

A technological revolution has been happening in the last few years in many areas related to computer science. One of the aspects that has most influenced this revolution is the storage, processing and analysis of large quantities of data generated by various companies and research centers. All this has led to the incorporation of statistical methods and techniques for knowledge acquisition in the area of Machine Learning has shown a large growth.

The purpose of this work is to investigate some Machine Learning algorithms that belong to the statistical paradigm for knowledge acquisition from datasets. In this investigation, the statistical algorithms *Naive Bayes*, *AutoClass*, *AutoClass Pro* and *K-Means* were used. Two case studies (one with a set about iris plants and another with a set about client households) were made to verify, among other things, the behavior of these algorithms and the relevance of the attributes in the datasets, and to present the clusters found, using visualization tools.

Capítulo 1

Introdução

Uma grande revolução tecnológica ocorreu nos últimos anos em diversas áreas relacionadas à ciência da computação. Um dos aspectos que mais influenciou esta revolução foi o armazenamento, o processamento e a análise de grande quantidade de dados gerados por várias empresas e centros de pesquisas.

Diante desse contexto, os pesquisadores da área de Inteligência Artificial (IA) estão cada vez mais investindo na aquisição de conhecimento de dados com a finalidade de apresentar soluções úteis em diferentes domínios.

Esse processo é realizado através da utilização de técnicas de aquisição de conhecimento implícito ou automática, que visam adquirir conhecimento de dados e/ou informações referentes ao domínio, por meio de sistemas de Aprendizado de Máquina (AM).

Uma questão a ser ressaltada na aquisição de conhecimento é que embora existam várias técnicas e algoritmos disponíveis para analisar ou explorar um conjunto de dados, é necessário determinar o tipo de tarefa de aprendizado que o usuário deseja realizar e principalmente o tipo de conhecimento que deseja obter para então definir qual a técnica e o algoritmo que deve ser utilizado.

De acordo com o conhecimento gerado, pode-se definir as tarefas de aprendizado como sendo preditiva ou descritiva. A tarefa de predição envolve o uso de um atributo-classe (variável) do conjunto de dados para prever um valor desconhecido a respeito de uma determinada situação. Já a tarefa de descrição envolve a busca de padrões interpretáveis pelos humanos descrevendo os próprios dados, bem como informações a respeito de relacionamentos entre os dados em questão. Embora a fronteira entre essas duas tarefas não seja bem

delineada (alguns modelos preditivos podem ser descritivos e vice-versa), a distinção básica entre elas é útil para observar como os algoritmos de AM aprendem (Meneses & Grinstein, 1998).

De maneira geral, existe uma forma básica de classificar os algoritmos de Aprendizado de Máquina, a partir da forma como eles aprendem, utilizando: aprendizado supervisionado e aprendizado não supervisionado. No aprendizado supervisionado, padrões são encontrados explorando um número de exemplos conhecidos que mostram ou implicam em padrões bem definidos: baseados nesses exemplos as generalizações são formadas. No aprendizado não supervisionado, padrões de dados são encontrados a partir de alguma caracterização lógica de regularidade (Decker & Focardi, 1995).

Vários algoritmos de AM (supervisionado e não supervisionado) que trabalham com modelos matemáticos e estatísticos foram gradualmente aceitos pelos pesquisadores de IA. Dentre esses modelos destacam a teoria da probabilidade e as redes Bayesianas.

O presente trabalho propõe uma investigação de alguns algoritmos de Aprendizado de Máquina pertencentes ao paradigma estatístico para a aquisição de conhecimento. Nessa investigação são usados os algoritmos *Naive Bayes*, *AutoClass*, *AutoClass Pro* e *K-Means*. Para auxiliar a análise e a compreensão do conhecimento obtido nesses conjuntos de dados são utilizados utilitários da biblioteca *MLC++* (*Machine Learning Library in C++*) e algumas ferramentas de visualização do software *MinesetTM*. Duas avaliações experimentais foram realizadas para investigar o comportamento desses algoritmos estatísticos.

A primeira avaliação utiliza um conjunto de dados de várias espécies de plantas iris, no qual os atributos-classes são definidos. Esse conjunto de dados foi obtido no repositório de Bases de Dados para Aprendizado de Máquina da *University of California at Irvine* (*UCI*). O objetivo principal desta avaliação é investigar como os algoritmos *Naive Bayes*, *AutoClass*, *AutoClass Pro* e *K-Means* se comportam quando é utilizado um conjunto com atributos-classes conhecidos. Nessa avaliação são realizadas duas análises com os algoritmos, uma considerando o atributo-classe como um atributo qualquer no conjunto de dados e outra eliminando o mesmo da descrição dos atributos (exceto para o algoritmo *Naive Bayes*).

A segunda avaliação usa um conjunto de dados de domicílios dos clientes de uma seguradora, no qual os atributos-classes são indefinidos. Esse conjunto de dados foi preparado pelo grupo de Pesquisa de Sistema de Informação da *Swiss Life* — uma das maiores companhias do ramo de seguros em toda a Europa. Um dos objetivos dessa avaliação é

identificar as classes (*clusters*) possíveis nesse conjunto de dados utilizando os algoritmos *AutoClass*, *AutoClass Pro* e *K-Means*, além de utilizar algumas ferramentas de visualização do *MinesetTM* para tentar identificar o conhecimento dessas classes. Cabe ressaltar que os dados são preservados e por não contar com apoio de um especialista do domínio, não foi possível avaliar o conhecimento obtido.

Este trabalho está organizado da seguinte forma.

No capítulo 2 é apresentada brevemente uma visão geral de aquisição de conhecimento de dados enfatizando alguns conceitos e tarefas básicas de Aprendizado de Máquina e também mostrando algumas das classificações possíveis dos algoritmos de Aprendizado de Máquina.

No capítulo 3 são descritos alguns termos e conceitos utilizados no paradigma estatístico para a aquisição de conhecimento de dados, destacando a teoria da probabilidade e as redes Bayesianas.

No capítulo 4 são apresentadas as principais vantagens de realizar o aprendizado não supervisionado. Além disso, é apresentada uma descrição detalhada da tarefa principal do aprendizado não supervisionado (*clustering*), bem como algumas medidas de similaridade, tipos de análise de *cluster* e um resumo dos principais algoritmos para análise de *cluster*.

No capítulo 5 são apresentados, de uma forma geral, o software *MinesetTM*, a biblioteca *MLC++*, e os algoritmos estatísticos *Naive Bayes*, *AutoClass*, *AutoClass Pro* e *K-Means* que são as ferramentas e os algoritmos utilizados nesse trabalho. E por fim, é apresentado um levantamento de algumas informações sobre alguns algoritmos estatísticos.

No capítulo 6 são apresentadas algumas avaliações experimentais realizadas com os algoritmos *Naive Bayes*, *AutoClass*, *AutoClass Pro* e *K-Means* utilizando dois conjuntos de dados: um de plantas iris e o outro de domicílios de clientes de uma determinada seguradora.

No capítulo 7 são expostas as conclusões deste trabalho e algumas sugestões para trabalhos futuros.

Finalmente, é apresentada a bibliografia que foi utilizada para que a realização deste trabalho fosse possível.

Capítulo 2

Aquisição de Conhecimento de Dados

2.1 Considerações Iniciais

Nos últimos 10 anos, três fatores têm mudado o panorama de análise de dados. Primeiro, a disponibilidade de grande quantidade de processamento computacional a um baixo custo. Segundo, o acúmulo de dados numa taxa explosiva, sendo esse acúmulo consequência do baixo custo de armazenamento combinado com o alto nível de automação de experimentos e a introdução de dispositivos eletrônicos. E o terceiro fator a introdução de novas tecnologias de processamento de dados (Fayyad et al., 1996).

Apesar das mudanças com relação à análise de dados, ainda existe uma grande necessidade de novas teorias computacionais e ferramentas para auxiliar a aquisição de conhecimento de dados.

A aquisição de conhecimento a partir dos dados tem como objetivo revelar informações “escondidas” em dados e também auxiliar a capacidade de análise, facilitando assim um processo de tomada de decisão (Wirth & Reinartz, 1996).

Com o objetivo de dar uma visão geral sobre a aquisição de conhecimento de dados, este capítulo apresenta alguns termos, conceitos e algumas tarefas da área de Aprendizado de Máquina. Essas tarefas tem como objetivo definir a forma de realizar aquisição de conhecimento a partir dos dados. É apresentado também duas formas para classificar os sistemas de aprendizado. A primeira com relação a maneira que eles aprendem: aprendizado supervisionado e não supervisionado e a segunda com relação ao paradigma de aprendizado.

2.2 , Aprendizado de Máquina

Numa visão geral, a aquisição de conhecimento de dados trabalha com métodos para solução de problemas que procuram uma descrição matemática ou lógica, eventualmente de natureza complexa, de padrões e regularidades em um determinado conjunto de dados (Decker & Focardi, 1995). Esses métodos consistem em aplicações de sistemas de Aprendizado de Máquina produzindo uma classificação de padrões (ou modelos) sobre os dados.

Sistema de aprendizado é um programa de computador que toma decisões baseado em experiências acumuladas contidas em exemplos resolvidos com sucesso (Weiss & Kulikowski, 1991). Como ilustrado na Figura 2.1, o objetivo principal de um sistema de aprendizado está em extrair conhecimento a partir de um conjunto de dados.

Figura 2.1: Sistema de Aprendizado.

Um sistema de aprendizado típico é projetado para trabalhar sob algum paradigma de aprendizado, tal como: simbólico, estatístico, conexionista, etc. Além disso, as duas tarefas básicas dos sistemas de aprendizado, predição e descrição, podem ser resumidas em encontrar uma função que mapeie exemplos com características comuns.

Na Figura 2.2 a) e b) são visualizados os elementos representativos de um conjunto de dados, e as relações entre o sistema de aprendizado, o paradigma e uma determinada tarefa de aprendizado. Cada observação do conjunto de dados, relevante à solução do problema, é denominado de atributo (variável).

Quando o sistema de aprendizado recebe um conjunto de dados com os valores de atributos, classes definidas (atributo-classe) e um determinado paradigma, o sistema fornece um preditor específico à uma aplicação. A forma mais simples de representar um preditor é como uma caixa preta que produz uma decisão de cada exemplo que foi apresentado a ele. Na Figura 2.2 c) é ilustrada uma estrutura simples de um sistema de predição que aceita

um exemplo novo como entrada e produz uma decisão da classe como saída.

Já nos casos que o sistema de aprendizado recebe um conjunto de dados com apenas os valores dos atributos e algum paradigma, o sistema fornece um descritor particular a uma aplicação. A maneira mais simples de representar um descritor seria como uma caixa preta que fornece uma descrição de cada exemplo que foi apresentado a ele. Na Figura 2.2 d) é ilustrada uma estrutura simples de um sistema de descrição que recebe um exemplo novo como entrada e produz uma descrição da classe como saída.

Figura 2.2: Representação das Tarefas Básicas de Aprendizado.

De maneira geral, a aquisição de conhecimento de dados envolve pesquisas relacionadas com algumas áreas e sub-áreas, tais como: Aprendizado de Máquina, Reconhecimento de Padrões, Base de Dados, Estatística e Visualização de Dados. Alguns dos principais

problemas relacionados a essa aquisição são:

- Dados com ruídos.
- Valores de atributos ausentes.
- Pré-processamento dos dados de entrada.
- Escolha de um algoritmo de Aprendizado de Máquina.
- Entendimento da saída do algoritmo.

Dentre os problemas relacionados com a aquisição de conhecimento de dados, a escolha de um sistema de aprendizado é um fator de fundamental importância. Nesta seção foi discutido as tarefas de aprendizado básicas (predição e descrição mostradas nas Figuras 2.1 e 2.2), a seguir serão vistas como essas tarefas funcionam mais detalhadamente.

2.3 Tarefas de Aprendizado de Máquina

Como discutido anteriormente, a tarefa de predição envolve o uso de alguns atributos de um conjunto de dados para prever um valor futuro de uma outra variável, ou seja, essa tarefa visa principalmente apoio à tomada de decisões. Já a tarefa de descrição procura padrões interpretáveis pelos humanos, no qual descreve os dados antes de prever algo. Essa tarefa visa o suporte à decisão (Meneses & Grinstein, 1998).

Ambas as tarefas, predição e descrição, são gerais e complementares e utilizam alguma das tarefas primárias de aprendizado, tais como: classificação, regressão, *clustering*, sumarização, modelagem de dependências, análise de links, análise sequencias e outras, que podem ser visualizadas na Figura 2.3.

Para ilustrar as idéias básicas dessas tarefas será utilizado o conjunto de dados empréstimos. Esse conjunto consiste de 23 exemplos e está representado num espaço bidimensional como pode ser visto na Figura 2.4. Cada ponto no espaço representa um cliente que obteve um empréstimo num determinado banco a um tempo atrás. O eixo horizontal representa o salário do cliente; e o eixo vertical representa a dívida total do cliente (hipoteca, pagamentos do carro e etc.). Os dados estão divididos em duas classes: os clientes representados com “x” tem faltado com o seu empréstimo e os clientes representados com “o” estão com um status bom de empréstimo com o banco (Fayyad et al., 1996).

Figura 2.3: Tarefas de Aprendizado de Máquina (Meneses & Grinstein, 1998).

A tarefa de classificação é uma função de aprendizado que mapeia (classifica) dados de entrada, ou conjuntos de dados de entrada, num número de categorias possíveis. Alguns exemplos dessa tarefa são: aplicações financeiras e identificação de objetos. Na Figura 2.5 é visualizado o efeito da classificação considerando o conjunto de dados empréstimo que mostra um simples particionamento em duas categorias possíveis. Note que não é possível separar as duas classes perfeitamente usando um limite de decisão linear (Fayyad et al., 1996).

A tarefa de regressão é uma função de aprendizado que mapeia dados de entrada, ou conjuntos de dados de entrada para um valor real variável. Alguns exemplos dessa tarefa são: prever a quantidade de biomassa presente numa floresta e a demanda de consumidores para novos produtos. Na Figura 2.6 é ilustrado o efeito da regressão considerando o conjunto de dados empréstimo que apresenta uma simples regressão linear, na qual a dívida total é ajustada como uma função linear dos salários (Fayyad et al., 1996).

Clustering é uma tarefa descritiva que procura identificar um conjunto finito de classes

Figura 2.4: Os Exemplos do Conjunto de Dados Empréstimo (Fayyad et al., 1996).

Figura 2.5: Uma Classificação Linear para o Conjunto de Dados Empréstimo (Fayyad et al., 1996).

ou *clusters* a partir dos dados. Os *clusters* podem ser mutuamente exclusivos e exaustivos ou consistir de uma representação valiosa, tal como: sobreposição de *clusters*. Alguns exemplos de *clustering* são: a descoberta de subpopulações homogêneas de consumidores e a identificação de subcategorias. Considerando o conjunto de dados empréstimo pode ser visualizado o efeito de *clustering* na Figura 2.7. Esse modelo de *clustering* mostra três *clusters* possíveis, e que esses *clusters* estão sobrepostos, e também possibilita que alguns dados pertençam a mais de um *cluster*. Como a tarefa de *clustering* não necessita da definição das classes do conjunto de dados original (“x” e “o”) então todas as classes foram substituídas por “+” (Fayyad et al., 1996).

A sumarização envolve métodos para encontrar uma descrição compacta para um

Figura 2.6: Uma Regressão Linear para o Conjunto de Dados Empréstimo (Fayyad et al., 1996).

subconjunto de dados. Um exemplo de sumarização é a definição da média e do desvio padrão de todos os campos. Os métodos mais sofisticados envolvem a derivação de regras resumidas e as técnicas de visualização (Fayyad et al., 1996).

A modelagem de dependências consiste em encontrar um modelo que descreva as dependências mais significantes entre as variáveis. Existem dois níveis de modelos de dependências: (1) o nível estrutural que especifica o modelo (frequentemente em forma gráfica), no qual as variáveis são dependentes localmente e (2) o nível quantitativo do modelo que especifica o peso das dependências usando uma escala numérica. Alguns exemplos da modelagem de dependências são: sistemas especialistas para diagnósticos e recuperação de informações (Fayyad et al., 1996).

A análise de links consiste em extrair correlações (*fitting*) dos atributos de um conjunto de dados. Os dados “linkados” são tipicamente modelados por um gráfico contendo nós que representam as entidades de interesse e os links propriamente ditos que representam os relacionamentos e as relações. A análise de links é distinta das técnicas que constrói as redes neurais, redes bayesianas e as regras de associação. Essas técnicas descobrem e representam associações baseadas em características estatísticas de um conjunto de instâncias. Em contraste, análise de links inicia com dados que podem ser representados como uma rede e infere conhecimento útil a partir dos nós e links da rede. Um exemplo dessa tarefa é a coleção de dados de telefones (número, tempo e duração) para uma investigação criminal (Goldberg, 1998).

Figura 2.7: Um Modelo de *Clustering* para o Conjunto de Dados Empréstimo (Fayyad et al., 1996).

A análise sequencial tem como finalidade auxiliar a detecção de ocorrência de padrões sequenciais num fluxo de identificação de categorias descrevendo as relações. Um exemplo dessa tarefa é o controle de um manufaturamento (Dietterich, 1996).

Em resumo, a meta da aquisição de conhecimento de dados depende do tipo de tarefa que o usuário deseja executar e das características dos conjuntos de dados (valores de atributos e classes definidas). Existe uma variedade de sistemas de aprendizado, cada um com as suas vantagens e suas desvantagens. Embora, não seja preciso entender profundamente seu funcionamento é importante conhecer de um modo geral como os sistemas funcionam e que tipo de saídas eles geram. Perante uma grande quantidade de sistemas de aprendizado desenvolvidos, é relevante usar uma classificação dos mesmos para um melhor entendimento do seu domínio de aplicação. A seguir são discutidas algumas formas para a classificação desses sistemas.

2.4 Classificação dos Algoritmos de Aprendizado

Classificar os algoritmos de Aprendizado de Máquina não é uma tarefa fácil, pois existem vários métodos para agrupá-los. Neste trabalho foi enfatizado dois métodos de classificação: o primeiro dependendo do tipo de aprendizado, e o segundo de acordo com um determinado paradigma de Aprendizado de Máquina.

2.4.1 Por Tipo de Aprendizado

Em geral, existem dois caminhos para classificar os algoritmos de Aprendizado de Máquina com relação a forma que eles aprendem: aprendizado supervisionado e aprendizado não supervisionado. Observando novamente a Figura 2.2, pode-se verificar que o aprendizado supervisionado está representado pelo lado esquerdo da figura enquanto que o aprendizado não supervisionado está representado pelo lado direito desta figura.

No aprendizado supervisionado, padrões são encontrados explorando um número de casos conhecidos que mostram ou implicam em padrões bem definidos: baseados nesses casos as generalizações são formadas. Alguns sistemas de aprendizado supervisionado conhecidos são: CN2 (Clark & Niblett, 1989), C4.5 (Quinlan, 1993), *Naive Bayes* (Mitchell, 1997) e FOIL (Quinlan & Carmeron-Jones, 1993).

É importante salientar que a palavra classificação é ambígua. Na maioria das vezes, ela significa associar um novo exemplo (objeto) a um conjunto existente de classes possíveis (aprendizado supervisionado). O outro significado de classificação tem como finalidade procurar suas próprias classes a partir de um conjunto de exemplos (objetos) não classificados (aprendizado não supervisionado) (Cheeseman et al., 1990).

No aprendizado não supervisionado, padrões de dados são encontrados a partir de alguma caracterização lógica de regularidades (Decker & Focardi, 1995). Para alguns autores, esse tipo de aprendizado é também chamado de *clustering*¹ (Cheeseman et al., 1990; Decker & Focardi, 1995).

Sistemas de aprendizado não supervisionado procuram uma organização conveniente e válida dos dados para futura separação dos mesmos em categorias. Nesse caso, os sistemas de aprendizado buscam nos dados uma tentativa de criar partições que “casam” com atributos descritos.

Os algoritmos não supervisionados buscam “regularidades” através de alguns atributos até chegar a um conceito, como por exemplo, a tarefa de identificação de classes. Uma das dificuldades desses algoritmos é encontrar técnicas para descobrir essas regularidades, que variam entre as diversas aplicações (Ester et al., 1996). Entre as técnicas existentes, destacam-se as estatísticas. Alguns sistemas de aprendizado não supervisionado mais co-

¹Nesse trabalho, os termos aprendizado não supervisionado e *clustering* são usados de forma distinta. *Clustering* é uma tarefa do aprendizado não supervisionado.

nhedidos são: *AutoClass* (Hanson et al., 1991), *Vizinho Mais Próximo* (Hastie & Tibshirani, 1995) e *K-Means* (Hartigan & Wong, 1978).

2.4.2 Por Tipo de Paradigma

Os algoritmos de Aprendizado de Máquina utilizados para obter padrões ou modelos seguem a um determinado paradigma de aprendizado. Dentre os vários paradigmas de aprendizado, os principais são brevemente discutidos a seguir:

Simbólico

Os sistemas de aprendizado simbólico tentam aprender construindo representações simbólicas de um conceito através de análises de exemplos. As representações simbólicas mais conhecidas são: expressão lógica, árvore de decisão, regras de produção, redes semânticas, entre outras (Morgan & Messenger, 1973; Quinlan, 1987). Os métodos de aprendizado simbólico podem ser tratados como:

- **Proposicionais:** estes métodos trabalham com enunciados declarativos, chamados de proposições (ou sentenças) e utilizam apenas as sentenças que têm valores verdadeiro ou falso. É possível construir sentenças compostas através de uso de conectivos lógicos do tipo *e* ($\wedge$), *ou* ($\vee$), *não* ($\neg$), *condicional* ($\rightarrow$), entre outros.
- **Relacionais:** uma das representações deste método é a Programação Lógica Indutiva (PLI) que é orientada a construir programas lógicos – restritos a cláusulas Horn. De maneira geral, a PLI pode ser definida como o uso da lógica simbólica para a representação explícita de problemas.

Estatístico

As técnicas estatísticas geralmente procuram focalizar tarefas em que todos os atributos têm valores contínuos ou ordinais. Muitos deles também são paramétricos, assumindo algum modelo, e então encontrando valores apropriados para os parâmetros do modelo a partir de dados (Aitkin et al., 1981). Neste trabalho será investigado o paradigma estatístico.

Conexionista

As redes neurais são construções matemáticas relativamente simples que utilizam o mecanismo de paralelismo, conectando um grande número de pequenas unidades de processamento ligadas em rede. Essas unidades são modeladas como os neurônios e foram inspirados no modelo biológico do sistema nervoso. Sua representação envolve unidades altamente interconectadas, no qual o nome “conexionismo” é utilizado para descrever a área de estudo (Haykin, 1994).

Instance-based

Uma forma de classificar um caso é lembrar de um caso similar cuja classe é conhecida e assumir que o novo caso terá a mesma classe. Essa filosofia exemplifica os sistemas instance-based, que classificam casos nunca vistos através de casos similares conhecidos (Quinlan, 1988).

Genético

Este paradigma é derivado do modelo evolucionário de aprendizado (Holland et al., 1986). Um classificador genético consiste de uma população de elementos de classificação que competem para fazer uma previsão. Elementos que possuem um desempenho fraco são descartados, enquanto que os elementos mais fortes proliferam, produzindo variações de si mesmos. Este paradigma possui uma analogia direta com a teoria de Darwin, na qual sobrevivem os elementos mais bem adaptados ao ambiente.

2.5 Considerações Finais

É importante ressaltar que a aquisição de conhecimento de dados visa extrair padrões embutidos em conjuntos de dados utilizando principalmente algoritmos de Aprendizado de Máquina. Embora, a aquisição de conhecimento de dados tem sido utilizada para uma grande variedade de tarefas de aprendizado, pode se dividir em duas tarefas principais: predição e descrição. O tipo de tarefa de aprendizado desempenhado pelos algoritmos depende da definição do problema pelo usuário e das características dos conjuntos de dados.

Os algoritmos podem ser classificados por diferentes métodos. Neste capítulo foi ressaltado a classificação com relação a maneira que eles aprendem: aprendizado supervisionado e aprendizado não supervisionado, com relação a um determinado paradigma de aprendizado.

O ponto central deste trabalho é investigar alguns dos algoritmos de Aprendizado de Máquina do paradigma estatístico, tanto no aprendizado supervisionado quanto no aprendizado não supervisionado.

No próximo capítulo será dada uma visão geral do paradigma estatístico para aprendizado bayesiano, incluindo alguns conceitos da teoria da probabilidade e redes bayesianas.

Capítulo 3

Paradigma Estatístico

3.1 Considerações Iniciais

A área de Estatística fornece um conjunto de princípios e métodos para planejar a coleta de dados, sumarizar e interpretar os dados e também extrair algumas conclusões. A utilização desses métodos estatísticos está relacionada com várias áreas da ciência e tecnologia (Achcar & Rodrigues, 1998).

Pesquisadores estatísticos têm desenvolvido diversos métodos de classificação, muitos deles semelhantes aos métodos empregados em Aprendizado de Máquina. Quando se trabalha nesse paradigma, a maioria das informações não possui respostas categóricas. Consequentemente, o agente (ser humano) trabalha sobre as incertezas (riscos) de um domínio. Segundo Russel, a incerteza surge por causa do entendimento incompleto e/ou incorreto de um agente sobre as propriedades de um determinado ambiente (Russell & Norvig, 1995). A manipulação desse conhecimento incerto é bastante utilizada em diagnósticos.

Nesses casos, o conhecimento de um agente sobre um determinado domínio fornece apenas um grau de crença sobre as sentenças. As ferramentas que manipulam os graus de crença são a teoria da probabilidade e as redes Bayesianas.

Os métodos de aprendizado Bayesiano são relevantes para o estudo de Aprendizado de Máquina por duas razões. A primeira é que os algoritmos de aprendizado Bayesiano que calculam a probabilidade explícita para as hipóteses, tal como o classificador *Naive Bayes* (detalhado na seção 4.3.3), estão entre as abordagens mais práticas para alguns tipos de problemas de aprendizado. A segunda razão é que os métodos de aprendizado Bayesiano

forneem uma perspectiva útil para o entendimento de muitos algoritmos de aprendizado que não manipulam probabilidades explicitamente (Mitchell, 1997).

Neste capítulo são apresentados brevemente alguns termos e conceitos utilizados no paradigma estatístico, discutindo a necessidade de representar e manipular incertezas em domínios cujos dados são imprecisos ou ausentes. Para isso, na seção 3.2 é apresentada a teoria da probabilidade e na seção 3.3 são apresentadas as redes Bayesianas.

3.2 Teoria da Probabilidade

Teoria da probabilidade é uma linguagem apropriada para descrever incertezas e para realizar inferências a partir de conhecimento incompleto. Seus métodos são aplicáveis a toda tarefa que requerer inferência a partir de informações incompletas (Filho, 1995; Francisco, 1993).

Essa teoria associa um grau numérico de crença entre os valores 0 (zero) e 1 (um). Quando uma probabilidade é igual a 0 corresponde a inequívoca crença que ela é falsa. Enquanto que uma probabilidade igual a 1 corresponde a inequívoca crença que ela é verdadeira. As probabilidades que possuem valores entre 0 e 1 correspondem a um grau de crença intermediário de verdade da sentença (Russell & Norvig, 1995).

É importante distinguir os termos grau de crença e grau de verdade. Por exemplo, a previsão de um time X ser o vencedor do campeonato. Na primeira interpretação, o termo grau de crença, significa que um determinado torcedor possua um grau de crença de 60% que o time X seja o campeão. Já a segunda interpretação, grau de verdade significa que baseados em campeonatos passados (arquivos), o time X tem uma chance de 10% de vencer o campeonato (Russell & Norvig, 1995).

Uma declaração probabilística indica uma informação que se refere à probabilidade que está sendo avaliada. Quando se recebe novas percepções, a avaliação probabilística deve ser atualizada para refletir em novas evidências.

Antes da evidência ser obtida, tem-se a probabilidade incondicional ou anterior. A probabilidade incondicional é usada na ausência de outra evidência e sua notação é dada por: $P(A)$, que representa a probabilidade de um evento A ocorrer. Por exemplo, a probabilidade de um paciente ter torcicolo é $1/20$, ou seja, $P(\text{torcicolo}) = 1/20 = 0,05$.

Depois da evidência ser obtida, tem-se a probabilidade condicional ou posterior. A probabilidade condicional é utilizada quando se tem alguma evidência no domínio e sua no-

tação é dada por: $P(A|B)$, que representa a probabilidade condicional de um evento A dado a ocorrência de um evento B. Por exemplo, a probabilidade de um paciente ter torcicolo dado que tenha meningite é 0,5, ou seja, $P(\text{torcicolo}|\text{meningite}) = 0,5$. A probabilidade incondicional é um caso especial da probabilidade condicional — $P(A|)$ — no qual a probabilidade está condicionada a nenhuma evidência.

A probabilidade condicional pode ser definida com base na probabilidade incondicional, como mostra a Equação 3.1:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} \quad (3.1)$$

Assegurando que $P(B) > 0$, a Equação 3.1 pode ser reescrita como a Equação 3.2:

$$P(A \wedge B) = P(A|B)P(B) \quad (3.2)$$

A Equação 3.2 é chamada de regra do produto. Essa regra pode também ser escrita como mostra a Equação 3.3, que é lida como: “para que A e B sejam verdadeiros, é necessário que A seja verdadeiro e então B ser verdadeiro dado A”.

$$P(A \wedge B) = P(B|A)P(A) \quad (3.3)$$

Entretanto, em algumas situações pode ser necessário encontrar outras probabilidades como, por exemplo, $P(A|B)$. Uma maneira de calcular essa probabilidade seria através de alguma forma de inferência probabilística.

A inferência probabilística pode ser realizada através da Distribuição Probabilística Conjunta (DPC), baseada num conjunto de variáveis do domínio. Numa visão geral, a DPC determina as probabilidades de todas as variáveis e pode ser representada por uma tabela n -dimensional.

Por exemplo, um diagnóstico médico com duas variáveis booleanas: torcicolo e meningite, a Distribuição Probabilística Conjunta pode ser observada na Tabela 3.1.

Pode também ser observado, que a partir da DPC existem várias maneiras de encontrar outras probabilidades do domínio, tal como, a probabilidade incondicional de uma variável, que pode ser determinada somando uma linha ou uma coluna. Por exemplo, $P(\text{torcicolo}) = 0,00001 + 0,04999 = 0,05$, mas pode também determinar probabilidades envolvendo alguns operadores lógicos, como: $P(\text{torcicolo} \vee \text{meningite})$.

	meningite	$\neg$ meningite
torcicolo	0,00001	0,04999
$\neg$ torcicolo	0,00001	0,94999

Tabela 3.1: Distribuição Probabilística Conjunta.

Por definição $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$, então o cálculo dessa probabilidade pode ser feito da seguinte forma:

$$P(\text{torcicolo} \vee \text{meningite}) = P(\text{torcicolo}) + P(\text{meningite}) - P(\text{torcicolo} \wedge \text{meningite})$$

$$P(\text{torcicolo} \vee \text{meningite}) = 0,05 + 0,00002 - 0,00001$$

$$P(\text{torcicolo} \vee \text{meningite}) = 0,05001$$

É importante ressaltar que em aplicações reais, o número de variáveis é bastante grande e conseqüentemente não é muito prático armazenar uma tabela com 2^n células, no qual n é o número de variáveis aleatórias.

No entanto, todas as probabilidades de um domínio podem ser encontradas, sem a necessidade de armazená-las, usando o teorema ou regra de Bayes. Essa regra pode ser obtida a partir das duas formas da regra do produto (Equações 3.2 e 3.3), igualando e dividindo essas equações por $P(A)$, como mostra a Equação 3.4:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \quad (3.4)$$

A Equação 3.4 permite encontrar a probabilidade $P(B|A)$ a partir das probabilidades $P(A|B)$, $P(B)$ e $P(A)$. No exemplo do diagnóstico médico é conhecido as probabilidades:

$$P(\text{torcicolo}) = 0,05$$

$$P(\text{meningite}) = 0,00002$$

$$P(\text{torcicolo}|\text{meningite}) = 0,5$$

Se houver a necessidade de calcular a probabilidade $P(\text{meningite}|\text{torcicolo})$, é possível aplicando-se apenas a regra de Bayes.

$$P(\text{meningite}|\text{torcicolo}) = \frac{P(\text{torcicolo}|\text{meningite})P(\text{meningite})}{P(\text{torcicolo})}$$

$$P(\text{meningite}|\text{torcicolo}) = \frac{0,5 \times 0,00002}{0,05}$$

$$P(\text{meningite}|\text{torcicolo}) = 0,0002$$

Pode-se observar que a utilização da regra de Bayes para calcular as probabilidades de um determinado domínio reduz o esforço computacional e o número de probabilidades condicionais. Essa regra representa a base da maioria dos sistemas de Inteligência Artificial para inferência probabilística (Russell & Norvig, 1995; Vapnik, 1995).

Nas aplicações de Aprendizado de Máquina, o objetivo principal é determinar a “melhor hipótese” h , a partir de algum espaço H de hipóteses e um conjunto de dados de treinamento D (Mitchell, 1997). Uma maneira de especificar a “melhor hipótese” é encontrar a hipótese “mais provável”, dado um conjunto de dados D e um conhecimento inicial sobre a probabilidade incondicional das várias hipóteses no espaço H . A hipótese mais provável é representada por h_{MAP} (*maximum a posteriori* - hipótese condicional máxima) e pode ser determinada pela regra de Bayes, usada para calcular a probabilidade condicional de cada hipótese candidata à mais provável, como descrita na Equação 3.5.

$$\begin{aligned} h_{MAP} &= \operatorname{argmax}_{h \in H} P(h|D) \\ h_{MAP} &= \operatorname{argmax}_{h \in H} \frac{P(D|h)P(h)}{P(D)} \end{aligned} \quad (3.5)$$

Como o termo $P(D)$ é uma constante independente de h , esse termo pode simplesmente ser eliminado da Equação 3.5, que pode ser reescrita como a Equação 3.6:

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(D|h)P(h) \quad (3.6)$$

Além disso, em alguns casos, pode-se assumir que qualquer hipótese candidata é, a priori, igualmente provável, ou seja, $P(h_i) = P(h_j)$ para todo h_i e h_j pertencentes a H . Deste modo, para o cálculo da hipótese mais provável, pode-se considerar apenas o termo $P(D|h)$ — probabilidade do conjunto D dado h , representada por h_{ML} (*maximum likelihood* - hipótese de máxima verossimilhança) na Equação 3.7.

$$h_{ML} = \operatorname{argmax}_{h \in H} P(D|h) \quad (3.7)$$

Para ilustrar o cálculo da hipótese mais provável, considere um problema de diagnóstico médico no qual existam duas hipóteses possíveis (Mitchell, 1997): o paciente tem câncer e o paciente não tem câncer. Os dados do paciente são obtidos a partir de um teste de laboratório, que possui dois resultados possíveis: positivo e negativo. Sabe-se também que apenas 0,008 da população sofre dessa doença. Além disso, os testes de laboratório apresentam acerto em 98% nos casos positivos e 97% nos casos negativos de câncer. Esta situação pode ser resumida pelas seguintes probabilidades:

- As probabilidades incondicionais são:

$$P(\text{câncer}) = 0,008$$

$$P(\neg\text{câncer}) = 0,992$$

- As probabilidades condicionais são:

$$P(\text{positivo}|\text{câncer}) = 0,98$$

$$P(\text{negativo}|\text{câncer}) = 0,02$$

$$P(\text{positivo}|\neg\text{câncer}) = 0,03$$

$$P(\text{negativo}|\neg\text{câncer}) = 0,97$$

Com base nessas probabilidades, qual seria o diagnóstico para um paciente cujo teste de laboratório deu positivo? A h_{MAP} pode ser calculada através da Equação 3.6.

$$P(\text{positivo}|\text{câncer})P(\text{câncer}) = 0,98 \times 0,008 = 0,0078$$

$$P(\text{positivo}|\neg\text{câncer})P(\neg\text{câncer}) = 0,03 \times 0,992 = 0,0298$$

Portanto, $h_{MAP} = \neg\text{câncer}$. O cálculo da probabilidade condicional de um paciente ter câncer dado o resultado do teste positivo pode ser calculada normalizando a probabilidade encontrada pelo h_{MAP} . Essa situação pode ser vista na Equação 3.8. É importante lembrar que a soma de $P(\text{câncer}|\text{positivo})$ e $P(\neg\text{câncer}|\text{positivo})$ deve ser igual a 1.

$$\begin{aligned} P(\text{câncer}|\text{positivo}) &= \frac{P(\text{positivo}|\text{câncer})}{P(\text{positivo}|\text{câncer}) + P(\text{positivo}|\neg\text{câncer})} \\ P(\text{câncer}|\text{positivo}) &= \frac{0,0078}{0,0078 + 0,0298} \doteq 0,21 \end{aligned} \quad (3.8)$$

O cálculo de $P(\text{câncer}|\text{positivo})$ pode também ser obtido por intermédio da regra de Bayes (descrita na Equação 3.4). Essa situação pode ser ilustrada pela Equação 3.9. Esta forma de adquirir $P(\text{câncer}|\text{positivo})$ é mais complexa, pois não foi especificado no problema o valor de $P(\text{positivo})$.

$$P(\text{câncer}|\text{positivo}) = \frac{P(\text{positivo}|\text{câncer})P(\text{câncer})}{P(\text{positivo})}$$

$$P(\text{positivo}) = P(\text{positivo}|\text{câncer})P(\text{câncer}) + P(\text{positivo}|\neg\text{câncer})P(\neg\text{câncer})$$

$$P(\text{positivo}) = 0,0376$$

$$P(\text{câncer}|\text{positivo}) = \frac{0,98 \times 0,008}{0,0376} \doteq 0,21 \quad (3.9)$$

Apesar da teoria da probabilidade ser a principal ferramenta para manipular incertezas, na prática os cálculos necessários para descrever todas as condições probabilísticas têm mostrado um alto custo computacional. Com a finalidade de minimizar essa dificuldade surgiu a abordagem probabilística baseada em redes Bayesianas.

3.3 Redes Bayesianas

Uma rede Bayesiana é um grafo que segundo (Russell & Norvig, 1995; Heckerman, 1997):

- Possui um conjunto de variáveis aleatórias que representam os nós da rede.
- Possui um conjunto de setas que ligam dois nós. Por exemplo, se uma seta parte de um nó X para Y, diz-se que X é predecessor de Y.
- Possui uma Tabela de Probabilidade Condicional (TPC) para cada nó, que quantifica a dependência em relação aos nós-pais.
- Não possui ciclos (acíclico).

As redes Bayesianas utilizam a teoria da probabilidade para representar as incertezas de um domínio e codificam a relevância entre as variáveis através da vizinhança dos nós do grafo. Dessa forma, consultando-se os nós vizinhos tem-se todas as informações que podem ser relevantes para a variável, os outros nós podem ser ignorados (Castineira, 1996).

As redes bayesianas são conhecidas também por redes de crença, redes probabilísticas, redes causais, mapas de conhecimento ou diagramas de influência.

A seguir são discutidos a independência condicional, representação, inferência e aprendizado de redes Bayesianas.

3.3.1 Independência Condicional

Para definir a independência condicional entre as variáveis em uma rede Bayesiana, considere inicialmente três variáveis discretas (X , Y e Z) e um conjunto finito de possíveis valores dessas variáveis V . Considere também que a variável X é independente condicionalmente de Y dado Z , se a distribuição probabilística que controla a variável X é independente do valor de Y dado um valor Z . Essa interpretação pode ser representada na Equação 3.10.

$$(\forall x_i, y_j, z_k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k) \quad (3.10)$$

no qual, $x_i \in V(X)$, $y_j \in V(Y)$ e $z_k \in V(Z)$. A Equação 3.10 pode também ser escrita de forma abreviada, como mostra a Equação 3.11

$$P(X|Y, Z) = P(X|Z) \quad (3.11)$$

Essa definição de independência condicional pode também ser estendida para um conjunto de variáveis. Por exemplo, um conjunto de variáveis $X_1, \dots, X_l$ é independente condicionalmente do conjunto de variáveis $Y_1, \dots, Y_m$ dado o conjunto $Z_1, \dots, Z_n$, e essa independência pode ser descrita pela Equação 3.12.

$$P(X_1, \dots, X_l | Y_1, \dots, Y_m, Z_1, \dots, Z_n) = P(X_1, \dots, X_l | Z_1, \dots, Z_n) \quad (3.12)$$

3.3.2 Representação

Uma rede Bayesiana representa a Distribuição Probabilística Conjunta (DPC) de um conjunto de variáveis. Para ilustrar considere o exemplo de uma rede Bayesiana com as variáveis Arrombamento (A), Terremoto (T), Alarme (AL), AvisaJohn (AJ), AvisaMary (AM) e as tabelas de probabilidade condicional dessas variáveis, como pode ser visto na Figura 3.1.

Em geral, uma rede Bayesiana representa uma DPC especificando um conjunto de suposições de independência condicional, juntamente com os conjuntos de probabilidades condicionais. Cada variável possui dois tipos de informações: primeiro, os arcos da rede

representam a independência condicional das variáveis com relação aos seus nós não predecessores; e segundo, uma tabela de probabilidade condicional é fornecida para cada variável, descrevendo os valores da distribuição probabilística com relação aos nós predecessores (Mitchell, 1997).

Figura 3.1: Rede Bayesiana e a Tabela de Probabilidade Condicional (Russell & Norvig, 1995).

Dessa forma, utilizando as redes Bayesianas é possível fornecer uma descrição completa do domínio. Cada célula da Distribuição Probabilística Conjunta pode ser calculada a partir das informações da rede. Uma célula genérica da DPC é a probabilidade da conjunção das variáveis da rede, que pode ser dado por: $P(Y_1 = y_1 \wedge \dots \wedge Y_n = y_n)$. Essa notação pode também ser descrita de forma abreviada como $P(y_1, \dots, y_n)$. O valor dessa célula é calculado a partir da Equação 3.13, que representa o produtório dos elementos da tabela de distribuição condicional na rede.

$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i | \text{Nós} - \text{pais} Y_i) \quad (3.13)$$

Para ilustrar o cálculo da Equação 3.13, imagine a seguinte situação: um novo sistema de alarme foi instalado em uma casa para detectar arrobamentos e terremotos (suponha que

esta casa esteja localizada na cidade de Los Angeles). Os dois vizinhos dessa casa são John e Mary, que na rede são representados pelos nós *AvisaJohn* e *AvisaMary*, respectivamente. Esses vizinhos tem permissão de avisar ao dono da casa, em seu trabalho, caso o alarme toque. É importante ressaltar que John sempre avisa quando o alarme toca e, às vezes, confunde o toque do telefone com o alarme e nesse caso também avisa. Já Mary gosta de ouvir música muito alto e, às vezes, não escuta o alarme tocar.

Dadas as probabilidades condicionais e incondicionais, pode-se estimar a probabilidade do alarme tocar, mas não ter ocorrido arrobamento e terremoto, e mesmo assim ambos os vizinhos (John e Mary) avisarem ao dono que o alarme tocou. Essa situação está representada na Equação 3.14.

$$\begin{aligned}
 P(AL \wedge \neg A \wedge \neg T \wedge AJ \wedge AM) &= P(AL|\neg A\neg T)P(\neg A)P(\neg T)P(AJ|AL)P(AM|AL) \\
 P(AL \wedge \neg A \wedge \neg T \wedge AJ \wedge AM) &= 0,001 \times 0,999 \times 0,998 \times 0,90 \times 0,70 \\
 P(AL \wedge \neg A \wedge \neg T \wedge AJ \wedge AM) &= 0,00062
 \end{aligned} \tag{3.14}$$

Lembrando-se que ainda existem casos em que o alarme pode “falhar” por causa da alta umidade, bateria descarregada e entre outros imprevistos. Entretanto, esses imprevistos não são considerados nesse exemplo porque não é a intenção deste tópico.

Uma vez representada as redes Bayesianas, o usuário pode ter a necessidade de realizar diversas inferências para a obtenção de algum conhecimento desejado.

3.3.3 Inferência

A tarefa básica para um sistema de inferência probabilística é computar a distribuição da probabilidade condicional para um conjunto de variáveis de consulta dado os valores exatos das variáveis de evidência. A inferência probabilística é representada pela seguinte expressão:

$$P(Var_cons|Var_evid)$$

O processo de inferência pode ser simples, caso os valores das variáveis de evidência sejam conhecidos. De maneira geral, uma rede Bayesiana pode ser usada para computar a distribuição probabilística para sub-conjuntos de variáveis da rede, a partir dos valores ou distribuições de outro subconjunto de variáveis.

Por exemplo, na Figura 3.1 pode-se usar a rede Bayesiana para inferir o valor da variável alarme (variável de consulta) dado os valores das variáveis arrombamento e terremoto (variáveis de evidência). Essa inferência, pode ser representada simplesmente por:

$$P(AL|A, T)$$

As inferências podem ser realizadas basicamente em dois tipos de redes Bayesianas: o primeiro tipo são as redes simplesmente conexas (também denominada *poly-tree*) que é uma rede na qual não existe mais do que um caminho entre dois nós quaisquer. E o segundo tipo são as redes multiconexas, que permitem a existência de mais de um caminho entre dois nós quaisquer (Castineira, 1996). Um exemplo desses dois tipos de redes Bayesianas podem ser observados na Figura 3.2.

Figura 3.2: Redes a) Simplesmente Conexas e b) Multiconexas (Castineira, 1996).

Após ter verificado que as redes Bayesianas são ferramentas úteis para a manipulação de incertezas, muitas vezes o usuário tem a necessidade de construí-las. A seguir serão apresentadas sucintamente as duas formas de aprendizado das redes Bayesianas e alguns algoritmos que desempenham essa tarefa.

3.3.4 Aprendizado de Redes Bayesianas

O aprendizado de redes Bayesianas pode ser considerado de duas formas. A primeira se a rede está construída ou necessita ser inferida a partir dos dados de treinamento. E a segunda forma é que todas ou algumas variáveis podem ser diretamente observadas em cada exemplo de treinamento (Mitchell, 1997).

No caso, em que a estrutura da rede é fornecida (utilizando o conhecimento *a priori* do domínio) e todas as variáveis são observáveis nos exemplos de treinamento, o aprendizado da tabela probabilística condicional é simples. Se a estrutura da rede é fornecida e apenas algumas variáveis são observáveis nos dados de treinamento, esse aprendizado é mais difícil. Esse aprendizado é bastante similar ao aprendizado dos pesos dos nós das camadas intermediárias em redes neurais artificiais.

No entanto, existem duas dificuldades práticas nas aplicações desses métodos bayesianos. A primeira é que os métodos requerem tipicamente um conhecimento *a priori* de várias probabilidades. Quando essas probabilidades não são conhecidas, elas são estimadas, por exemplo, com base em conhecimento de fundo. A segunda dificuldade prática é o custo computacional requerido para determinar a “melhor hipótese” no caso geral (proporcional ao número de hipóteses candidatas) (Mitchell, 1997; Cohen, 1995).

Vários pesquisadores estão desenvolvendo algoritmos para auxiliar a automatizar a aquisição do conhecimento e a construção de redes Bayesianas a partir de dados de treinamento. Alguns algoritmos que permitem gerar redes Bayesianas são: *BKD (Bayesian Knowledge Discoverer)* (Ramoni & Sebastiani, 1997) e Hugin (Andersen et al., 1989). De maneira geral, esses algoritmos oferecem ambientes para realizar a aquisição do conhecimento de um domínio, representar e realizar inferências num modelo bem como ajudar o apoio às decisões. Informações mais detalhadas sobre as principais características, plataforma e referências bibliográficas/e-mail de alguns algoritmos que trabalham com redes Bayesianas podem ser encontradas em (Castineira, 1996).

3.4 Considerações Finais

Neste capítulo foram descritas duas abordagens utilizadas para representar e manipular aplicações que existem incertezas: teoria da probabilidade e redes Bayesianas.

A teoria da probabilidade é uma ferramenta importante para calcular o impacto que uma certa informação exerce sobre as condições do problema. Atualmente, essa abordagem é amplamente aceita e utilizada, embora já existam diversas outras técnicas para lidar com o problema da incerteza.

Em geral, existem duas formas para entender uma rede Bayesiana: a primeira consiste em observar a rede como uma representação da distribuição probabilística, que é útil no

entendimento da construção da rede; e a segunda consiste em codificar uma coleção de estruturas de independência condicional, que é útil para o desenvolvimento de procedimentos de inferência (Russell & Norvig, 1995). Redes Bayesianas são apropriadas tanto no processo de modelagem quanto no cálculo associado ao processo de inferência. A possibilidade de representar explicitamente as dependências do problema permite criar modelos mais realistas, enquanto a suposição de independência, sempre que possível, reduz as complexidades de tempo de computação e de armazenamento (Castineira, 1996).

O propósito deste capítulo foi dar uma visão geral do paradigma estatístico, pois uma familiaridade básica desse paradigma é importante para o entendimento das operações dos algoritmos de Aprendizado de Máquina que serão apresentados neste trabalho. A seguir será mostrada uma visão geral do aprendizado não supervisionado e uma descrição detalhada da tarefa de *clustering*.

Capítulo 4

Aprendizado Não Supervisionado e *Clustering*

4.1 Considerações Iniciais

Como apresentado na seção 2.4.1 página 12, existem duas abordagens a partir das quais os algoritmos de Aprendizado de Máquina podem aprender: o aprendizado supervisionado e o aprendizado não supervisionado. Neste capítulo são apresentadas algumas vantagens de trabalhar com o aprendizado não supervisionado e uma descrição detalhada de uma das suas principais tarefas (*clustering*).

Duas maneiras podem ser utilizadas para realizar a tarefa de *clustering*. A primeira utilizando os conceitos estatísticos vistos no capítulo 3 e a segunda enfatizando as medidas de similaridade, os tipos de análise de *cluster* e um resumo de alguns dos principais algoritmos para análise de *clusters*.

4.2 Aprendizado Não Supervisionado

Uma das razões de se utilizar os sistemas de aprendizado não supervisionado é quando a meta é aprender novos conceitos, que expressem a teoria do domínio a partir de conjuntos de dados, os quais não possuem o atributo-classe especificado. Nesse tipo de aprendizado, os padrões de dados são encontrados a partir de alguma caracterização lógica de regularidade (Decker & Focardi, 1995).

Um fato que deve ser ressaltado nessa abordagem é a existência de investigações desde décadas passadas (Duda & Hart, 1973). Basicamente, existem três grandes razões de interesse no aprendizado não supervisionado. Primeira, a coleção e o rotulamento de um grande conjunto de exemplos podem ser surpreendentemente caro em termos de custo e tempo. Segunda, em muitas aplicações as características dos exemplos podem mudar vagarosamente com o passar do tempo. E a terceira razão é que em alguns estágios da investigação, a descoberta de características deve ser valiosa para adquirir alguma percepção da natureza ou estrutura dos dados. Porém ainda hoje, o interesse na abordagem de aprendizado não supervisionado permanece.

Dentre as diversas tarefas de aprendizado não supervisionado (visto na Figura 2.3 página 8), este capítulo abordará detalhadamente apenas a tarefa de *clustering*.

4.3 *Clustering*

A tarefa de *clustering* é útil para explorar um conjunto de dados, que não seja bem conhecido. Essa tarefa é descritiva, como foi mostrado na Figura 2.3, ou seja, não é necessário designar uma coluna específica (atributo-classe). Além disso, o conjunto de dados não é dividido em conjuntos de treinamento e de teste. De maneira geral, o entendimento da tarefa de *clustering* pode ser visualizado na Figura 4.1.

Figura 4.1: Tarefa de *Clustering*.

Durante a revisão bibliográfica observou-se a existência de duas técnicas para realizar a tarefa de *clustering*.

A primeira é quando o algoritmo “aprende” através de cálculos de probabilidades envolvendo independência condicional e regra de Bayes. Um algoritmo que utiliza essa técnica é o *AutoClass* que será descrito na seção 5.4.2 (Cheeseman & Stutz, 1996). Como os

conceitos utilizados nessa técnica já foram vistos no capítulo 3, será dado maior enfoque na segunda técnica de *clustering*.

A segunda é quando o algoritmo “aprende” utilizando alguma medida de similaridade (Mizoguchi & Shimura, 1980; Nilsson, 1996). Alguns algoritmos que utilizam essa técnica são: *K-Means* (Hartigan & Wong, 1978) e Vizinho Mais Próximo (Ling & Wang, 1997).

Essa tarefa cria uma partição dos dados de modo que todos os membros de cada conjunto sejam similares. Por exemplo, suponha um conjunto de bolas, algumas delas são brancas e algumas vermelhas, uma partição natural é agrupar todas as bolas brancas e todas as vermelhas em dois conjuntos distintos. Se, no entanto, as bolas possuem dois tamanhos diferentes, grande e pequena, é natural criar uma partição com quatro conjuntos distintos: grande-branca, grande-vermelha, pequena-branca e pequena-vermelha.

Pode-se observar que em algumas situações não há um critério “claro” de partição dos dados. Como por exemplo, criar partições em um grupo de estudantes de uma determinada escola. Uma forma seria definir intervalos de variáveis contínuas, tal como idade, e posteriormente criar as partições possíveis (Decker & Focardi, 1995). Exemplos típicos de partição de alguns dados detectados pelos seres humanos podem ser observados na Figura 4.2 (Mizoguchi & Shimura, 1980).

Figura 4.2: Exemplos de Partição dos Dados (Mizoguchi & Shimura, 1980).

Sempre que se tem um número pequeno de fatores, o número de partições normalmente é baixo, e não muito útil. Dessa forma, surge a pergunta “Qual é o objetivo da

partição?”. Um objetivo poderia seria reunir os estudantes para encontrar a melhor classe possível. Um critério para essa partição seria a homogeneidade de classes, ou seja, uma classe com todos os seus membros similares. É claro, que, às vezes, o usuário tem uma noção antes da classificação e nesses casos o agrupamento se torna mais fácil.

Para realizar as partições dos dados é necessário determinar uma medida de similaridade, que geralmente envolve várias técnicas de formação de *clusters*. Uma classe dessas técnicas consiste na formação de regras que dita o relacionamento de um determinado grupo baseado num nível de similaridade entre seus membros. Essas regras não são triviais visto que a similaridade não é uma relação transitiva: se A é similar a B e B é similar a C, não é sempre verdade que A é similar a C (Decker & Focardi, 1995).

A tarefa de *clustering* pode também ser utilizada no modo supervisionado, iniciando a partir de um conjunto dado de *clusters*. Novos membros são então adicionados a cada classe em função de sua similaridade com os membros das classes já existentes (Decker & Focardi, 1995).

A seguir são apresentadas algumas medidas de similaridades para a realização das partições dos dados e em seguida os tipos de análise de *cluster* e alguns algoritmos de *clustering*.

4.3.1 Medidas de Similaridades

Como o objetivo da tarefa de *clustering* é encontrar uma partição natural num conjunto de dados, o grande problema é medir a similaridade entre os exemplos. A medida de similaridade mais simples entre dois exemplos é a distância entre eles.

Uma maneira para começar uma modelagem de *clustering* é definir uma distância apropriada e computar uma matriz de distâncias entre todos os pares de exemplos. As distâncias são boas medidas de similaridades, por isso espera-se que a distância entre dois exemplos num mesmo *cluster* seja mais significativas do que em *clusters* diferentes. Dessa forma, é essencial que as medidas de similaridades estejam bem definidas, considerando que elas descrevem o alto grau de associação entre os objetos de um determinado domínio. Três medidas de similaridade típicas usadas para a formação de *clusters* são: distância Euclidiana, *Manhattan* e Qui-quadrado.

A distância Euclidiana descreve o menor caminho entre dois pontos i -dimensionais,

como pode ser representada pela Equação 4.1 (Ling & Wang, 1997). Uma ilustração dessa distância utilizando dois pontos quaisquer pode ser vista na Figura 4.3.

$$d(x, y) = \sqrt{\sum (x_i - y_i)^2} \quad (4.1)$$

Figura 4.3: Uma Ilustração da Distância Euclidiana.

A distância de *Manhattan* descreve um caminho ortogonal entre dois pontos no espaço (Larson & Odoni, 1943). Essa distância pode ser representada pela Equação 4.2.

$$d = |x_1 - x_2| + |y_1 - y_2| \quad (4.2)$$

Uma forma simples de ilustrar a distância de *Manhattan* utilizando dois pontos quaisquer está exibida na Figura 4.4.

Outra distância usada na medida de similaridade é a Qui-quadrado (*Chi-squared*) (Hastie & Tibshirani, 1995) que pode ser observada na Equação 4.3. Nessa distância x_0 é o ponto de teste, X é o vizinho mais próximo e $p(j|x_0)$ é a probabilidade do ponto de teste x_0 pertencer a classe j .

$$r(X, x_0) = \sum_{j=1}^J \frac{[p(j|X) - p(j|x_0)]^2}{p(j|x_0)} \quad (4.3)$$

Dentre as medidas de similaridades existentes, a distância Euclidiana, descrita na Equação 4.1, é uma das mais utilizadas para calcular a similaridade entre *clusters* (Wong & Lane, 1983; McAuliffe & Affi, 1984). A escolha de uma dessas medidas envolve alguns

Figura 4.4: Uma Ilustração da Distância de *Manhattan*.

fatores importantes, tais como: o tipo das variáveis (discreta, contínua ou binária) e a escala de medições.

Para calcular as distâncias, é usual padronizar as variáveis para que estas tenham importância equivalente na determinação das medidas de similaridade. A codificação das variáveis é para que estas tenham o valor da média 0 e variância 1. Alternativamente, cada variável pode ser codificada para ter valor mínimo 0 e máximo 1 (Oliveira, 1997).

No caso dos objetos não poderem ser representados por medidas p -dimensionais significativas, as distâncias entre pares de objetos podem ser calculadas com base na presença ou ausência de certas características. Obviamente, os objetos similares terão mais características em comum do que objetos não similares. A presença ou ausência de uma característica pode ser descrita matematicamente através da introdução de uma variável binária que assume, por exemplo, o valor 1 se a característica estiver presente e o valor 0 se a característica estiver ausente. Seja x_{ij} o valor (0 ou 1) da j -ésima variável binária para o i -ésimo objeto e x_{kj} o valor (0 ou 1) da j -ésima variável para o k -ésimo objeto, e $j = 1, 2, \dots, p$. Consequentemente,

$$(x_{ij} - x_{kj})^2 = \begin{cases} 1, & x_{ij} \neq x_{kj} \\ 0, & \text{caso contrário} \end{cases}$$

A distância Euclidiana ao quadrado, $\sum_{j=1}^p (X_{ij} - X_{kj})^2$ fornece o número de características presentes em um dos objetos mas ausentes no outro (Oliveira, 1997).

Suponha que dois exemplos só pertençam ao mesmo *cluster* se a distância Euclidiana entre eles for menor do que um *threshold* d_0 . Assim, a escolha de d_0 é muito importante,

pois se d_0 for muito grande, todos os exemplos poderão pertencer a um único *cluster*. E se d_0 for muito pequeno, cada exemplo poderá formar um *cluster* isolado. Para obter *clusters* “naturais”, d_0 terá que ser maior do que as distâncias entre os exemplos num *cluster* e menor do que as distâncias entre os *clusters* existentes. Um exemplo com um conjunto de dados aleatório variando apenas o *threshold* d_0 , pode ser visualizado na Figura 4.5.

Figura 4.5: Distâncias do *Threshold* d_0 a) grande b) intermediário e c) pequeno (Duda & Hart, 1973).

Todas essas medidas de similaridade têm um mínimo de variância, e usualmente têm os mesmos resultados se os *clusters* são compactados e bem separados. De qualquer maneira, se os *clusters* estão próximos uns dos outros, ou se suas estruturas não são basicamente hiperesféricas, resultados diferentes podem ser obtidos. Pode-se observar que na Figura 4.6 são mostrados três exemplos de conjuntos de pontos bidimensionais que possuem umas dessas diferenças. A mudança de posição de alguns pontos altera diretamente na delimitação dos *clusters* (Duda & Hart, 1973). Na seção 4.3.3 são apresentadas essas mudanças utilizando o algoritmo de *clustering* Vizinho Mais Próximo.

Como já foi discutido uma idéia geral de *clustering* e as medidas de similaridade utilizadas para a formação de *clusters*, será apresentado a seguir os tipos de análises de *cluster*.

Figura 4.6: Três Exemplos de Conjuntos de Pontos no Espaço (Duda & Hart, 1973).

4.3.2 Análise de *Cluster*

A análise de *cluster* tem como objetivo descobrir agrupamentos naturais de objetos ou exemplos, tal que os objetos ou exemplos “similares” pertençam a mesma classe, sendo, para isso, necessário definir, primeiramente, uma escala quantitativa para medir a similaridade (associação) entre os objetos (Oliveira, 1997).

Análise de *cluster* é uma técnica na qual não é feita nenhuma suposição envolvendo o número ou a estrutura das classes. Os agrupamentos dos objetos são feitos com base em similaridades ou distâncias. As entradas requeridas para essa análise são, portanto, alguma medida de similaridade entre objetos ou dados, e a partir desses objetos são computadas tais medidas, como mostrado na seção 4.3.1.

Devido à dificuldade em explicar explicitamente o significado de “*cluster*”, foram propostos vários métodos que realizam tal análise. Entre eles, dois tipos de algoritmos que podem ser destacados são (Ester et al., 1996): algoritmos não-hierárquicos (ou particionamento) e hierárquicos.

Algoritmos de *Clustering* Não-Hierárquicos

Os algoritmos não-hierárquicos envolvem o conceito de particionamento, que constrói uma partição no conjunto de dados D de n objetos em um conjunto de k *clusters* (classes). O k é um parâmetro de entrada para esses algoritmos, isto é, algum conhecimento é necessário antes da sua execução.

Em seguida, todos os objetos são alocados em algumas classes. Um objeto é movido para uma classe nova se a distância entre o objeto e a classe nova for menor do que a distância entre o objeto e a classe atual. Classes próximas são unidas enquanto que as classes dispersas são divididas. Esse processo continua iterativamente até que um número pré-determinado de classes k seja atingido.

Os passos principais do algoritmo de *clustering* não-hierárquico são apresentados no seguinte procedimento:

1. Inicialmente todos os membros são alocados em algumas classes e são calculados os centróides (centro dos *clusters*).
2. Os membros são realocados em *clusters* mais próximos através do cálculo das distâncias entre os membros e os *clusters*.
3. Os centróides são recalculados e o passo 2 é repetido até que ocorra a convergência.
4. Se o número de *clusters* está dentro do número especificado, e as distâncias entre os *clusters* satisfaz um *threshold* prescrito, o *clustering* é considerado completo.

Os algoritmos de *clustering* não-hierárquico mais conhecidos são *K-Means* discutido brevemente na seção 4.3.3 (Hartigan & Wong, 1978) e *CLARANS* (*Clustering Large Applications based on RANdomized Search*) (Ester et al., 1996).

Algoritmos de *Clustering* Hierárquicos

Os algoritmos hierárquicos criam uma decomposição hierárquica no conjunto de dados D . Essa decomposição pode ser representada por um dendrograma que são vistos como uma árvore que mostra como os exemplos são agrupados. Existem dois processos que realizam essa decomposição: o processo de aglomeração (criado a partir das folhas até a raiz), e o

processo de divisão (criado a partir da raiz até as folhas), emergindo ou dividindo os *clusters* em cada passo.

Se o processo escolhido for o de aglomeração (*bottom-up, clumping*), todos os objetos começam isolados em classes de apenas um objeto. As classes “semelhantes” são gradualmente unidas até que todos os objetos estejam numa única classe.

Considere uma sequência de partições de n exemplos em c *clusters*. Na primeira partição existirá n *clusters*, cada *cluster* contendo exatamente um exemplo. A próxima partição terá $n - 1$ *clusters*, a próxima $n - 2$ *clusters* e isso continua até que os (n -ésimo) exemplos formem um único *cluster*. A obtenção da quantidade de *clusters* formados em cada nível k pode ser dada seguinte forma: $c = n - k + 1$. Dessa forma, o nível 1 corresponde a todos os *clusters* com 1 exemplo e o nível n com apenas 1 *cluster* de n exemplos.

Um dendrograma que representa um problema hipotético envolvendo seis exemplos é mostrado na Figura 4.7. O nível 1 mostra os 6 exemplos, cada um como um *cluster*. O nível 2 mostra que os exemplos X_3 e X_5 foram agrupados para formar um *cluster*, e assim por diante. Se é possível medir a similaridade entre *clusters*, então o dendrograma é usualmente indicado para mostrar as similaridades dos *clusters* que serão agrupados. Por exemplo, os *clusters* que são unidos no nível 6 possui um valor de similaridade igual a 30 (Duda & Hart, 1973). Os valores de similaridade são frequentemente utilizados para auxiliar a determinar quando os agrupamentos são naturais ou forçados. Para esse problema hipotético, os agrupamentos dos níveis 2, 3, 4 e 5 são naturais, mas o agrupamento do nível 6 é considerado forçado pelo fato de existir uma baixa similaridade entre os *clusters*.

Figura 4.7: Um Dendrograma para *Clustering* Hierárquico (Duda & Hart, 1973).

Os passos principais do algoritmo de *clustering* hierárquico aglomerativo são:

1. Fixar $X_i = \{x_i\}, i = 1, 2, \dots, n$ e $\hat{c} = n$ ($\hat{c}$ - é a quantidade de *clusters* no nível 1).
2. Se $\hat{c} \leq c$, pare.
3. Procurar pares de exemplos próximos em *clusters* distintos, por exemplo, X_i e X_j .
4. Agrupar X_i e X_j , remover X_j e decrementar $\hat{c}$ em 1 (um).
5. Voltar ao item 2.

Esse procedimento finaliza quando o número de *clusters* identificados tiver sido obtido. De qualquer maneira, se continuar até que $c = 1$ pode-se produzir um dendograma como o mostrado na Figura 4.7. Em algum nível a distância entre os *clusters* mais próximos pode fornecer um valor de similaridade para aquele nível.

Caso o processo escolhido do algoritmo hierárquico for o de divisão (*top-down, splitting*), todos os objetos começam em uma única classe e forma uma sequência de divisões de classes sucessivamente até que cada classe contenha somente um objeto. As aplicações do processo de divisão ocorrem quando o usuário não tem conhecimento sobre o número de classes a ser descoberto ou sobre a medida de distância no qual se baseia a classificação.

Os passos principais do algoritmo de *clustering* hierárquico divisivo são bastantes semelhantes ao algoritmo hierárquico aglomerativo, como pode ser visto a seguir:

1. Fixar $X_i = \{x_i\}, i = 1, 2, \dots, n$ e $\hat{c} = 1$.
2. Se $\hat{c} \geq c$, pare.
3. Procurar pares de exemplos distantes num mesmo *cluster*, por exemplo, X_i e X_j .
4. Separar X_i e X_j , adicionar X_j e incrementar $\hat{c}$ em 1 (um).
5. Voltar ao item 2.

Os algoritmos de *clustering* hierárquicos mais conhecidos são Vizinho Mais Próximo (Ling & Wang, 1997) e *Furthest Neighbour* (Duda & Hart, 1973). Na seção 4.3.3, o algoritmo Vizinho Mais Próximo será discutido com maiores detalhes.

De forma geral, a principal diferença entre os dois tipos de algoritmos de *clustering* apresentados é que nos algoritmos hierárquicos, ao contrário dos algoritmos não-hierárquicos, não é necessário especificar o número de *clusters* k como entrada.

Após a apresentação de uma visão geral das medidas de similaridade e os tipos de análise de *cluster* existentes, a seguir serão abordados dois algoritmos para análise de *cluster*. Um algoritmo de *clustering* hierárquico (Vizinho Mais Próximo) e um algoritmo de *clustering* não-hierárquico (*K-Means*).

4.3.3 Algoritmos para Análise de *Cluster*

Como mencionado anteriormente, existem muitos algoritmos para análise de *cluster*. Entretanto, não se pode afirmar que um método utilizado por um algoritmo é melhor do que o outro, já que algoritmos diferentes não produzem necessariamente os mesmos resultados para um determinado conjunto de dados.

Para tanto, será apresentado uma breve descrição de dois algoritmos para análise de *clusters* que são: Vizinho Mais Próximo e *K-Means*.

Vizinho Mais Próximo

Vizinho Mais Próximo (*Nearest Neighbour*) é um dos algoritmos de aprendizado mais estudados e populares (Ling & Wang, 1997). Originou-se do campo de reconhecimento de padrões e tem sido bastante aplicado nas tarefas de classificação e de *clustering* (Cover & Hart, 1967; McAuliffe & Affi, 1984).

Esse algoritmo determina a classificação de exemplos de teste de acordo com a similaridade (proximidade) dos exemplos de treinamento. Essa classificação pode ser dividida em dois passos:

1. Armazenar todos os exemplos de treinamento num espaço.
2. Utilizar uma função de similaridade (distância) para calcular a similaridade de um exemplo de teste com o conjunto k de exemplos de treinamento mais próximo.

A função de similaridade utilizada no segundo passo da classificação é a distância Euclidiana, descrita na seção 4.3.1. Nessa distância, x é o exemplo de teste, y é o exemplo

de treinamento armazenado, x_i e y_i são os valores normalizados dos i -ésimo atributos. De qualquer maneira, a função de similaridade descrita na Equação 4.1 trata todos os atributos igualmente, no qual para muitas aplicações não é interessante.

Algumas vezes, um determinado atributo pode ser completamente irrelevante para a classificação, assim esse atributo não deve ser incluído na função de similaridade. Uma função de similaridade mais eficiente pode ser mostrada na Equação 4.4:

$$d(x, y) = \sqrt{\sum w_i^2 \times (x_i - y_i)^2} \quad (4.4)$$

O termo w_i é uma constante determinada pelo usuário que identifica a importância do i -ésimo atributo na função de similaridade (Ling & Wang, 1997). Fixar os pesos na função de similaridade apropriadamente melhora a generalização do algoritmo Vizinho Mais Próximo, especialmente em domínios com atributos irrelevantes.

Intuitivamente, aos atributos mais importantes devem ser associados pesos maiores do que aos atributos menos importantes, ou seja, a um atributo totalmente irrelevante deve ser associado um peso 0. Os pesquisadores Ling e Wang têm investigado a especificação dos pesos dos atributos no algoritmo Vizinho Mais Próximo. Estudos mostram que os pesos dos atributos podem ser determinados por um algoritmo de aprendizado popular, tal como: ID3 ou C4.5 (Ling & Wang, 1997).

Uma ilustração da busca do algoritmo Vizinho Mais Próximo utilizando os três exemplos de um conjunto de pontos descrito na seção 4.3.2, Figura 4.6, pode ser observado na Figura 4.8.

Em todos os três exemplos, o algoritmo encontrou dois *clusters*. No primeiro caso (a), os *clusters* são compactados e bem separados. No segundo caso (b), a presença de alguns pontos fez com que o algoritmo encontrasse um *cluster* pequeno e outro *cluster* grande e alongado. No terceiro caso (c), ocorreram algumas mudanças de pontos e o algoritmo encontrou dois *clusters* alongados.

O algoritmo Vizinho Mais Próximo possui uma classificação extremamente flexível e não envolve nenhum pré-processamento (*fitting*) dos dados de treinamento (Hastie & Tibshirani, 1995).

Figura 4.8: Busca de *Clusters* com o Algoritmo Vizinho Mais Próximo (Duda & Hart, 1973).

K-Means

O algoritmo de *clustering K-Means* é aplicado tanto na área de Aprendizado de Máquina quanto em Redes Neurais Artificiais (Hartigan & Wong, 1978; Ismail & Kamel, 1989; Chinrungrueng & Séquin, 1995).

K-Means é um algoritmo que tem como finalidade agrupar vários exemplos num determinado número de *clusters* utilizando uma medida de similaridade, tais como: Euclidiana e *Manhattan*. Como o algoritmo *K-Means* está implementado no software *MinesetTM*, decidiu-se descrevê-lo detalhadamente na seção 5.4.4, após a descrição do software.

4.4 Considerações Finais

Neste capítulo foi apresentada uma visão geral do aprendizado não supervisionado, bem como algumas motivações para a utilização desse tipo de aprendizado.

Além disso, foi apresentada as duas formas de realizar partições dos dados. A primeira através dos cálculos de probabilidades, uma vez que os conceitos já tinham sido descritos

no capítulo 3. A segunda forma é através de medidas de similaridades entre os pontos num espaço p -dimensional.

Foi mostrado também as medidas de similaridades mais conhecidas, os dois tipos de algoritmos para análise de *cluster*: não-hierárquico e hierárquico e um resumo dos algoritmos Vizinho Mais Próximo e *K-Means*.

No próximo capítulo será apresentada uma breve descrição das ferramentas e dos algoritmos que foram utilizados neste trabalho.

Capítulo 5

Ferramentas e Algoritmos Utilizados

5.1 Considerações Iniciais

Neste capítulo é apresentada uma breve descrição das ferramentas e dos algoritmos utilizados neste trabalho. Inicialmente, é apresentado o software *MinesetTM* que foi utilizado para a formação e visualização de *clusters*. Posteriormente, é mostrada a biblioteca de algoritmos de Aprendizado de Máquina *MLC⁺⁺* (*Machine Learning Library in C⁺⁺*) usada para a execução do algoritmo *Naive Bayes*.

E finalmente, são apresentados os principais conceitos usados pelos algoritmos estatísticos: *Naive Bayes*, *AutoClass*, *ACPro* (*AutoClass Pro*) e *K-Means* mostrando alguns detalhes operacionais e discutindo algumas das suas limitações.

5.2 *MinesetTM*

O software *MinesetTM*, versão 2.6, é uma ferramenta da *Silicon Graphics*, para auxiliar a aquisição de conhecimento de dados. Ela consiste de um conjunto de algoritmo de Aprendizado de Máquina implementados na biblioteca *MLC⁺⁺* (que será detalhada na próxima seção) e de várias ferramentas de visualização de dados num espaço multidimensional.

O *MinesetTM* permite a geração e análise de regras de associação e de *clusters*, classificação e indução de árvore de decisão, regressão, importância da coluna e o *plugin API* (*Application Programming Interface*) *ACPro*. Além disso, possui ainda operações adicionais, tais como: discretização de atributos, seleção, criação ou eliminação de atributo, filtragem

de exemplos e mudança de tipo de dados.

Uma visão geral da estrutura do *MinesetTM* e dos elementos que fazem parte desse software pode ser visualizada na Figura 5.1 (Félix, 1998). Esse software utiliza como suporte principal a biblioteca *MLC⁺⁺* para trabalhar na etapa de mineração de dados. O utilitário *MindUtil* é o responsável pela interface entre a *MLC⁺⁺* e o *MinesetTM*. O *DataMover* é um processo que roda no servidor, sempre que o software é iniciado. O *ToolManager*, que é a interface gráfica utilizada pelos usuários do *MinesetTM*, permite interagir com os algoritmos e com as ferramentas de visualização.

Figura 5.1: Visão Geral dos Elementos Constituintes do *MinesetTM* (Félix, 1998).

A interface gráfica do *MinesetTM* chamada *Tool Manager* pode ser vista na Figura 5.2 e suporta três componentes básicos: as ferramentas de visualização, de mineração e o módulo de controle centralizado.

As ferramentas de visualização permitem ilustrar os dados de uma forma interativa e animada. As ferramentas de visualização disponíveis são: *Tree Visualizer*, *Scatter Visualizer*, *Map Visualizer*, *Rule Visualizer*, *Evidence Visualizer*, *Cluster Visualizer*, *Record Visualizer*, *Statistic Visualizer* e *Splat Visualizer*.

Dentre as ferramentas de visualização descritas, *Cluster Visualizer*, *Evidence Visualizer* e *Scatter Visualizer* são os visualizadores utilizados nesse trabalho para a análise dos *clusters* encontrados pelos algoritmos *ACPro* e *K-Means*.

A ferramenta *Cluster Visualizer* (*ClusterViz*) é uma tabela *n*-dimensional que mostra

Figura 5.2: Interface Gráfica do Software *Mineset*TM.

em cada linha um atributo do conjunto de dados e em cada coluna o *cluster* encontrado. A primeira coluna, também chamada de *Population*, descreve uma estatística de cada atributo do conjunto de dados. Quando o atributo é do tipo discreto, a célula apresenta todos os possíveis valores, bem como a quantidade total que cada valor possui (histograma). E quando o atributo é do tipo contínuo, a célula apresenta a média, mediana, mínimo e máximo dos possíveis valores. Nas demais colunas (*clusters* encontrados), se os atributos forem do tipo discretos, as células possuem todos os possíveis valores, bem como a quantidade que cada *cluster* possui. E se forem do tipo contínuos, as células indicam a média, mediana, mínimo e máximo dos valores encontrados nesse *cluster*.

A ordem em que os atributos são apresentados no *ClusterViz* é significativa, por

default, os atributos são mostrados na ordem de importância para a discriminação dos *clusters*. Uma ilustração dessa ferramenta utilizando um conjunto de dados de seguros pode ser observada na Figura 5.3.

Figura 5.3: Interface da Ferramenta *Cluster Visualizer*.

A ferramenta *Evidence Visualizer* (*EvidenceViz*) mostra a estrutura de um classificador de evidências e auxilia o entendimento da importância de valores de atributos específicos para a classificação. Uma ilustração dessa ferramenta utilizando o conjunto de dados de tipos de plantas (*iris*) pode ser observada na Figura 5.4.

Essa ferramenta pode ser analisada de duas formas distintas. A primeira forma mostra as probabilidades diretas com gráficos de tortas tridimensionais, no qual a altura da torta representa o número de exemplos para aquele valor ou intervalo. No lado esquerdo estão apresentados os atributos e à direita a distribuição das classes do conjunto de dados. A segunda forma está baseada na seleção de uma única classe, no qual os gráficos de torta são substituídos por barras tridimensionais. A altura de cada barra representa a probabilidade logarítmica (evidência) a favor da classe especificada, dado que um único atributo é fixado a um valor específico.

Ambas as formas suportam uma classificação interativa dos dados. Os usuários podem fazer perguntas do tipo: “como seria se eu tivesse um exemplo com certos valores” e observar as predições do classificador.

Figura 5.4: Interface da Ferramenta *Evidence Visualizer*.

A ferramenta *Scatter Visualizer* fornece uma visão geral do conjunto de dados num espaço n -dimensional. Essa ferramenta permite analisar os relacionamentos entre vários atributos estatisticamente ou por animação. Os atributos do conjunto de dados podem ser definidos como eixos do espaço, tamanho e cor. Uma visualização da ferramenta *Scatter Visualizer* utilizando um conjunto de dados de tipo de plantas iris pode ser vista na Figura 5.5.

Para maiores informações, encontra-se disponível um relatório técnico sobre as ferramentas de visualização do *Mineset*TM (Oliveira & Rezende, 1998), além do guia do usuário do próprio software.

Como já mencionado, as ferramentas de mineração suportam a geração de regras de associação e de *clusters*, classificação e indução de árvore de decisão, regressão, importância da coluna e *ACPro*. Estas ferramentas combinadas com alguns visualizadores possibilitam explorar os modelos, determinar a precisão dos algoritmos e fazer perguntas do tipo *What-if*. As ferramentas de mineração para análise de *clusters* utilizadas neste trabalho são a importância de coluna, o gerador de *cluster* baseado no algoritmo *K-Means* (que será discutido na seção 5.4.4) e o *plugin ACPro* (que será discutido na seção 5.4.3) .

A ferramenta de mineração importância de coluna (*Column Importance*) determina quais atributos são mais importantes para a discriminação de valores de um atributo es-

Figura 5.5: Interface da Ferramenta *Scatter Visualizer*.

pecífico (geralmente é o atributo-classe), e indica o grau de pureza dos atributos. O grau de “pureza” (número entre 0 e 100) indica o quanto os atributos diferenciam desse atributo específico. Essa ferramenta é bastante utilizada para determinar os atributos mais relevantes do conjunto de dados e com isso mapeá-los para os eixos do espaço n -dimensional no *Scatter Visualizer*.

O módulo de controle centralizado consiste da interface gráfica *Tool Manager* e um processo *DataMover* que é executado no servidor sempre que o software começa a operar.

5.3 MLC^{++}

MLC^{++} é uma biblioteca de classes em C^{++} de algoritmos de Aprendizado de Máquina. Essa biblioteca foi projetada para auxiliar na seleção de algoritmos e/ou no desenvolvimento de novos algoritmos de Aprendizado de Máquina.

A biblioteca MLC^{++} é utilizada tanto como um sistema para os usuários finais quanto para desenvolvedores de algoritmos. Nessa biblioteca existem mais de 30 algoritmos de Aprendizado de Máquina, além de fornecer utilitários e ferramentas que podem ser usadas isoladamente (Félix et al., 1998). A MLC^{++} encontra-se disponível em <http://www.sgi.com/Technology/mlc/index.html>.

Um fator interessante na *MLC++* é que como existem vários algoritmos, recomenda-se testar vários algoritmos com um mesmo conjunto de dados e posteriormente comparar e determinar qual algoritmo apresenta o melhor desempenho para esse conjunto.

Os usuários da biblioteca *MLC++* devem conhecer o funcionamento básico dos algoritmos de Aprendizado de Máquina, pois não existe um algoritmo considerado “ótimo” para todos os conjuntos de dados. Existem algoritmos que possuem um desempenho ótimo para alguns conjuntos de dados, porém para outros apresentam um desempenho fraco. Isso se deve a vários fatores, tais como: tamanho do arquivo, número de variáveis, quantidade de valores contínuos e discretos e etc. (Kohavi & Kunz, 1997).

Alguns dos algoritmos implementados na *MLC++* são: *Naive Bayes* (Domingos & Pazzani, 1996), C4.5 (Quinlan, 1993), CN2 (Clark & Niblett, 1989) e *Nearest Neighbour* (Wettschereck, 1994). Uma descrição completa sobre a biblioteca *MLC++* e como utilizar alguns algoritmos de Aprendizado de Máquina pode ser encontrado em (Félix et al., 1998).

Dentre os utilitários da biblioteca *MLC++*, apenas o utilitário *Dotty* é utilizado neste trabalho. Esse utilitário permite visualizar o espaço de busca utilizado pelo algoritmo com um determinado conjunto de dados. Esse espaço de busca é armazenado diretamente num arquivo específico com extensão “.dot”. Na Figura 5.6 é apresentado o espaço de busca gerado pelo *Naive Bayes* utilizando o conjunto de dados iris.

Na Figura 5.6, cada nó possui as seguintes informações:

1. A primeira linha indica o número do nó, que está localizado logo após o símbolo #. Esses números são mostrados a partir da ordem em que foram calculados. Os números que estão entre colchetes mostram um conjunto de atributos.
2. A segunda linha apresenta a estimativa de precisão (por exemplo, *cross-validation*, *bootstrap* e *holdout*) com a média do desvio padrão.
3. A terceira linha aparecerá somente se os nós no qual a precisão real foi calculada (a precisão no conjunto de teste). Por *default*, apenas os nós que foram considerados os “melhores” em alguma fase terão este número. Assim, pode-se verificar a combinação de relevância entre os atributos.

Figura 5.6: Interface do Utilitário *Dotty*.

5.4 Algoritmos Estatísticos

Nesta seção são apresentados alguns dos algoritmos estatísticos mais conhecidos. Esses algoritmos trabalham diretamente com cálculos de probabilidades, regra de Bayes (mostrada na seção 3.2) ou até mesmo com algumas medidas de similaridades (apresentadas na seção 4.3.1).

Uma visão geral dos algoritmos estatísticos: *Naive Bayes*, *AutoClass*, *AutoClass Pro* e *K-Means* é mostrada a seguir.

5.4.1 *Naive Bayes*

O algoritmo *Naive Bayes* é baseado na abordagem Bayesiana, na qual as classes são descritas a partir da distribuição probabilística sobre os atributos dos exemplos.

O algoritmo supervisionado *Naive Bayes* está entre os métodos de aprendizado Bayesiano. A entrada desse algoritmo consiste de um conjunto de dados, na forma de atributo/valor. O *Naive Bayes* projeta um classificador com base nas probabilidades incondicionais do atributo-classe a partir do conjunto de treinamento. Como esse algoritmo está implemen-

tado na biblioteca *MLC++*, a sua saída depende das opções que forem fixadas pelo usuário. Algumas dessas saídas são a matriz de confusão que mostra uma estatística dos erros que o algoritmo cometeu com o conjunto de teste; o espaço de busca do algoritmo usando o utilitário *Dotty* da própria biblioteca *MLC++* e a distribuição probabilística das classes que podem ser visualizadas utilizando a ferramenta *Evidence Visualizer* do *MinesetTM*.

Idéia Geral do Algoritmo

O *Naive Bayes* “aprende” a partir de um conjunto de exemplos, no qual cada exemplo x é descrito por uma conjunção de valores de atributos e um atributo-meta ou atributo-classe $f(x)$, que pode obter um valor possível do conjunto de classes V . Um conjunto de exemplos de treinamento com o atributo-classe é fornecido e um novo exemplo é apresentado, contendo as tuplas de atributo/valor $(a_1, a_2, \dots, a_n)$ (Mitchell, 1997).

O *Naive Bayes* classifica um novo exemplo associando a classe mais provável, v_{MAP} , a partir dos valores de seus atributos $(a_1, a_2, \dots, a_n)$, como pode ser visto na Equação 5.1.

$$v_{MAP} = \operatorname{argmax}_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n) \quad (5.1)$$

Essa equação pode também ser reescrita utilizando a regra de Bayes, como mostra a Equação 5.2.

$$\begin{aligned} v_{MAP} &= \operatorname{argmax}_{v_j \in V} \frac{P(a_1, a_2, \dots, a_n | v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)} \\ v_{MAP} &= \operatorname{argmax}_{v_j \in V} P(a_1, a_2, \dots, a_n | v_j) P(v_j) \end{aligned} \quad (5.2)$$

Para determinar a probabilidade de uma classe $P(v_j)$ num conjunto de dados, basta contar a frequência que cada v_j ocorre no conjunto de treinamento.

Entretanto, determinar os termos $P(a_1, a_2, \dots, a_n | v_j)$ desta maneira não é fácil, principalmente em conjuntos de dados muito grande. O problema é que o número de termos é igual ao número de exemplos possíveis vezes o número de classes possíveis. Dessa forma, é necessário observar várias vezes cada exemplo no espaço para obter uma estimativa confiável.

O classificador *Naive Bayes* é baseado nas suposições dos valores dos atributos e que esses são independente condicionalmente dado o atributo-classe. Em outras palavras, a suposição dada a uma classe de um exemplo é o produtório das probabilidades dos atributos

individuais: $P(a_1, a_2, \dots, a_n | v_j) = \prod_i P(a_i | v_j)$. Substituindo essa expressão na Equação 5.2, tem-se a abordagem usada pelo classificador *Naive Bayes*, que pode ser vista na Equação 5.3.

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j) \quad (5.3)$$

no qual v_{NB} denota o valor da classe do classificador *Naive Bayes*.

É importante notar que no classificador, o número de termos distintos $P(a_i | v_j)$ que deve ser estimado dos dados de treinamento é o número de atributos distintos vezes o número de classes distintas (Mitchell, 1997).

No método de aprendizado do classificador *Naive Bayes*, os termos $P(v_j)$ e $P(a_i | v_j)$ são estimados baseados nas suas frequências no conjunto de treinamento. O conjunto dessas estimativas corresponde às hipóteses aprendidas. Essas hipóteses são então utilizadas para classificar cada novo exemplo aplicando a Equação 5.3. Quando a suposição do classificador de independência condicional é satisfeita, a classificação v_{NB} é idêntica à classificação h_{MAP} , descrita na Equação 3.2 página 20.

O algoritmo *Naive Bayes* é bastante robusto para atributos irrelevantes, e a classificação explora muitos atributos para fazer a predição final, uma característica importante quando não existe um “atributo principal” (Kohavi, 1996). As probabilidades são estimadas diretamente dos dados baseados em contagens, tais como: *Laplace* ou *m-estimate*.

O *Naive Bayes* trabalha com o conceito de redes Bayesianas, no qual a variável da classe é v_j e os atributos são $a_1, a_2, \dots, a_n$, como pode ser visualizada na Figura 5.7. A indução do *Naive Bayes* é rápida quando se trabalha com conjuntos de dados no qual existem apenas atributos discretos. Porém, num conjunto de dados que possui atributos contínuos é viável realizar uma fase de discretização desses atributos para que tenham somente um número pequeno de valores possíveis (Dietterich, 1997).

Uma diferença interessante entre o método de aprendizado do classificador *Naive Bayes* e os outros métodos de aprendizado é que nesses métodos não existem busca explícita através do espaço de hipóteses possíveis (nesse caso, o espaço de hipóteses possíveis é o espaço de valores possíveis que pode ser associado aos vários termos de $P(v_j)$ e $P(a_i | v_j)$). Em alguns domínios o desempenho do classificador *Naive Bayes* tem sido comparável com o aprendizado de Redes Neurais Artificiais e árvores de decisão, tais como: C4.5 e ID3

Figura 5.7: Rede Bayesiana para o Classificador *Naive Bayes* (Dietterich, 1997).

(Pazzani, 1995; Mitchell, 1997).

Descrição do Algoritmo

O algoritmo *Naive Bayes* usa a base dos métodos de aprendizado Bayesiano que requerem conhecimento sobre as probabilidades incondicionais das hipóteses e sobre a probabilidade de observação de dados a partir das hipóteses.

Naive Bayes é um algoritmo de aprendizado Bayesiano que tem sido bastante útil em muitas aplicações práticas. Esse algoritmo é chamado de “*naive*” porque incorpora uma suposição simples de que cada valor de atributo são condicionalmente independente dado a classificação de um exemplo.

O algoritmo, em alto nível, envolvendo as etapas de Treinamento e Teste, que mostra como o *Naive Bayes* classifica seus exemplos é descrito a seguir:

- **Etapas de Treinamento**

1. Calcular as probabilidades das classes $P(v_j)$. Para cada v_j de V faça:
 - (a) Selecionar um subconjunto de exemplos que pertençam a uma classe v_j ($class_j$).
 - (b) Fixar a probabilidade de cada classe pela sua frequência:

$$P(v_j) = \frac{|class_j|}{|Todos_Exemplos|}$$

- **Etapas de Teste**

1. Selecionar um novo exemplo e calcular o v_{NB} desse exemplo da seguinte forma:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in \text{posições } P(a_i|v_j)}$$

2. Normalizar a probabilidade da classe pertencer ao novo exemplo baseado no valor de v_{NB} da seguinte forma:

(a) Selecionar o maior valor de v_{NB} ($v_{NB\text{maior}}$).

(b) Calcular a probabilidade normalizada:

$$Prob - Normalizada = \frac{v_{NB\text{maior}}}{\sum_i^n v_{NBi}}$$

Exemplo: Aprendizado para Classificar Textos

Para ilustrar a importância do algoritmo *Naive Bayes*, considere um aprendizado no qual os exemplos são documentos textos. Por exemplo, considere um usuário que deseje ler sobre as novidades eletrônicas ou páginas da rede que discutam tópicos sobre Inteligência Artificial. Em ambos os casos, se um computador pudesse aprender os objetivos do usuário, isso filtraria bastante o grande volume de dados e apresentaria somente os documentos mais relevantes (Mitchell, 1997).

Considere um espaço de exemplos X consistindo de todos os possíveis documentos textos (todas as possíveis palavras, pontuação e *tokens*). Os exemplos de treinamento dados possuem um atributo-classe $f(x)$ que pode obter um valor do conjunto finito V . A tarefa é aprender a partir desses exemplos de treinamento para prever um atributo-classe para os documentos textos posteriores. Nesse exemplo, considere a função alvo como *gosta* e *não gosta* para indicar as duas classes.

Num parágrafo de um documento cada palavra corresponde a um atributo. Dessa forma, supondo um parágrafo com 110 palavras tem-se 110 atributos. O valor do primeiro atributo é a palavra “Esta”, o valor do segundo atributo é a palavra “abordagem” e assim por diante, até o valor do último atributo que é a palavra “claro”.

A partir dessa representação para documentos textos, pode-se aplicar o classificador *Naive Bayes*, assumindo que 700 documentos de treinamento estão classificados como

não gosta, e 300 documentos classificados como *gosta*. Para determinar a classe de um novo documento texto, supondo novamente o parágrafo com as 110 palavras, será necessário instanciar a Equação 5.3 para calcular a classificação do *Naive Bayes*, na qual pode ser observada na Equação 5.4.

$$\begin{aligned} v_{NB} &= \operatorname{argmax}_{v \in \text{gosta}, \text{não gosta}} P(v_j) \prod_{i=1}^{110} P(a_i|v_j) \\ v_{NB} &= \operatorname{argmax}_{v \in \text{gosta}, \text{não gosta}} P(v_j) P(a_1 = \text{"Esta"}|v_j), \dots, P(a_{110} = \text{"claro"}|v_j) \quad (5.4) \end{aligned}$$

Em resumo, a classificação *Naive Bayes* v_{NB} é a classificação que aumenta a probabilidade de observação das palavras que são encontradas no documento, assunto para a suposição de independência do classificador. Essa suposição de independência $P(a_1, \dots, a_{110}|v_j) = \prod_{i=1}^{110} P(a_i|v_j)$ indica que a probabilidade da primeira palavra é independente das outras palavras que estão em outras posições, dado a classificação do documento v_j , porém note que essa suposição é incorreta. Por exemplo, a probabilidade de encontrar a palavra “Artificial” em alguma posição deve ser muito maior se a palavra “Inteligência” preceder.

Para calcular v_{NB} usando a expressão anterior, é necessário estimar as probabilidades de $P(v_j)$ e $P(a_i = w_k|v_j)$, no qual w_k indica k -ésimo palavras no vocabulário. As probabilidades incondicionais das classes podem ser estimadas diretamente do conjunto de treinamento, como, por exemplo, $P(\text{gosta}) = 0,3$ e $P(\text{não gosta}) = 0,7$. Estimar as probabilidades condicionais da classe (p.ex., $P(a_1 = \text{"Esta"}|\text{não gosta})$) é muito problemático, pois para isto é necessário ter uma probabilidade para cada combinação da posição no texto, palavra e classe. Infelizmente, existe aproximadamente 50.000 palavras distintas no vocabulário, 2 classes possíveis e 110 posições de palavras no texto exemplo, devendo-se estimar $50.000 \times 2 \times 110 \approx 10$ milhões de termos.

Felizmente, o número de termos que deve ser estimado pode ser reduzido. A rede determina 2×50.000 termos distintos da forma $P(w_k|v_j)$. Essa quantidade assume que os atributos são independentes e distribuídos identicamente, dado a classificação, que é $P(a_i = w_k|v_j) = P(a_m = w_k|v_j)$ para todos os i, j, k e m . Para completar a estrutura do algoritmo de aprendizado, o método escolhido para estimar as probabilidades é o m -estimate, descrito na Equação 5.5, na qual pode se estimar a $P(w_k|v_j)$.

$$\frac{n_k + 1}{n + |\text{Vocabulário}|} \quad (5.5)$$

Nesta equação n é o número total de posições de palavras no conjunto de treinamento no qual a classe é v_j , n_k é o número de vezes que a palavra w_k é encontrada entre n posições de palavras, e $|\text{Vocabulário}|$ é o número total de palavras distintas (e outros *tokens*) encontradas no conjunto de treinamento.

No final, o algoritmo usa o classificador *Naive Bayes* juntamente com as suposições das probabilidades de ocorrência das palavras no texto. O algoritmo é composto de 2 procedimentos que são descritos a seguir.

Durante a fase de aprendizado, o procedimento *Learn-Naive-Bayes-Text* examina todos os documentos de treinamento para extrair o vocabulário com todas as palavras e *tokens* que aparecem no texto, e então conta sua frequência entre as diferentes classes para obter a estimativa de probabilidade necessária.

Posteriormente, para um novo documento ser classificado, o procedimento *Classify-Naive-Bayes-Text* usa as estimativas de probabilidade para calcular v_{NB} de acordo com a Equação 5.3. Note que algumas palavras apresentadas no novo documento podem não ter sido observadas no conjunto de treinamento, e nesse caso são simplesmente ignoradas pelo *Classify-Naive-Bayes-Text*.

A seguir será apresentado o algoritmo em alto nível para classificar textos baseado no classificador *Naive Bayes*. Esse algoritmo está dividido em duas fases. A primeira realiza um processo de aprendizagem com um conjunto de textos. A segunda realiza o processo de classificação de um texto novo.

Learn-Naive-Bayes-Text(Exemplos. V)

Exemplos é um conjunto de documentos textos com suas respectivas classes. V é o conjunto de possíveis valores de classes. Esse procedimento aprende os termos probabilísticos $P(w_k|v_j)$, descrevendo a probabilidade de uma palavra w_k aleatoria em documento na classe v_j . Esse procedimento aprende também a probabilidade incondicional da classe $P(v_j)$. A seguir, são apresentados os dois passos do procedimento de aprendizado.

1. Coletar todas as palavras, pontuação e outros *tokens* que ocorre em *Exemplos*
 - Vocabulário $\leftarrow$ um conjunto de todas as palavras distintas e outros *tokens* ocorridos em algum documento texto de *Exemplos*.

2. Calcular a probabilidade dos termos requeridos $P(v_j)$ e $P(w_k|v_j)$:

- Para cada classe v_j em V faça:
 - $docs_j \leftarrow$ o subconjunto de documentos de *Exemplos* para cada qual a classe é v_j .
 - $P(v_j) \leftarrow \frac{|docs_j|}{|Exemplos|}$
 - $Texto_j \leftarrow$ um documento criado pela concatenação de todos os membros de $docs_j$.
 - $n \leftarrow$ número total de posições de palavras distintas no $Texto_j$.
 - para cada palavra w_k no *Vocabulário*:
 - * $n_k \leftarrow$ número de vezes que a palavra w_k ocorre no $Texto_j$.
 - * $P(w_k|v_j) \leftarrow \frac{n_k+1}{|n+Vocabulário|}$.

Classify-Naive-Bayes-Text(Doc)

Retorna a classe estimada para um documento *Doc*, e a_i denota a palavra encontrada na posição i -ésimo de *Doc*.

- Posições $\leftarrow$ todas as posições das palavras em *Doc* que contém *tokens* encontrados no *Vocabulário*.
- Retorna v_{NB} , no qual

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in \text{posições}} P(a_i|v_j).$$

5.4.2 *AutoClass*

*AutoClass*¹ é um algoritmo de *clustering* baseado na abordagem Bayesiana. Vale lembrar que nessa abordagem, as classes são descritas a partir da distribuição probabilística sobre os atributos dos exemplos, especificados por uma função modelo e por seus parâmetros (Cheeseman et al., 1990). Definir uma classe nesse algoritmo é descrever (não listar) os exemplos que pertencem a essa classe.

A entrada desse algoritmo consiste de um conjunto de dados, na forma de atributo/valor, definição de modelos e de parâmetros de busca. O *AutoClass* procura um conjunto

¹A versão descrita é o *AutoClass C 3.1*.

de classes que seja altamente provável com os dados e modelos especificados. A saída desse algoritmo é a descrição probabilística das classes identificadas e dos exemplos pertencentes a essas classes.

Idéia Geral do Algoritmo

No algoritmo *AutoClass*, a determinação do número de classes para uma classificação é opcional, ou seja, o usuário tem a opção de fornecer o número de classes que deseja numa determinada classificação. Caso o usuário não determine, o algoritmo tenta encontrar as classes automaticamente para a melhor classificação. Decidir quando deve suspender a formação de classes é uma questão fundamental na classificação desse algoritmo, mas o que limita o número de classes é o uso do conhecimento de fundo.

Os exemplos não são absolutamente associados às classes. O cálculo da probabilidade de cada exemplo pertencer a uma determinada classe fornece uma classificação mais intuitiva do que as técnicas de particionamento.

Todos os atributos do conjunto de dados são potencialmente relevantes, isso representa uma vantagem sobre a classificação humana, que só permite a manipulação de poucos atributos. O *AutoClass* utiliza todos os atributos simultaneamente, permitindo uma consideração uniforme de todos os dados. Os tipos de atributos aceitáveis nesse algoritmo são real ou discreto.

O usuário pode executar o *AutoClass* com um número de classes J maior do que o número esperado. Se todas as classes têm uma probabilidade significativa, então o usuário pode incrementar J até que algumas classes estejam vazias. O *AutoClass*, então ignora as classes vazias e as classes que possuem exemplos representam um classificador ótimo dos dados a partir das classes assumidas na função modelo.

O *AutoClass* trabalha com o conceito de redes Bayesianas, no qual existe uma variável discreta “escondida” H que possui possíveis valores que correspondam as classes nos dados, como pode ser apresentada na Figura 5.8. Por razões de eficiência computacional, o algoritmo assume que os atributos discretos (D_1, D_2 e D_3) e os conjuntos de atributos contínuos definidos pelo usuário ($\{C_1, C_2, C_3\}$ e $\{C_4, C_5\}$) são mutuamente independentes da variável H . Apresentado como entrada um conjunto de dados, o *AutoClass* busca variações no número de valores da variável “escondida” e seleciona uma variação que possua a melhor probabilidade relativa (Heckerman, 1997).

Figura 5.8: Rede Bayesiana para o Algoritmo *AutoClass* (Heckerman, 1997).

A execução do algoritmo *AutoClass* pode ser dividida em três fases principais: a fase de busca, de geração de relatório e de descrição.

- Fase de Busca: nessa fase o algoritmo realiza as buscas de classes no conjunto de exemplos de treinamento, observando o tipo de cada atributo, os modelos e os parâmetros de busca definidos. Uma característica interessante no algoritmo *AutoClass* é a possibilidade de realizar várias buscas de classes numa única execução. Além disso, existe possibilidades de recommear as buscas de classes realizadas anteriormente numa outra execução.
- Fase de Relatório: nessa fase o algoritmo realiza uma descrição completa das classes identificadas e o relacionamento parcial dos exemplos nessas classes. Os tipos de relatórios existentes são:
 1. Relatório de referência cruzada através dos números dos exemplos: descreve as classes prováveis para cada exemplo, ordenado por número de exemplo.
 2. Relatório de referência cruzada através do número de classes: cada classe é apresentada separadamente, e em cada uma dessas classes os exemplos são ordenados por número de exemplo.
 3. Relatório de influência relativa ou significativa dos atributos de dados: apresenta um resumo da busca gerada, uma lista ordenada das classes encontradas, bem como, uma lista ordenada das influências dos atributos sobre todas as classes.

- Fase de Descrição: essa fase realiza a descrição de novos exemplos (teste), prevendo o relacionamento probabilístico desses exemplos com as classes geradas na fase da busca (treinamento).

O algoritmo *AutoClass* procura um conjunto de classes que seja altamente provável em relação aos dados e modelos. A definição desses modelos indica como os atributos são tratados para a determinação do número de classes. Existem cinco tipos de modelos que são explicados a seguir:

1. *single_multinomial*: esse modelo é utilizado para atributos discretos ou inteiros, que são condicionalmente independentes dos outros atributos dado a classe. Permite a existência de valores ausentes. Neste caso, o algoritmo acrescenta esse valor (p.ex. “?”) para o conjunto de valores possíveis daquele atributo.
2. *single_normal_cn*: esse modelo é utilizado para atributos reais, que são condicionalmente independente dado a classe (distribuição normal²). Não permite a existência de valores ausentes. É diretamente aplicável aos atributos reais positivos. Quando se trabalha com atributos reais no *AutoClass* é criada uma cópia do mesmo com os valores da distribuição utilizada e também é incrementada a palavra *log* no início do nome para distinguir do atributo original.
3. *single_normal_cm*: esse modelo é utilizado para atributos reais, que são condicionalmente independente dado a classe (distribuição normal), mas permite a existência de valores ausentes. Neste caso, o algoritmo determina duas categorias para esse atributo. Para os “*missing*” utiliza uma probabilidade discreta binária P (normal) e para os “*known*” uma probabilidade igual a $1 - P$. É diretamente aplicável aos atributos reais positivos.
4. *multi_normal_cn*: esse modelo é utilizado para conjuntos de valores do tipo real, que são condicionalmente independente dos outros atributos dado a classe (distribuição covariante). Em outras palavras, esse modelo é utilizado quando alguns atributos podem ser especificados conjuntamente para o cálculo da probabilidade relativa, nesse caso é necessário ter um conhecimento de fundo. Não permite a existência de valores ausentes.

²Essa distribuição também é conhecida como Gaussiana.

5. ignore: esse modelo é utilizado para atributos discretos e contínuos, no qual um ou mais atributos não são considerados na classificação.

No *AutoClass*, os relacionamentos entre as classes são descritos através de probabilidades. Dessa forma, cada exemplo tem uma determinada probabilidade de pertencer a “uma ou algumas” das classes possíveis. Esse relacionamento probabilístico entre classes deve somar 1 a cada exemplo.

Quando um exemplo possui uma probabilidade maior que 0,99 em uma classe, é provável que as classes sejam bem separadas e também bem definidas. Se existe alguns exemplos que possuam uma probabilidade 0,5 em alguma classe, isso significa que algumas classes estão sobrepostas. Neste caso, a combinação de classes nos conjuntos de dados é usualmente mais “difícil” do que considerar cada classe separadamente.

Uma classe é definida como um conjunto particular de valores de parâmetros e modelos associados. Uma classificação é definida como um conjunto de classes e suas respectivas probabilidades. Essa classificação consiste em escolher uma classe-modelo apropriada (ou um conjunto de classe-modelos), procurar boas classificações e finalmente fixar uma probabilidade relativa das classificações obtidas. Ainda não existe métodos realmente aceitos para determinar a probabilidade relativa das classificações obtidas.

O cálculo da probabilidade relativa no *AutoClass* é feita utilizando a função de verossimilhança $P(X|H, p)$ que determina a probabilidade do conjunto de dados X independente condicionalmente dos modelos H e parâmetros p . Um fato que deve ser ressaltado referentes aos valores das verossimilhanças é que as comparações só podem ser realizadas utilizando o mesmo conjunto de dados e além disso, esses valores são apresentados na forma logarítmica.

As classes encontradas arbitrariamente são numeradas de 0 a n , na ordem descendente de influência. Uma classe com valor de influência 34,1 significa que a soma das probabilidades dos exemplos para essa classe é 34,1. Note que uma classe com influência igual a 34 não necessariamente possui 34 exemplos pertencentes àquela classe, desde que muitos exemplos possuem somente um relacionamento parcial daquela classe. Dentro de cada classe, os atributos ou conjuntos de atributos estão ordenados por uma “influência” dos seus modelos.

Usualmente, a classificação é um passo da análise de dados para examinar um conjunto de exemplos verificando se os exemplos podem ser agrupados, ou seja, se esses exemplos são similares. O próximo passo é tentar “explicar” porque alguns exemplos são similares com

um grupo, e em que esses diferem dos outros grupos. Frequentemente, o conhecimento do domínio sugere alguma resposta. Por exemplo, uma classificação de pessoas baseada na sua renda, compras, idade, e moradia entre outras devem revelar classes particulares que não são óbvias antes de uma análise de classificação.

Após um conjunto de dados ser analisado, é provável que muitas classes possuam vários atributos em comum. Por exemplo, num conjunto de dados de animais, as classes cachorro e gato têm alguns atributos que possuem os mesmos valores, tais como: número de patas, possui pêlos, etc.

Embora o *AutoClass* seja designado para a tarefa de *clustering*, o mesmo também permite classificação supervisionada. Isso acontece quando o usuário deseja confirmar se alguns exemplos estão em determinadas classes (Cheeseman et al., 1990).

Descrição do Algoritmo

O algoritmo *AutoClass* cria repetidamente várias classificações aleatórias e então tenta atribuir a cada classificação uma alta probabilidade.

Se o usuário prevê (crê) que o número de classes provável num determinado conjunto de dados seja 10, então é interessante especificar para o algoritmo alguns números de classes possíveis, tais como: 5, 6, ..., 12, ao invés de deixar o próprio algoritmo tentar encontrar as classes existentes.

O algoritmo, em alto nível, que mostra como o *AutoClass* identifica as classes a partir de um conjunto de dados e posteriormente classifica novos exemplos é descrito a seguir:

- **Etapas de Treinamento**

1. Determinar os parâmetros, os modelos e o número de classes. Se a definição do número de classes foi feita pelo usuário então vá para o passo 2. Caso contrário vá para o passo 3.
2. Encontrar as classificações possíveis com o número de classes especificado e atribuir as probabilidades. Vá para o passo 5.
3. Calcular os relacionamentos probabilísticos dos exemplos com as classes identificadas. Repita este passo enquanto existir classes vazias.
 - (a) Para cada classificação encontrada atribuir uma probabilidade.
 - (b) Incrementar o número de classes.

4. Ignorar as classes que estejam vazias.
5. Apresentar algumas das melhores classificações encontradas e seleccionar a que obtiver a maior probabilidade.

- **Etapas de Teste**

1. Para cada exemplo novo faça:
 - (a) Calcular a probabilidade desse exemplo pertencer a cada uma das classes identificadas.
 - (b) Descrever os relacionamentos probabilísticos desse exemplo entre as classes identificadas.

5.4.3 *AutoClass Pro (ACPro)*

O algoritmo *AutoClass Pro* é um *plugin API* baseado nos algoritmos *AutoClass* e *Snob*, e foi desenvolvido pela empresa privada *Ultimode Systems* (Baxter & Oliver, 1997). Esse algoritmo está disponível no software *MinesetTM*.

O *ACPro* tem como objetivo agrupar vários exemplos em *clusters*. Os exemplos possuem determinadas probabilidades de pertencer aos *clusters* encontrados.

A entrada desse algoritmo consiste de: um conjunto de dados na forma de atributo/valor num arquivo “schema” (formato de entrada do software *MinesetTM*); a definição do método a ser utilizado; o número (ou limites) de *clusters*; e outras informações adicionais com relação ao conjunto de dados. Algumas dessas informações são: definição de modelos nos atributos (normal, multinomial e ignore), número máximo de iterações e tempo máximo permitido. O *ACPro* então procura um conjunto de *clusters* que seja altamente provável, de acordo com os conjuntos de dados e modelos especificados. A saída desse algoritmo são os *clusters* encontrados que podem ser visualizados na ferramenta *Cluster Visualizer*.

Como já citado, o algoritmo *ACPro* é baseado no *AutoClass*, com alguns melhoramentos significativos, tais como (Drakos, 1998):

- Usar uma busca iterativa “inteligente” ao invés de uma busca exaustiva na procura de *clusters* (principalmente para conjuntos de dados considerados grandes).
- Detectar atributos irrelevantes no conjunto de dados auxiliando assim a busca rápida e o entendimento dos *clusters* identificados.

No *MinesetTM* versão 2.6 estão disponíveis dois métodos distintos de *clustering* realizado pelo algoritmo *ACPro*:

- Método *ACPro* Manual

Nesse método o usuário define apenas o número de *clusters* desejado.

- Método *ACPro* Automático

Nesse método o usuário define 2 parâmetros: o limite inferior e o limite superior para o número de *clusters*.

Os principais passos do algoritmo *ACPro* para encontrar os *clusters* a partir de um conjunto de dados, em alto nível, são descritos a seguir:

1. Iniciar com um simples *cluster* (todos os exemplos).
2. Para cada *cluster*, tentar dividir o mesmo em dois sub-*clusters* que forneçam melhores descrições dos dados.
3. Se um *cluster* não é útil na análise dos dados, o mesmo será deletado.
4. Estimar os melhores parâmetros para o número de *clusters* corrente.
5. Repetir o passo 2 até que nenhum melhoramento possa ser feito, ou até o tempo máximo de execução tenha sido atingido.

Como já mencionado, os *clusters* encontrados podem ser visualizados automaticamente na ferramenta *Cluster Visualizer*. A ordem em que os atributos são mostrados no *Cluster Viz* é significativa. Por *default*, os atributos são mostrados na ordem de importância para a discriminação dos *clusters*. A ordem de importância dos atributos em cada um dos *clusters* identificados pode ser diferente.

Um fato que deve ser ressaltado é que o algoritmo *ACPro* comprime o tamanho do conjunto de dados para encontrar o número ideal de *clusters* e as características que melhor descrevam os dados (Drakos, 1998).

Alternativamente, o usuário pode aplicar o modelo de *clustering* criado e alguns atributos do conjunto de dados para visualizar os *clusters* identificados na ferramenta *Scatter Visualizer* do *MinesetTM* (ilustrada na Figura 5.5 página 49).

5.4.4 *K-Means*

O *K-Means* é um algoritmo de *clustering* que tem como finalidade agrupar vários exemplos num determinado número de *clusters*. Os agrupamentos dos exemplos são realizados utilizando uma medida de similaridade, tais como: Euclidiana e *Manhattan*. Esse algoritmo também está disponível no software *MinesetTM*.

Para rodar o *K-Means* no software *MinesetTM*, a entrada consiste de: um conjunto de dados na forma de atributo/valor num arquivo “.schema”; a definição do método de *clustering*; o número (ou limites) de *clusters*; e outras informações adicionais como a determinação da medida de similaridade, de pesos para alguns atributos e do número máximo de iterações. O *K-Means* então procura um conjunto de *clusters* que seja altamente provável considerando os dados e o número (ou limites) de *clusters*. A saída desse algoritmo é um modelo de *clustering*, no qual os *clusters* encontrados podem ser visualizados na ferramenta *Cluster Visualizer*.

O *MinesetTM* versão 2.6 fornece dois métodos distintos de *clustering* usando o algoritmo *K-Means*:

- Método *Single K-Means*

O método *single K-Means* é a forma mais simples de *clustering* no *MinesetTM*. O usuário especifica o número desejado de *clusters* (o *default* é igual a 3) e o método agrupa os exemplos para minimizar as dispersões³ em cada *cluster*.

Os passos principais do algoritmo *K-Means* utilizando o método *single* são:

1. Especificar uma quantidade de *clusters*, por exemplo, 5.
2. Os cinco centróides são inicializados em posições aleatórias no espaço. Diferentes escolhas de parâmetros pré-selecionados produzirão diferentes posições.
3. Cada exemplo é associado ao centróide mais próximo. Os centróides são recalculados com base no novo dado. O próximo passo é executado repetidamente até que nenhum melhoramento seja atingido.
4. Se existir exemplos que estejam mais próximos de centróides diferentes dos quais eles estão estabilizados, esses exemplos são movidos para os

³Dispersão refere-se a uma medida de distância entre cada registro e os *clusters* associados.

clusters mais próximos e os centróides são recalculados baseados nos novos dados.

O método *single K-Means* é repetido até nenhum exemplo necessite ser movido ou até que o algoritmo tenha sido executado o número máximo de iterações. O passo 4 é o responsável pela maior parte do tempo de execução do modelo de *clustering*. Aos *clusters* gerados são associados números sequenciais iniciando a partir do valor 1.

Cada modelo de *clustering* criado é avaliado pela média de dispersão de cada *cluster*. Se o número de *clusters* aumenta, a média de dispersão diminui. De qualquer maneira, essa diminuição não é uniforme. Uma vez criado o modelo de *clustering*, o usuário pode visualizar os centróides diretamente utilizando a ferramenta *Cluster Visualizer* do *MinesetTM*.

- Método *Iterative K-Means*

O método *iterative K-Means* é uma extensão mais complexa do método *single K-Means*. Esse método requer que sejam especificados 3 parâmetros: limite inferior (*default* igual a 1), limite superior (*default* igual a 10) e um ponto de escolha (*default* igual a 0,5). O algoritmo determinará o número de *clusters* entre os limites inferior e superior, que seja apropriado para o conjunto de dados. O ponto de escolha é um valor entre 0 e 1, que auxilia na determinação do número de *clusters*. Um ponto de escolha alto sugere que o número de *clusters* seja mais próximo do limite superior, enquanto um ponto de escolha baixo sugere que o número de *clusters* seja mais próximo do limite inferior. É importante ressaltar que o ponto de escolha influencia diretamente na determinação do número de *clusters*.

A tarefa de *clustering* é uma operação computacionalmente intensa e consome bastante tempo num conjunto de dados considerado grande, especialmente quando está sendo executado no modo *iterative K-Means*. Caso o conjunto de dados possua mais de 10.000 exemplos, sugere-se que o *clustering* seja executado com apenas uma amostra desses dados.

Os passos principais do algoritmo *K-Means* utilizando o método iterativo são:

1. Usar o número mínimo de *clusters* para uma execução do método *single K-Means*. Isso produzirá um *clustering* inicial. Os próximos passos são

executados repetidamente até que o número máximo de *clusters* seja atingido.

2. Encontrar um *cluster* com a maior dispersão e dividi-lo ao meio, criando dois novos *clusters*. Recalcular os centróides dos *clusters* baseados nos dados que eles contém.
3. Se existir algum exemplo que esteja mais próximo de um *cluster* diferente do que ele está atualmente, esse exemplo é movido para o *cluster* mais próximo. Os centróides dos *cluster* são então recalculados baseados nos novos dados de cada *cluster*.

Esse método tem como finalidade encontrar um número de *cluster* que esteja entre os limites (inferior e superior) definidos. Como já foi mencionado, cada modelo de *clustering* criado é avaliado pela média de dispersão de cada *cluster*. Nesse método, o ponto de escolha seleciona o grau desejado de dispersão, medido como a proporção de uma dispersão entre o número mínimo e máximo de *clusters*.

Os *clusters* gerados são nomeados com base na derivação dos mesmos durante o processo de divisão. O *clustering* inicial (baseado no número mínimo de *clusters*) é nomeado usando o número sequencial como no *single K-Means*. Toda vez que um *cluster* é dividido, os dois *clusters* novos são nomeados com base no *cluster* que foi dividido e acrescidos da letra A e B. Por exemplo, os *clusters* 2-A e 2-B foram derivados do *cluster* 2 no *clustering* inicial.

Um fator importante nos modelos de *clustering* é que a medida de dispersão nos *clusters* não são imediatamente comparáveis quando se determina conjuntos de dados ou números de *clusters* diferentes. Com o propósito de descobrir modelos de *clustering* mais entendíveis, o usuário pode determinar pesos (valor numérico) nos atributos dos conjuntos de dados. Esses pesos definem que os atributos tenham uma influência maior ou menor para a realização de *clustering*.

Embora já comentado, o usuário pode também aplicar o modelo de *clustering* criado pelo *K-Means* para visualizar os *clusters* identificados na ferramenta *Scatter Visualizer* do *MinesetTM* (ilustrada na Figura 5.5 página 49).

5.5 Considerações Finais

Neste capítulo foram abordados brevemente as ferramentas utilizadas neste trabalho e alguns algoritmos estatísticos. A escolha desses algoritmos foi motivada por duas razões: a primeira, por estarem disponíveis no Laboratório de Inteligência Computacional do ICMC-USP, onde foram realizados todos os experimentos e a segunda por adotarem várias estratégias para a busca no espaço de hipóteses.

Em muitas aplicações, a precisão do algoritmo *Naive Bayes* é igual ou maior a dos mais sofisticados algoritmos de Aprendizado de Máquina (Pazzani, 1995).

Os métodos Bayesianos podem ser usados para determinar a hipótese mais provável a partir dos dados - hipótese de máxima probabilidade, isto é, a hipótese considerada ótima. O classificador bayes combina as predições de todas as hipóteses, com o peso da sua probabilidade incondicional, para calcular a classificação mais provável de cada novo exemplo.

Foram vistos também os parâmetros e modelos que podem ser utilizados para melhorar o desempenho da classificação. As saídas geradas pelo algoritmo *AutoClass* dependem diretamente da execução de cada uma das fases. Relembrando que algumas dessas saídas são de difícil compreensão para os usuários. É importante ressaltar que o *AutoClass* é um algoritmo bastante conhecido e que está em constante atualização (novas versões) pelos pesquisadores da área.

O tempo de processamento é diretamente linear à quantidade de dados (Cheeseman & Stutz, 1996). Uma das suposições mais importante do *AutoClass* é o uso de modelos para variáveis reais. Vale lembrar também que, ao contrário de outros métodos, o algoritmo *AutoClass* expressa relacionamentos probabilísticos entre as classes.

Para maiores informações dos algoritmos *Naive Bayes* e *AutoClass*, encontra-se disponível um relatório técnico sobre esses algoritmos Bayesianos (Padilha & Rezende, 1999).

Um fator observado nas descrições dos algoritmos estatísticos citados é a variedade de métodos que eles trabalham. Na Tabela 5.1 é apresentado um resumo contendo alguns algoritmos estatísticos com suas principais características, o tipo de acesso (domínio público ou comercial), bem como algumas referências bibliográficas/e-mail para contato. Uma característica comum nos algoritmos apresentados é o uso da plataforma Unix.

Nome do Algoritmo	Principais Características	Tipo de Acesso	Referências/ Contato
<i>Naive Bayes</i>	teoria da probabilidade	comercial	(Kohavi, 1996) <i>MLC++</i>
<i>AutoClass</i>	teoria da probabilidade definição de modelos	domínio público	(Cheeseman & Stutz, 1996) cheesem@ptolemy.arc.nasa.gov
<i>ACPro</i>	teoria da probabilidade definição de modelos <i>plugin API</i>	comercial	(Baxter & Oliver, 1997) sales@ultimode.com
<i>K-Means</i>	medidas de distâncias	comercial	(Hartigan & Wong, 1978) (Hastie & Tibshirani, 1995)
Vizinho Mais Próximo	medidas de distâncias	comercial	<i>MLC++</i>

Tabela 5.1: Levantamento de Alguns Algoritmos Estatísticos.

O próximo capítulo mostra algumas avaliações experimentais de alguns desses algoritmos utilizando dois conjuntos de dados diferentes.

Capítulo 6

Avaliação Experimental de Alguns Algoritmos Estatísticos

6.1 Considerações Iniciais

Neste capítulo são realizadas algumas avaliações experimentais usando os algoritmos estatísticos apresentados no capítulo 5: *Naive Bayes*, *AutoClass*, *AutoClass Pro* e *K-Means*. Estas avaliações têm como objetivo principal investigar esses algoritmos realizando dois estudos de casos com características distintas. No primeiro estudo de casos é utilizado um conjunto de dados de plantas iris e no segundo é utilizado um conjunto de dados de domicílios de clientes de uma seguradora¹. Vale ressaltar que não é o objetivo deste trabalho concluir qual é o melhor algoritmo estatístico para cada um dos conjuntos de dados selecionados.

No estudo de casos usando o conjunto de dados iris foram usados os 4 algoritmos descritos e no outro estudo de casos foram utilizados somente os algoritmos: *AutoClass*, *AutoClass Pro* e *K-Means* visando encontrar *clusters* no conjunto de dados seguros.

A seguir, é apresentada a descrição dos 2 conjuntos de dados utilizados e as avaliações experimentais desses conjuntos usando os algoritmos citados.

¹Apartir deste ponto esse conjunto de dados será identificado como conjunto de dados seguros.

6.2 Descrição dos Conjuntos de Dados Utilizados

Para a avaliação experimental deste trabalho foram escolhidos 2 conjuntos de dados. Um conjunto de dados que os atributos-classes são conhecidos e outro conjunto de dados que os atributos-classes são desconhecidos.

Os conjuntos de dados utilizados foram divididos em três sub-conjuntos com a finalidade de descrever as possíveis diferenças de comportamento de cada um desses sub-conjuntos nas investigações com os algoritmos estatísticos. Para cada sub-conjunto de dados criado usou-se 2/3 dos exemplos para o conjunto de treinamento e 1/3 dos exemplos para o conjunto de teste (*3-fold cross validation*). Nas investigações experimentais, os três sub-conjuntos criados de cada conjunto de dados são identificados por Dados1, Dados2 e Dados3.

É importante relembrar que nos algoritmos *AutoClass*, *ACPro* e *K-Means* os conjuntos de teste são ignorados, uma vez que esses algoritmos trabalham apenas com os conjuntos de treinamento. Os conjuntos de dados para os dois estudos de casos selecionados são descritos a seguir.

6.2.1 Conjunto de Dados Iris

O conjunto de dados iris foi retirado do repositório de Bases de Dados para Aprendizado de Máquina da *University of California at Irvine (UCI)*, e envolve a análise de várias espécies de plantas iris.

Esse conjunto de dados é composto de 150 exemplos com 4 atributos e um atributo-classe (setosa, versicolor e virginica). Algumas características desse conjunto de dados, tais como: nome, descrição e tipo do atributo pode ser visualizada na Tabela 6.1. Vale a pena salientar que esse conjunto de dados possui uma distribuição uniforme entre os atributos-classes existentes (50 exemplos para cada atributo-classe) e não possui valores ausentes.

6.2.2 Conjunto de Dados Seguros

O conjunto de dados seguros foi preparado pelo grupo de Pesquisa de Sistema de Informação da *Swiss Life* — uma das maiores companhias do ramo de seguros em toda a Europa.

Esse conjunto de dados consiste de vários arquivos no formato ASCII, cada qual contendo registros na forma “*Prolog-like*” (parecidos com a representação utilizada pela

Nome do Atributo	Descrição do Atributo	Tipo do Atributo
<i>sepal_lenght (s_l)</i>	Comprimento da sépala	Real
<i>sepal_width (s_w)</i>	Largura da sépala	Real
<i>petal_lenght (p_l)</i>	Comprimento da pétala	Real
<i>petal_width (p_w)</i>	Largura da pétala	Real
<i>class (class)</i>	Tipo da planta iris	String

Tabela 6.1: Descrição dos Atributos do Conjunto de Dados Iris.

linguagem *Prolog*) e organizados em uma tupla por linha. A organização dos registros é da seguinte maneira:

$\langle \text{relação} \rangle (\text{componente1}, \text{componente2}, \dots, \text{componenteN})$

no qual, a $\langle \text{relação} \rangle$ representa o nome de uma relação existente no conjunto de dados formada por vários componentes representados por *componente1*, *componente2*, ..., *componenteN*.

Os arquivos que formam o conjunto de dados possuem registros de relações específicas. Existem no total dez relações que descrevem os clientes ou parceiros da *Swiss Life*, endereços, contratos de seguros, apólices de seguros, domicílios e componentes de tarifas de seguros. As relações existentes no conjunto de dados são:

- PART - relação que representa os clientes da *Swiss Life*.
- EADR - relação que contém endereços eletrônicos de clientes.
- PADR - relação que contém endereços postais de clientes.
- HHOLD - relação que armazena os dados sobre os domicílios dos clientes.
- VVERT - relação que representa a apólice de seguro.
- PARROL - relação que representa o papel que os clientes fazem com relação a uma determinada apólice de seguros.
- TFROL - relação que representa os papéis de tarifa e especifica maiores informações caso um cliente seja uma pessoa segurada.

- TFKOMP - relação que está associada aos papéis de clientes.
- PROD - relação que representa os produtos.
- LVTARF - relação que representam tarifas dedicadas de seguro.

A companhia *Swiss Life* descreveu três problemas possíveis para serem atacados utilizando os algoritmos de Aprendizado de Máquina. Um dos problemas, *TaskC*, trata-se da busca de agrupamentos (*clusters*) existentes na relação HHOLD, que possui os dados referentes aos domicílios dos clientes da seguradora e é o objeto de trabalho neste estudo de caso. O conjunto de dados disponibilizado possui 340.573 exemplos distribuídos nas relações existentes. O número de exemplos em cada relação, bem como a quantidade de atributos são mostrados na Tabela 6.2.

Relação	Nº de Exemplos	Quant. de Atributos
VVERT	34.986	18
PARROL	111.077	5
PART	17.267	9
HHOLD	12.934	43
EADR	505	3
PADR	17.970	4
TFKOMP	73.502	26
PROD	184	3
TFROL	73.332	8
LVTARF	239	9
Total	340.573	128

Tabela 6.2: Número de Exemplos Parcial e Total do Conjunto de Dados Seguros.

Na Tabela 6.3 são apresentadas algumas características da relação HHOLD, tais como: nome, descrição e tipo do atributo. Um fator que deve ser ressaltado é que esse conjunto de dados possui valores ausentes para alguns atributos.

Nome do Atributo	Descrição do Atributo	Tipo do Atributo
HHID	Identificador de domicílio	Real
Atributo2..11	Sem descrição (para proteger informações privadas)	String
Atributo12..15	Sem descrição (para proteger informações privadas)	Real
Atributo16..19	Sem descrição (para proteger informações privadas)	String
Atributo20..21	Sem descrição (para proteger informações privadas)	Real
Atributo22	Sem descrição (para proteger informações privadas)	String
Atributo23..27	Sem descrição (para proteger informações privadas)	Real
Atributo28..30	Sem descrição (para proteger informações privadas)	String
Atributo31..43	Sem descrição (para proteger informações privadas)	Real

Tabela 6.3: Descrição dos Atributos da Relação HHOLD.

6.3 Investigação dos Algoritmos Utilizando o Conjunto de Dados Iris

Nessa avaliação experimental, o objetivo principal é investigar os algoritmos *Naive Bayes*, *AutoClass*, *AutoClass Pro* e *K-Means* para verificar como esses algoritmos se comportam utilizando um conjunto de dados com atributos-classes definidos.

Com a finalidade de observar se os algoritmos de *clustering* utilizados nessa investigação reproduzem os atributos-classes do conjunto de dados foram realizadas duas análises. Na primeira análise, o atributo-classe foi considerado como um atributo qualquer e na segunda análise o atributo-classe foi eliminado dos dados. Essas duas análises tem como propósito verificar se os *clusters* encontrados pelos algoritmos são semelhantes aos atributos-classes especificados no conjunto de dados original. Além das avaliações com os três sub-conjuntos de dados, os algoritmos de *clustering* também são executados com o conjunto de dados completo.

6.3.1 *Naive Bayes*

O algoritmo *Naive Bayes* constrói um modelo no qual assume que as probabilidades de cada valor de atributo são independentes dado o atributo-classe. Como esse algoritmo está

implementado na biblioteca *MLC++* (descrita na seção 5.3 página 49), o usuário tem várias opções para definir o tipo de saída. Algumas saídas são: a matriz de confusão, o espaço de busca que pode ser visualizado usando o utilitário *Dotty* da própria *MLC++* e o modelo de probabilidades que pode ser mostrado pela ferramenta *Evidence Visualizer* do *MinesetTM*. Uma representação da matriz de confusão para problemas que envolvem três atributos-classes pode ser vista na Tabela 6.4.

Atr-Classes Definidos	Atr-Classe1	Atr-Classe2	Atr-Classe3
Atr-Classe1	<i>a</i>	<i>b</i>	<i>c</i>
Atr-Classe2	<i>d</i>	<i>e</i>	<i>f</i>
Atr-Classe3	<i>g</i>	<i>h</i>	<i>i</i>

Tabela 6.4: Matriz de Confusão com Três Atributos-Classes.

no qual,

- *a* representa o número de exemplos pertencentes ao atributo-classe1, corretamente classificados;
- *b* e *c* representa o número de exemplos pertencentes ao atributo-classe1 incorretamente classificados como pertencentes aos outros atributos-classes;
- *d* representa o número de exemplos pertencentes ao atributo-classe2, corretamente classificados;
- *e* e *f* representa o número de exemplos pertencentes ao atributo-classe2 incorretamente classificados como pertencentes aos outros atributos-classes;
- *g* representa o número de exemplos pertencentes ao atributo-classe3, corretamente classificados;
- *h* e *i* representa o número de exemplos pertencentes ao atributo-classe3 incorretamente classificados como pertencentes aos outros atributos-classes.

No decorrer deste capítulo os símbolos *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h* e *i* são utilizados para descrever as taxas de erros nos atributos-classes e nos três sub-conjuntos de dados, e os Atr-Classes Definidos Atr-Classe1, Atr-Classe2 e Atr-Classe3 são considerados como iris-setosa, iris-versicolor e iris-virginica, respectivamente.

Como foi dito anteriormente, para as avaliações foram utilizados três sub-conjuntos de dados. Dessa forma, com as três matrizes de confusão é possível encontrar as taxas de erro para cada atributo-classe e a taxa de erro do sub-conjunto de dados. Na Tabela 6.5 são apresentados como esses valores são obtidos.

Atr-Classes Definidos	Atr-Classes Predizidos			Taxa de Erro no Atr-Classe	Taxa de Erro no Sub-conjunto
	X	Y	Z		
X	a	b	c	$\frac{b+c}{a+b+c}$	$\frac{b+c+d+f+g+h}{a+b+c+d+e+f+g+h+i}$
Y	d	e	f	$\frac{d+f}{d+e+f}$	
Z	g	h	i	$\frac{g+h}{g+h+i}$	

Tabela 6.5: Matriz de Confusão e Taxas de Erro de Classificação.

As matrizes de confusão, as taxas de erro de cada atributo-classe e a taxa de erro dos três sub-conjuntos de dados obtidas com o algoritmo *Naive Bayes* são exibidas na Tabela 6.6.

Sub-conjunto de Dados	Matriz de Confusão	Taxas de Erro no Atr-Classe	Taxa de Erro no Sub-conjunto
Dados1	15.0 0.0 0.0	0,0%	0,063%
	0.0 15.0 2.0	0,133%	
	0.0 1.0 17.0	0,058%	
Dados2	14.0 0.0 0.0	0,0%	0,08%
	0.0 14.0 4.0	0,285%	
	0.0 0.0 18.0	0,0%	
Dados3	21.0 0.0 0.0	0,0%	0,06%
	0.0 14.0 1.0	0,071%	
	0.0 2.0 12.0	0,166%	

Tabela 6.6: Matriz de Confusão e Taxas de Erro dos Três Sub-conjuntos de Dados Iris.

Baseados nas taxas de erro no atributo-classe (Tabela 6.6) pode-se observar que em todos os sub-conjuntos a classe iris-setosa teve uma taxa de erro igual a 0,0%. Além disso, pode-se verificar que as taxas de erro dos três sub-conjuntos de dados tiveram uma pequena variação e o sub-conjunto Dados3 apresentou uma precisão um pouco melhor em vista das

demais.

Como já comentado, as ferramentas de visualização cada vez mais estão sendo importantes para análises e interpretações dos resultados obtidos pelos algoritmos de Aprendizado de Máquina. Dentre os três sub-conjuntos de dados, será explorado apenas o sub-conjunto Dados1. Para explorar o sub-conjunto Dados1 será usado o utilitário *Dotty* da biblioteca *MLC++* e algumas ferramentas de visualização do *MinesetTM*.

Com o uso do utilitário *Dotty* foi possível visualizar alguns atributos relevantes do sub-conjunto Dados1 baseados nos pesos dos nós do espaço de busca encontrado pelo algoritmo *Naive Bayes*, como mostrado na Figura 6.1. Nesse espaço de busca existem 14 nós, que possuem várias combinações dos atributos existentes. Em geral, a estimativa de precisão apresentada pelos nós tem uma pequena variação, mas naqueles nós que possuem a precisão real (calculada com o conjunto de teste) o valor é igual para todos. O nó em destaque representa o atributo 3 *petal_width* que é considerado o mais relevante entre os demais, sua precisão é 0.00%+- 0.85% (calculado com o conjunto de treinamento) e a precisão real é 6.00% +- 3.39% (calculado com o conjunto de teste).

Figura 6.1: *Dotty* do Sub-conjunto Dados1 gerado pelo *Naive Bayes*.

Estudos realizados mostram que melhores resultados são comumente alcançados discretizando os atributos contínuos (Dougherty et al., 1995). Na própria biblioteca *MLC++*,

existe um indutor chamado **disc-naive-bayes** que faz esse pré-processamento. Utilizando este indutor foi gerado um modelo de distribuição dos atributos-classes existentes que pode ser visualizado no *Evidence Visualizer* do *Mineset*TM.

Com a utilização da ferramenta *Evidence Visualizer* foi possível detectar alguns casos interessantes baseados nas importâncias dos valores de atributos do sub-conjunto Dados1 para a classificação. Um desses casos pode ser visualizado na Figura 6.2. Nesse caso, uma planta com *sepal_lenght* > 5,55 polegadas e *petal_width* > 1,75 polegadas possui uma probabilidade de pertencer ao atributo-classe iris-virginica igual a 97,55%. Os usuários podem interagir com a visualização selecionando alguns valores de atributos (torta da esquerda) e observando como a probabilidade condicional dos atributos-classes muda (torta da direita).

Figura 6.2: *Evidence Visualizer* do Sub-conjunto Dados1 gerado pelo *Naive Bayes*.

6.3.2 *AutoClass*

Nas investigações realizadas com o algoritmo *AutoClass* para o conjunto de dados iris foi especificado que o número de *clusters* a ser encontrado seja igual a 3 e o modelo para os atributos reais seja *single_normal_cn* (visto na seção 5.4.2).

Duas análises foram realizadas com o algoritmo *AutoClass*, a primeira considerando o atributo-classe (*class*) como um atributo normal e a segunda eliminando o mesmo da descrição dos dados. Como descrito na seção 5.4.2, cada *cluster* encontrado pelo *AutoClass* pos-

sui um descrição dos exemplos pertencentes e também um determinado valor de influência. Vale lembrar que o valor de influência de um *cluster* é a soma total das probabilidades dos exemplos e não a quantidade de exemplos que o *cluster* cobre.

Em uma primeira análise foi observado, nos relatórios gerados pelo *AutoClass*, a descrição probabilística dos exemplos nos três *clusters* encontrados, o valor de influência de cada *cluster* e os três atributos mais relevantes para cada sub-conjunto e também para o conjunto de dados completo. Algumas dessas informações podem ser visualizadas na Tabela 6.7².

Sub-conjunto de Dados	Nº do <i>Cluster</i>	Valor de Influência com Atr.-classe	Atributos Mais Relevantes
Dados1	0	35	<i>log p-l</i>
	1	34	<i>log p-w</i>
	2	31	<i>class</i>
Dados2	0	36	<i>log p-l</i>
	1	33	<i>log p-w</i>
	2	31	<i>class</i>
Dados3	0	29	<i>log p-w</i>
	1	36	<i>log p-l</i>
	2	35	<i>class</i>
Completo	0	50	<i>log p-l</i>
	1	52	<i>log p-w</i>
	2	48	<i>class</i>

Tabela 6.7: Informações dos *Clusters* Encontrados pelo *AutoClass* com o Atributo-classe.

Analisando os atributos mais relevantes, pode-se observar que os atributos baseados tanto na largura quanto no comprimento da pétala (*log p-w* e *log p-l*) são mais importantes que o atributo-classe nos três sub-conjuntos e no conjunto de dados completo. Pode-se concluir nessa avaliação que o *AutoClass* prefere trabalhar com atributos reais com as suas devidas distribuições do que o atributo discreto para a identificação de *clusters*.

Na Figura 6.3³ é ilustrado parcialmente, o relatório com a descrição dos três *clusters*

²Considerando que o *cluster* 0 corresponde a iris-setosa, 1 a iris-versicolor e 2 a iris-virginica.

³Nos relatórios do algoritmo *AutoClass* os *clusters* identificados são representados por CLASS.

encontrados no sub-conjunto Dados1. Foi observado nos relatórios que todos os exemplos (como é o caso dos exemplos de número 2, 5 e 8) que pertencem ao *cluster* 0 (iris-setosa) do sub-conjunto Dados1 obtiveram uma probabilidade igual a 1. Além disso, foi notado que os *clusters* 1 (iris-versicolor) e 2 (iris-virginica) possuem vários exemplos que tem probabilidades de pertencer a outros *clusters* simultaneamente. Observando o *cluster* 1, pode-se verificar que o exemplo número 1 (em negrito na Figura 6.3) possui uma probabilidade de pertencer a esse *cluster* igual a 0,995 e uma probabilidade de pertencer ao *cluster* 2 de 0,005.

CLASS = 0						
Case #	sepal-length	sepal-width	petal-length	petal-width	class	(Cls Prob)
2	5.8	4	1.2	0.2	Iris-setosa	1.000
5	5	3.4	1.5	0.2	Iris-setosa	1.000
8	4.4	2.9	1.4	0.2	Iris-setosa	1.000
CLASS = 1						
Case #	sepal-length	sepal-width	petal-length	petal-width	class	(Cls Prob)
1	6.3	3.3	4.7	1.6	Iris-versicolor	0.995
					2	0.005
4	6.1	2.8	4	1.3	Iris-versicolor	1.000
6	6.2	2.2	4.5	1.5	Iris-versicolor	1.000
CLASS = 2						
Case #	sepal-length	sepal-width	petal-length	petal-width	class	(Cls Prob)
3	6.7	3.3	5.7	2.1	Iris-virginica	1.000
7	7.2	3.6	6.1	2.5	Iris-virginica	1.000
11	6.3	2.8	5.1	1.5	Iris-virginica	0.871
					1	0.129

Figura 6.3: Relatório Parcial do Sub-conjunto Dados1 gerado pelo *AutoClass* com o Atributo-classe.

Na segunda análise do algoritmo *AutoClass* foi utilizado o conjunto de dados iris ignorando o atributo-classe e foram consideradas as mesmas especificações quanto ao número de *clusters* e o modelo do atributo real da primeira análise. Os valores de influência e os atributos mais relevantes em cada sub-conjunto e no conjunto de dados completo sem o atributo-classe pode ser observado na Tabela 6.8.

Comparando os resultados obtidos pelo algoritmo *AutoClass* das duas análises (com e sem o atributo-classe) foi observado que o número de exemplos pertencentes ao *cluster* 0 nos três sub-conjuntos e no conjunto de dados completo foram iguais. E com relação aos atributos mais relevantes apenas no sub-conjunto Dados2 houve uma semelhança com a análise anterior.

Observando os relatórios gerados pelo algoritmo *AutoClass*, ignorando o atributo-

Sub-conjunto de Dados	Nº do <i>Cluster</i>	Valor de Influência sem Atr.-classe	Atributos Mais Relevantes
Dados1	0	35	<i>log p-w</i>
	1	40	<i>log p-l</i>
	2	25	<i>log s-l</i>
Dados2	0	36	<i>log p-l</i>
	1	32	<i>log p-w</i>
	2	32	<i>log s-l</i>
Dados3	0	29	<i>log p-l</i>
	1	48	<i>log p-w</i>
	2	23	<i>log s-l</i>
Completo	0	50	<i>log s-l</i>
	1	39	<i>log p-l</i>
	2	61	<i>log p-w</i>

Tabela 6.8: Informações dos *Clusters* Encontrados pelo *AutoClass* sem o Atributo-classe.

classe, foi constatado que os três sub-conjuntos e o conjunto de dados completo conseguiram distinguir bem o *cluster* 0 (iris-setosa) fornecendo uma probabilidade igual a 1 para todos os exemplos pertencentes (as primeiras linhas em negrito na Figura 6.4) . Essa avaliação foi realizada com o auxílio do conjunto de dados completo que possuem o atributo-classe. Na Figura 6.4 é ilustrado parcialmente o relatório dos *clusters* encontrados no sub-conjunto Dados1 ignorando o atributo-classe. Sob o aspecto de avaliar a identidade dos *clusters* encontrados, um cuidado especial deve ser tomado devido ao fato que o algoritmo *AutoClass* trabalha com valores de influência.

6.3.3 *AutoClass Pro*

Para verificar o comportamento do algoritmo *AutoClass Pro* com os três sub-conjuntos e o conjunto de dados completo iris, foi especificado o método de busca manual e o número de *clusters* igual a 3. Esse algoritmo foi executado com os parâmetros *default* do software *MinesetTM*.

Na Tabela 6.9 são exibidos os *clusters* encontrados, juntamente com o número de

CLASS = 0					
Case #	sepal-length	sepal-width	petal-length	petal-width	(Cls Prob)
2	5.8	4	1.2	0.2	1.000
5	5	3.4	1.5	0.2	1.000
8	4.4	2.9	1.4	0.2	1.000
CLASS = 1					
Case #	sepal-length	sepal-width	petal-length	petal-width	(Cls Prob)
1	6.3	3.3	4.7	1.6	0.973
4	6.1	2.8	4	1.3	0.027
6	6.2	2.2	4.5	1.5	1.000
CLASS = 2					
Case #	sepal-length	sepal-width	petal-length	petal-width	(Cls Prob)
3	6.7	3.3	5.7	2.1	1.000
7	7.2	3.6	6.1	2.5	1.000
21	6.5	3.2	5.1	2	0.982
					0 0.018

Figura 6.4: Relatório do Sub-conjunto Dados1 gerado pelo *AutoClass* sem o Atributo-classe.

exemplos pertencentes e os atributos mais relevantes para cada *cluster* considerando o atributo-classe. Pode-se observar que foram identificados *clusters*, que possuem um número bastante reduzido de exemplos, como é o caso do *cluster* 1 do sub-conjunto Dados1.

Como visto na seção 5.4.3 página 64, o algoritmo *ACPro* reduz o tamanho do conjunto de dados para obter uma melhor descrição dos *clusters* encontrados. Baseados nesse conceito pode-se verificar que em todos os sub-conjuntos e o conjunto de dados completo ocorreu esta redução. Particularmente, no sub-conjunto Dados1 ocorreu uma redução de 4 exemplos dos 100 disponíveis.

Uma ilustração dos *clusters*, identificados no sub-conjunto Dados1 considerando o atributo-classe, na ferramenta *Cluster Visualizer* pode ser vista na Figura 6.5. Como mencionado na seção 5.2, a primeira coluna da ferramenta *Cluster Visualizer* (Population) descreve uma estatística dos valores dos atributos do conjunto de dados. Observando essa figura foi notado que a média dos valores do atributo *sepal-length* no *cluster* 1 é 4,9, no *cluster* 2 é 5,0 e no *cluster* 3 é 6,37.

Sub-conjunto de Dados	Nº do <i>Cluster</i>	Nº de Exemplos com Atr.-classe	Atributos Mais Relevantes
Dados1	1	01	<i>class, p_l e p_w</i>
	2	33	<i>s_w, s_l e class</i>
	3	62	<i>s_w, s_l e p_w</i>
Dados2	1	34	<i>class, s_w e s_l</i>
	2	59	<i>s_w, s_l e p_w</i>
	3	02	<i>class, s_w e p_l</i>
Dados3	1	28	<i>class, s_w e s_l</i>
	2	34	<i>s_l, s_w e class</i>
	3	33	<i>s_w, s_l e class</i>
Completo	1	48	<i>s_w, s_l e class</i>
	2	86	<i>s_w, s_l e p_w</i>
	3	10	<i>s_w, class e p_w</i>

Tabela 6.9: Informações dos *Clusters* Encontrados pelo *ACPro* com o Atributo-classe.

Na maioria das vezes uma visualização dos dados, num espaço multidimensional, auxilia o entendimento do domínio, mas é muito importante possuir um conhecimento prévio da aplicação. Uma maneira mais clara de observar os *clusters* encontrados no sub-conjunto Dados1 pode ser utilizando a ferramenta *Scatter Visualizer* do *MinesetTM*.

Embora já tenha sido identificados os atributos mais relevantes de cada *cluster* no sub-conjunto Dados1 (Tabela 6.9), também foi utilizada a ferramenta *Column Importance* do *MinesetTM* para detectar os três atributos mais relevantes do sub-conjunto Dados1, uma vez que essa ferramenta é bastante útil para esse tipo de aplicação. Dentre os atributos do sub-conjunto Dados1, a ferramenta *Column Importance* determinou um atributo mais relevante *class* com o grau de pureza igual a 100.

Após definido o atributo mais relevante, o atributo *class*, o mesmo foi associado para o eixo 1. Para os demais eixos foram associados o atributo *sepal_width* para o eixo 2 e *sepal_lenght* para o eixo 3 por serem considerados relevantes para alguns *clusters* do sub-conjunto Dados1. Para representar a cor dos *clusters* identificados foi associado o modelo de

Figura 6.5: *ClusterViz* do Sub-conjunto Dados1 gerado pelo *ACPro* com o Atributo-classe.

clustering fornecido pelo algoritmo *AutoClass Pro*. Na Figura 6.6 são mostrados os *clusters* identificados do sub-conjunto Dados1 considerando o atributo-classe utilizando a ferramenta *Scatter Visualizer*.

Os exemplos representados pela cor verde pertencem ao *cluster* 1 (*iris-setosa*), os de cor vermelha pertencem ao *cluster* 2 (*iris-versicolor*) e de cor azul pertencem ao *cluster* 3 (*iris-virginica*). O usuário pode clicar num exemplo que o considera interessante e obter algumas informações desse exemplo, tal como: os valores dos atributos e o *cluster* pertencente.

Depois da análise dos *clusters* considerando o atributo-classe como um atributo normal, foi analisado como os três sub-conjuntos e o conjunto de dados completo se comportaram quando o mesmo foi eliminado. Na Tabela 6.10 é apresentada uma descrição geral desta análise mostrando o número de exemplos pertencentes e os atributos mais relevantes em cada *cluster*. Fazendo uma comparação com a primeira análise (Tabela 6.9) foi observado que mesmo eliminando o atributo-classe, o algoritmo identificou o mesmo número de exemplos pertencentes ao *cluster* 1 nos sub-conjuntos e o conjunto de dados completo, exceto para o sub-conjunto Dados2 que obteve o mesmo resultado em todos os *clusters*. Mesmo que o atributo-classe seja considerado um dos atributos mais relevantes para alguns *clusters* na primeira análise a ausência do mesmo não prejudicou a delineação desses *clusters*. Com relação a determinação dos atributos mais relevantes, pode-se notar que todos os atributos

Figura 6.6: *Scatter Visualizer* do Sub-conjunto Dados1 gerado pelo *ACPro* com o Atributo-classe.

foram considerados os mais relevantes pelo menos uma vez nos conjuntos de dados.

Os *clusters* identificados e os três atributos mais relevantes do sub-conjunto Dados1 ignorando o atributo-classe podem ser visualizados na ferramenta *Cluster Visualizer*, como mostrada na Figura 6.7.

Com a finalidade de observar como os *clusters* estão distribuídos, a ferramenta *Column Importance* foi executada para indicar os três atributos mais relevantes. Os valores que estão entre parênteses indicam o grau de pureza do atributo.

- *petal_lenght* (84,24)
- *sepal_lenght* (91,61)
- *petal_width* (97,48)

Para visualizar os *clusters* identificados foi associado o atributo *sepal_lenght* para o eixo 1, *sepal_lenght* para o eixo 2, *petal_width* para o eixo 3 e o modelo de *clustering* gerado pelo *ACPro* sem o atributo-classe para representar a cor dos *clusters* na ferramenta *Scatter Visualizer*, como mostrado na Figura 6.8.

Os exemplos representados pela cor verde pertencem ao *cluster* 1, os de cor vermelha pertencem ao *cluster* 2 e de cor azul pertencem ao *cluster* 3. Observando os valores dos

Sub-conjunto de Dados	Nº do <i>Cluster</i>	Nº de Exemplos sem Atr.-classe	Atributos Mais Relevantes
Dados1	1	01	<i>s_w</i> , <i>p_l</i> e <i>p_w</i>
	2	33	<i>s_w</i> , <i>s_l</i> e <i>p_w</i>
	3	61	<i>s_w</i> , <i>s_l</i> e <i>p_w</i>
Dados2	1	34	<i>s_w</i> , <i>p_l</i> e <i>p_w</i>
	2	59	<i>s_w</i> , <i>s_l</i> e <i>p_w</i>
	3	02	<i>s_w</i> , <i>p_l</i> e <i>p_w</i>
Dados3	1	28	<i>s_w</i> , <i>s_l</i> e <i>p_w</i>
	2	50	<i>p_l</i> , <i>s_w</i> e <i>p_w</i>
	3	18	<i>s_w</i> , <i>s_l</i> e <i>p_w</i>
Completo	1	48	<i>s_w</i> , <i>s_l</i> e <i>p_w</i>
	2	53	<i>p_w</i> , <i>s_l</i> e <i>p_w</i>
	3	44	<i>s_l</i> , <i>s_w</i> e <i>s_l</i>

Tabela 6.10: Informações dos *Clusters* gerados pelo *ACPro* sem o Atributo-classe.

atributos do único exemplo pertencente ao *cluster* 1 foi constatado que esse exemplo pertence a classe iris-versicolor baseado no sub-conjunto de dados que possui o atributo-classe.

6.3.4 *K-Means*

Nas investigações experimentais do algoritmo *K-Means* com o conjunto de dados iris foi especificado o método *single*, o número de *clusters* igual a 3 e como medida de similaridade a distância Euclidiana. Para as execuções feitas com esse algoritmo foram utilizados os parâmetros *default* do software *MinesetTM*.

Na Tabela 6.11 são mostrados os três *clusters* encontrados com o número de exemplos, o grau de dispersão, bem como os atributos mais relevantes de cada *cluster* dos três sub-conjuntos e do conjunto de dados completo. Nesse algoritmo foi observado que o atributo discreto (*class*) teve uma relevância considerada absoluta. Com relação ao grau de dispersão de cada *cluster* encontrado, o conjunto de dados completo possui o menor grau de dispersão entre os exemplos (destacado na Tabela 6.11).

Na Figura 6.9 são mostrados parcialmente os *clusters* encontrados com o sub-conjunto

Figura 6.7: *ClusterViz* do Sub-conjunto Dados1 gerado pelo *ACPro* sem o Atributo-classe.

Dados1 utilizando a ferramenta *ClusterViz*. Como já foi dito anteriormente, quando se trabalha com algoritmos de *clustering* no *MinesetTM* o usuário pode visualizar os atributos mais proeminentes do conjunto de dados, bem como algumas informações adicionais de cada *cluster* gerado.

Uma outra maneira de visualizar os *clusters* encontrados no sub-conjunto Dados1 pode ser utilizando a ferramenta *Scatter Visualizer* do *MinesetTM*. Como já foi determinado o atributo mais relevante do sub-conjunto Dados1 na investigação do algoritmo *AutoClass Pro* (seção 6.3.3), o mesmo foi associado para o eixo 1. Para os outros eixos foram associados o atributo *petal_width* para o eixo 2 e *petal_lenght* para o eixo 3. Para representar a cor dos *clusters* identificados foi associado o modelo de *clustering* fornecido pelo algoritmo *K-Means*.

Na Figura 6.10 são exibidos os *clusters* identificados do sub-conjunto Dados1 utilizando a ferramenta *Scatter Visualizer*. Os exemplos representados pela cor verde pertencem ao *cluster* 1 (iris-setosa), a cor vermelha pertencem ao *cluster* 2 (iris-versicolor) e a cor azul pertencem ao *cluster* (iris-virginica).

Após realizada a análise do algoritmo *K-Means* com o conjunto de dados iris considerando o atributo-classe, foi feita uma análise ignorando o mesmo. O número de exemplos pertencentes, o grau de dispersão e os atributos mais relevantes em cada *cluster* identificado são apresentados na Tabela 6.12.

Figura 6.8: *Scatter Visualizer* do Sub-conjunto Dados1 gerado pelo *ACPro* sem o Atributo-classe.

Com a ferramenta *Cluster Visualizer* foi possível observar os *clusters* gerados pelo algoritmo *K-Means*, sem considerar o atributo-classe, como pode ser visto na Figura 6.11.

Para observar como os dados estão distribuídos, foi executada a ferramenta *Scatter Visualizer* com o sub-conjunto Dados1, sem o atributo-classe. O atributo *petal_lenght* foi associado para o eixo 1, *sepal_lenght* para o eixo 2, *petal_width* para o eixo 3 e para representar a cor dos *clusters* identificados foi associado um modelo de *clusters* fornecido pelo algoritmo de *clustering K-Means*. Na Figura 6.12 são apresentados como os *clusters* estão distribuídos sem utilizar o atributo-classe. Foi notado que o *cluster* que obteve o menor grau de dispersão (destacado na Tabela 6.12) os atributos que são considerados mais relevantes são os mesmos identificados pela ferramenta *Column Importance*.

Com a ferramenta *Scatter Visualizer* é possível visualizar os *clusters* do sub-conjunto Dados1 e principalmente observar que o *cluster* 2 está linearmente separável dos demais *clusters*.

6.3.5 Análise dos Resultados

Nesta seção são apresentadas as análises dos resultados obtidos pelos quatro algoritmos citados utilizando o conjunto de dados iris.

Sub-conjunto de Dados	Nº do <i>Cluster</i>	Nº de Exemplos	Grau de Dispersão	Atributos Mais Relevantes
Dados1	1	35	0.203 +- 0.105	<i>class, p-w e p-l</i>
	2	32	0.278 +- 0.122	<i>class, p-w e p-l</i>
	3	33	0.230 +- 0.090	<i>class, p-w e p-l</i>
Dados2	1	32	0.264 +- 0.123	<i>class, p-l e p-w</i>
	2	36	0.186 +- 0.983	<i>class, p-w e p-l</i>
	3	32	0.210 +- 0.984	<i>class, p-l e p-w</i>
Dados3	1	36	0.267 +- 0.118	<i>class, p-w e p-l</i>
	2	29	0.208 +- 0.107	<i>class, p-w e p-l</i>
	3	35	0.244 +- 0.100	<i>class, p-w e p-l</i>
Completo	1	50	0.192 +- 0.099	<i>class, p-w e p-l</i>
	2	50	0.264 +- 0.117	<i>class, p-w e p-l</i>
	3	50	0.223 +- 0.093	<i>class, p-w e p-l</i>

Tabela 6.11: Informações dos *Clusters* Encontrados pelo *K-Means* com o Atributo-classe.

O algoritmo *AutoClass* conseguiu distinguir bem a estrutura dos três *clusters* utilizando o atributo-classe como um atributo qualquer apesar de não determiná-lo como o atributo mais relevante e na análise que o atributo-classe foi eliminado o algoritmo conseguiu reproduzir eficientemente apenas a classe iris-setosa.

Utilizando o modelo de *clustering* do algoritmo *K-Means* na ferramenta *Scatter Visualizer* (Figura 6.10), foi observado que o *cluster* 1 (iris-setosa) é linearmente separável dos outros *clusters*.

Na Tabela 6.13 é apresentada uma síntese geral do comportamento dos algoritmos utilizados com os três sub-conjuntos de dados iris considerando o atributo-classe, bem como algumas informações adicionais. Devido as características do algoritmo *Naive Bayes*, algumas células não foram totalmente preenchidas. Como descrito na seção 5.4.3, o tempo de execução do algoritmo *ACPro* é relativamente menor em relação ao algoritmo *AutoClass* como destacado na Tabela 6.13. Uma observação importante com relação ao tempo de processamento dos algoritmos *Naive Bayes*, *ACPro* e *K-Means* é que nas avaliações experimentais foi utilizada uma *Silicon Graphics, SGI Irix* com 64MB de RAM. E o tempo de

Figura 6.9: *ClusterViz* do Sub-conjunto Dados1 gerado pelo *K-Means* com o Atributo-classe.

processamento apresentado nas avaliações do algoritmo *AutoClass* foi utilizado um *Pentium MMX 200*, *Windows NT4.0* com 64MB de RAM.

Analisando a distribuição do atributo-classe (*class*) nos três sub-conjuntos de dados criados para esse estudo de casos foi observado que cada sub-conjunto possui as seguintes informações:

- Dados1: iris-setosa: 35; iris-versicolor: 33 e iris-virginica: 32
- Dados2: iris-setosa: 36; iris-versicolor: 32 e iris-virginica: 32
- Dados3: iris-setosa: 29; iris-versicolor: 35 e iris-virginica: 36

Com essas informações foi possível ter um maior entendimento na formação de alguns *clusters* encontrados pelos algoritmos estatísticos.

Figura 6.10: *Scatter Visualizer* do o Sub-conjunto Dados1 gerado pelo *K-Means* com o Atributo-classe.

6.4 Investigação dos Algoritmos Utilizando o Conjunto de Dados Seguros

Como mostrado na seção 6.1, o objetivo da investigação nos dados seguros é encontrar *clusters* na relação HHOLD. Vale ressaltar que nesses dados não existe um atributo-classe definido, dessa forma o algoritmo *Naive Bayes* não foi utilizado.

Nessa investigação não foi utilizado o conjunto de dados completo seguros por possuir um número de exemplos elevado e também pelo fato que não é recomendado utilizar conjuntos de dados com mais de 10.000 exemplos nos algoritmos de *clustering* disponíveis no *MinesetTM*.

6.4.1 *AutoClass*

Nas investigações realizadas com o algoritmo *AutoClass*, não foi determinado o número de *clusters* a serem identificados nos três sub-conjuntos de dados seguros. Os atributos reais

Sub-conjunto de Dados	Nº do <i>Cluster</i>	Nº de Exemplos sem Atr.-classe	Grau de Dispersão	Atributos Mais Relevantes
Dados1	1	38	0.230 +- 0.081	<i>p-l</i> , <i>p-w</i> e <i>s-w</i>
	2	35	0.203 +- 0.105	<i>p-w</i> , <i>p-l</i> e <i>s-l</i>
	3	27	0.244 +- 0.101	<i>p-l</i> , <i>s-l</i> e <i>p-w</i>
Dados2	1	24	0.223 +- 0.103	<i>p-l</i> , <i>p-w</i> e <i>s-l</i>
	2	36	0.186 +- 0.098	<i>p-w</i> , <i>p-l</i> e <i>s-l</i>
	3	40	0.219 +- 0.083	<i>p-w</i> , <i>p-l</i> e <i>s-l</i>
Dados3	1	29	0.240 +- 0.093	<i>p-l</i> , <i>s-l</i> e <i>p-w</i>
	2	29	0.208 +- 0.107	<i>p-w</i> , <i>p-l</i> e <i>s-l</i>
	3	42	0.236 +- 0.084	<i>p-l</i> , <i>p-w</i> e <i>s-l</i>
Completo	1	39	0.230 +- 0.094	<i>p-l</i> , <i>p-w</i> e <i>s-l</i>
	2	50	0.192 +- 0.099	<i>p-w</i> , <i>p-l</i> e <i>s-l</i>
	3	61	0.224 +- 0.080	<i>p-l</i> , <i>p-w</i> e <i>s-l</i>

Tabela 6.12: Informações dos *Clusters* Identificados pelo *K-Means* sem o Atributo-classe.

foram definidos com o modelo `single_normal_cm`, pois os sub-conjuntos de dados possuem somente dados reais positivos e também pelo fato que esse modelo permite a ocorrência de valores ausentes.

Na Tabela 6.14 são apresentados os números de *clusters* encontrados, os valores da verossimilhança e os atributos mais relevantes dos três sub-conjuntos de dados. Nesta tabela são apresentadas também duas das várias buscas encontradas pelo *AutoClass* com o sub-conjunto Dados1.

Observando os resultados da Tabela 6.14, pode-se verificar que os valores das verossimilhanças das duas buscas encontradas no sub-conjunto Dados1 teve uma pequena diferença. De certa forma, para os usuários é mais viável manipular o número de *clusters* identificado na segunda busca, mas para o algoritmo *AutoClass* o que mais importa é o valor da verossimilhança e nesse caso a primeira busca é a mais importante. Vale lembrar que o valor da verossimilhança está na forma logarítmica, por isso, quanto menor for este valor melhor foram as buscas por *clusters* num determinado conjunto de dados. Embora tenha alterado o número de *clusters* identificados nas buscas com o sub-conjunto Dados1, os atributos mais

Figura 6.11: *ClusterViz* do Sub-conjunto Dados1 gerado pelo *K-Means* sem o Atributo-classe.

relevantes foram considerados os mesmos.

Na Figura 6.13 é mostrado parcialmente o relatório do *cluster* 0 encontrado no sub-conjunto Dados1. Por razões de clareza nesse relatório, os exemplos são apresentados apenas com as probabilidades de pertencer a esse ou a outros *clusters* simultaneamente, ignorando assim os valores dos 43 atributos existentes. Um caso que vale ser observado nesse relatório parcial é o exemplo de número 4901, que possui uma probabilidades de pertencer ao *cluster* 0 igual a 0,791, ao *cluster* 9 igual a 0,199 e ao *cluster* 5 igual a 0,010.

6.4.2 *AutoClass Pro*

Nas avaliações realizadas com o *AutoClass Pro* utilizando os três sub-conjuntos de dados, o método especificado foi o automático, os limites de *clusters* mínimo 1 e máximo 40. Nesse caso, o algoritmo encontrou os possíveis *clusters* automaticamente no intervalo destes limites. A definição do limite máximo de *clusters* foi baseado no número de *clusters* gerados pelo algoritmo *AutoClass* durante as avaliações.

Na Tabela 6.15 são apresentados alguns resultados fornecidos pelo algoritmo *ACPro* utilizando os três sub-conjuntos de dados seguros. Vale observar que nos três sub-conjuntos de dados utilizados encontraram os mesmos três atributos mais relevantes.

Na Figura 6.14 é mostrada uma ilustração parcial dos *clusters* identificados utilizando

Figura 6.12: *Scatter Visualizer* do Sub-conjunto Dados1 gerado pelo *K-Means* sem o Atributo-classe.

a ferramenta *ClusterViz* para o Sub-conjunto Dados1.

Uma maneira de observar os *clusters* identificados no espaço n -dimensional é através do uso da ferramenta *Scatter Visualizer*. Para determinar os três atributos mais relevantes do sub-conjunto Dados1 foi utilizado a ferramenta *Column Importance* do *MinesetTM* que forneceu os seguintes resultados:

- a39 (51,5)
- a2 (74,7)
- a21 (88,58)

Os três atributos mais relevantes (a39, a2 e a21) foram mapeados para os eixos do espaço tridimensional e com isso foi possível visualizar os *clusters* identificados do sub-conjunto Dados1 na ferramenta *Scatter Visualizer*, como é mostrado a Figura 6.15. Nesses experimentos, foi observado que quando existe um número de *clusters* elevado, na maioria das vezes o resultado torna-se cansativo e não muito satisfatório para os usuários.

Analisando os 26 *clusters* identificados no sub-conjunto Dados1, decidiu-se explorar o *cluster* 6 por possuir o menor número de exemplos (57 exemplos). Na Tabela 6.16 é apresentada uma descrição detalhada dos três atributos mais relevantes desse *cluster*.

Algoritmo Utilizado	Sub-conjunto de Dados	Tempo de Processamento	Nº de Exemplos nos <i>Clusters</i>	Atributos Mais Relevantes
<i>Naive Bayes</i>	Dados1	07s		<i>p_w</i>
	Dados2	06s		
	Dados3	07s		
<i>AutoClass</i>	Dados1	14s	35, 34 e 31	<i>log p_l, log p_w e class</i>
	Dados2	15s	36, 33 e 31	<i>log p_l, log p_w e class</i>
	Dados3	13s	29, 36 e 35	<i>log p_w, log p_l e class</i>
<i>ACPro - manual</i>	Dados1	04s	01, 33 e 62	<i>class, p_l e p_w</i>
	Dados2	04s	34, 59 e 02	<i>s_w, s_l e class</i>
	Dados3	05s	28, 34 e 33	<i>s_w, s_l e p_w</i>
<i>K-Means - single</i>	Dados1	02s	35, 32 e 33	<i>class, p_w e p_l</i>
	Dados2	02s	32, 36 e 32	<i>class, p_l e p_w</i>
	Dados3	02s	36, 29 e 35	<i>class, p_w e p_l</i>

Tabela 6.13: Síntese Geral dos Algoritmos Utilizados com o Conjunto de Dados Iris.

Na coluna Valores Possíveis é mostrado um dos possíveis valores de cada atributo relevante e a quantidade total de valores possíveis entre parênteses. Na coluna Quantidade no Dados1 é definida a quantidade de exemplos que possuem o valor determinado na segunda coluna no sub-conjunto Dados1, ou seja, existem 1.819 exemplos no sub-conjunto Dados1 que o atributo a4 possui o valor 74. Na coluna Quantidade no *Cluster* é determinada a quantidade de exemplos que possui esse valor no *cluster* 6. E por fim a coluna Probabilidade no *Cluster* indica a probabilidade de exemplos que possuem esse valor possível nesse *cluster*.

Sub-conjunto de Dados	Nº de <i>Clusters</i>	Valor da Verossimilhança	Atributos Mais Relevantes
Dados1	25	-799565.690	<i>log a19, log a24 e log a20</i>
	15	-804263.512	<i>log a19, log a24 e log a20</i>
Dados2	25	-806152.008	<i>log a23, log a19 e log a24</i>
Dados3	30	-798044.041	<i>log a19, log a23 e log a24</i>

Tabela 6.14: Busca Automática de *Clusters* Utilizando o *AutoClass*.

CLASS = 0					
Case #	Prob	(Class Prob)	(Class Prob)	(Class Prob)	(Class Prob)
22	0.626	8	0.374		
41	1.000				
60	0.961	9	0.039		
93	1.000				
116	1.000				
126	1.000				
141	1.000				
157	0.994	1	0.006		
4901	0.791	9	0.199	5	0.010

Figura 6.13: Relatório do Sub-conjunto Dados1 gerado pelo *AutoClass*.

Sub-conjunto de Dados	Nº de <i>Clusters</i>	Atributos Mais Relevantes
Dados1	26	HHID, a1 e a2
Dados2	27	HHID, a1 e a2
Dados3	34	HHID, a1 e a2

Tabela 6.15: Busca Automática com os Três Sub-conjuntos Utilizando o *ACPro*.

Analisando a probabilidade do atributo a4, destacado na Tabela 6.16, pode-se observar que o mesmo não tem um potencial marcante na identificação do *cluster* 6. Uma possível explicação na ocorrência deste fato é a existência de um número elevado de *clusters* (26) no sub-conjunto Dados1, tornando assim difícil identificar as características marcantes de um determinado *cluster*.

6.4.3 *K-Means*

Mediante o uso do algoritmo *K-Means* com os três sub-conjuntos de dados, o método especificado foi o *iterative*, o número de *clusters* mínimo 1 e máximo 40 e o ponto de escolha igual a 0,5.

Na Tabela 6.17 são apresentados os números de *clusters* encontrados e os três atributos mais relevantes gerado pelo algoritmo *K-Means* utilizando os três sub-conjuntos de dados.

Figura 6.14: *ClusterViz* do Sub-conjunto Dados1 gerado pelo *ACPro*.

Nº do Atributo	Valores Possíveis	Quantidade no Dados1	Quantidade no <i>Cluster</i>	Probabilidade no <i>Cluster</i>
a4	74(27)	1.819	16	0,87%
a16	71838(283)	106	23	21,69%
a2	7(60)	1.399	14	1,00%

Tabela 6.16: Explorando o *Cluster 6* do Sub-conjunto Dados1 gerado pelo *ACPro*.

Um fato a ser observado foi a semelhança encontrada pelo algoritmo *K-Means* tanto no número de *clusters* quanto na determinação dos atributos mais relevantes para os três sub-conjuntos de dados seguros.

Uma visualização parcial dos *clusters* encontrados no sub-conjunto Dados1 na ferramenta *ClusterViz* pode ser vista na Figura 6.16.

Como já foi determinado os três atributos mais relevantes do sub-conjunto Dados1 (vistos na seção 6.4.2), estes foram utilizados para visualizar os *clusters* encontrados pelo algoritmo *K-Means* na ferramenta *Scatter Visualizer*, como é mostrado a Figura 6.17.

Com a finalidade de explorar melhor o sub-conjunto Dados1, um resumo contendo o número de exemplos, o grau de dispersão e os atributos mais relevantes para cada *cluster* encontrado, como pode ser visto na Tabela 6.18.

Figura 6.15: *Scatter Visualizer* gerado pelo *ACPro* com o Sub-conjunto Dados1.

Sub-conjunto de Dados	Nº de <i>Clusters</i>	Atributos Mais Relevantes
Dados1	6	a39, a36 e a38
Dados2	6	a39, a36 e a38
Dados3	6	a39, a36 e a38

Tabela 6.17: Busca Iterativa com os Três Sub-conjuntos Utilizando o *K-Means*.

Observando o número de exemplos em cada *cluster* encontrado foi selecionado o *cluster* 3 para realizar uma análise mais detalhada a fim de verificar algum conhecimento desse *cluster*. Na Tabela 6.19 é apresentada uma descrição dos três atributos considerados relevantes no *cluster* 3. Na coluna Valores Possíveis é mostrado um dos valores possíveis de cada atributo relevante e a quantidade total de valores possíveis entre parênteses. Na coluna Quantidade no Dados1 é definida a quantidade de exemplos que possuem esse valor no sub-conjunto Dados1. Na coluna Quantidade no *Cluster* é determinada a quantidade de exemplos que possui esse mesmo valor possível no *cluster* 3. E por fim, na coluna Probabilidade no *Cluster* indica a probabilidade de exemplos que possuem esse valor no *cluster* 3.

Um fato que merece ser destacado neste experimento é a influência do atributo a5 com o valor 297 na identificação do *cluster* 3 do sub-conjunto Dados1. Observando a ordem

Figura 6.16: *ClusterViz* com o Sub-conjunto Dados1 gerado pelo *K-Means*.

geral dos atributos mais relevantes do sub-conjunto Dados1, o atributo a5 é considerado como o oitavo mais relevante entre os 43 existentes.

6.4.4 Análise dos Resultados

Nesta seção são apresentadas as análises de resultados obtidos pelos algoritmos de *clustering* utilizando o conjunto de dados seguros. Os algoritmos descritos anteriormente foram utilizados com a finalidade de extrair *clusters* nos três sub-conjuntos de dados.

Nº do <i>Cluster</i>	Nº de Exemplos	Grau de Dispersão	Atributos Mais Relevantes
1	1.052	2.51 +- 0.25	a5, a2 e a6
2	2.369	2.69 +- 0.20	a36, a40 e a30
3	747	2.65 +- 0.30	a5, a2 e a1
4	1.833	2.78 +- 0.22	a39, a21 e a18
5	1.165	2.84 +- 0.22	a23, a21 e a18
6	1.457	2.84 +- 0.25	a39, a36 e a38

Tabela 6.18: Explorando os *Clusters* Identificados no Sub-conjunto Dados1 Seguros.

Figura 6.17: *Scatter Visualizer* do Sub-conjunto Dados1 gerado pelo *K-Means*.

Nº do Atributo	Valores Possíveis	Quantidade no Dados1	Quantidade no <i>Cluster</i>	Probabilidade no <i>Cluster</i>
a5	297(75)	179	165	92,17%
a2	18(59)	544	305	56,06%
a1	1(28)	1.686	306	18,14%

Tabela 6.19: Explorando o *Cluster* 3 do Sub-conjunto Dados1.

A quantidade de *clusters* encontrados automaticamente pelos algoritmos *AutoClass* e *ACPro* foi grande (vistos nas Tabelas 6.15 e 6.17) comparado com o algoritmo *K-Means*. Uma explicação possível para este fato é que nos algoritmos baseados em probabilidades os exemplos podem pertencer a várias *clusters* simultaneamente.

Uma síntese geral dos algoritmos usados com os três sub-conjuntos de dados seguros pode ser visualizada na Tabela 6.20. Um ponto a ser destacado é a variedade do tempo gasto nas execuções dos algoritmos, merecendo ser destacado o tempo do sub-conjunto Dados2 utilizando o algoritmo *AutoClass* e o sub-conjunto Dados3 utilizando o algoritmo *K-Means*. Com relação ao número de *clusters* identificados e aos atributos mais relevantes nesses sub-conjuntos de dados pode-se observar que houve uma diferença grande na determinação dos mesmos. Um fato interessante foi a semelhança encontrada pelo algoritmo *K-Means* tanto

no número de *clusters* quanto na determinação dos atributos mais relevantes para os três sub-conjuntos de dados seguros.

Algoritmo Utilizado	Sub-conjunto de Dados	Tempo de Processamento	Nº de <i>Clusters</i>	Atributos Mais Relevantes
<i>AutoClass</i>	Dados1	15m16s	25	<i>log a19, log a24 e log a20</i>
	Dados2	12m57s	25	<i>log a23, log a19 e log a24</i>
	Dados3	25m9s	30	<i>log a19, log a23 e log a24</i>
<i>ACPro - automático</i>	Dados1	33m6s	26	HHID, a1 e a2
	Dados2	21m18m	27	HHID, a1 e a2
	Dados3	34m49s	34	HHID, a1 e a2
<i>K-Means - iterative</i>	Dados1	19h12m36s	6	a39, a36 e a38
	Dados2	18h36s	6	a39, a36 e a38
	Dados3	21h45m	6	a39, a36 e a38

Tabela 6.20: Síntese Geral dos Algoritmos Utilizados com o Conjunto de Dados Seguros.

6.5 Considerações Finais

Neste capítulo foi realizado um estudo experimental de alguns algoritmos estatísticos utilizando dois conjuntos de dados. Estes experimentos tiveram como objetivo principal investigar como esses algoritmos se comportavam utilizando conjuntos de dados com características muito diferentes.

Embora todos esses algoritmos avaliados pertençam ao paradigma estatístico, eles não foram comparativamente avaliados, de maneira sistemática, em situações empíricas. O estudo e a avaliação de cada um deles evidenciaram situações que merecem algumas considerações, tais como: o número de *clusters* identificados, a determinação de atributos relevantes e o tempo de processamento.

A ausência de análises de dados definidas para determinar qual algoritmo de aprendizado terá um bom comportamento sobre um conjunto de dados faz com que estudos experimentais sejam relevantes para a área de Aprendizado de Máquina.

O algoritmo *Naive Bayes* possui limitações, que devem ser respeitadas antes de aplicá-lo nos experimentos. É importante ressaltar que um cuidado especial deve ser tomado ao

comparar os números de *clusters* encontrados por vários algoritmos no *MinesetTM*. Para realizar comparações devem ser considerados vários fatores, tais como a escolha do algoritmo de *clustering* e principalmente o método a ser utilizado. Uma vez que o algoritmo *K-Means* trabalha com medidas de distâncias e o algoritmo *ACPro* trabalha com cálculos de probabilidades.

E, por último, cabe ressaltar que a importância do significado dos atributos nos conjuntos de dados é um fator fundamental na análise e compreensão dos *clusters* encontrados pelos algoritmos de *clustering*.

Capítulo 7

Conclusões

Diante da grande quantidade de sistemas de aprendizado desenvolvidos, é relevante utilizar uma classificação dos mesmos para um melhor entendimento de suas aplicações. Duas entre as possíveis maneiras de classificar os algoritmos de Aprendizado de Máquina são de acordo com um determinado paradigma de Aprendizado de Máquina (estatístico, simbólico, conexionista, etc.) ou também com relação a forma que eles aprendem: aprendizado supervisionado (necessita da definição dos atributos-classes) e aprendizado não supervisionado (não necessita da definição dos atributos-classes).

O paradigma estatístico possui diversas técnicas para análise e compreensão de dados na área de Aprendizado de Máquina. Entre essas técnicas disponíveis destacam-se a teoria da probabilidade (regra de Bayes), medidas de similaridades (distância Euclidiana e *Manhattan*) e redes Bayesianas.

Em geral, os algoritmos de aprendizado realizam duas tarefas básicas a partir dos conjuntos de dados: predição e descrição. A tarefa de predição, considerada por alguns autores a mais utilizada, é caracterizada pelo uso dos atributos-classes nos conjuntos de dados para prever um valor em novos exemplos. Já a tarefa de descrição é caracterizada pela maneira de descrever os exemplos ou padrões identificados a partir dos conjuntos de dados. A principal tarefa de descrição é o *clustering*, que visa identificar um conjunto finito de *clusters* a partir dos dados que podem ser mutuamente exclusivos ou exaustivos, fornecendo assim uma representação para a descrição dos dados.

Para a realização de uma das tarefas de aprendizado é necessário conhecer bem as características dos conjuntos de dados a serem utilizados (se o atributo-classe está especificado

ou não) e também o paradigma de aprendizado que o algoritmo irá trabalhar.

A contribuição principal deste trabalho foi a investigação de alguns algoritmos de Aprendizado de Máquina pertencentes ao paradigma estatístico visando a aquisição de conhecimento gerados por eles. A idéia central dessa investigação foi abordar vários algoritmos estatísticos que trabalham com modos de aprendizado diferentes. No aprendizado supervisionado, foi abordado o algoritmo *Naive Bayes* que trabalha com cálculos de probabilidades (regra de Bayes). No aprendizado não supervisionado, mais especificamente *clustering*, foram abordados os algoritmos *AutoClass*, *AutoClass Pro* e *K-Means*. Entre os algoritmos de *clustering* abordados, *AutoClass* e *AutoClass Pro* trabalham com cálculos de probabilidades (regra de Bayes) e *K-Means* trabalha com medidas de similaridades (distância Euclidiana e *Manhattan*).

Um dos principais problemas encontrados nesse trabalho foi a dificuldade em compreender a linguagem de descrição gerada pelos algoritmos estatísticos. Assim, muitos dos algoritmos estatísticos buscam ferramentas de apoio para compreender o conhecimento gerado, tais como, ferramentas de visualização. Um problema encontrado quando envolve algoritmos de *clustering* é a análise do número elevado de *clusters* identificados numa determinada aplicação e um fato que facilita esse problema é o conhecimento prévio dos conjuntos de dados a serem utilizados.

Nesse presente trabalho foram realizadas duas avaliações experimentais, nas quais foi observado como os algoritmos estatísticos se comportavam utilizando dois conjuntos de dados com características distintas. Na primeira avaliação foi utilizado o conjunto de dados de várias espécies de plantas iris, no qual os atributos-classes são definidos (setosa, versicolor e virginica) e também deve-se ressaltar que esse conjunto não possui valores ausentes. Nessa avaliação foram realizadas duas análises com os algoritmos, uma considerando o atributo-classe como um atributo qualquer no conjunto de dados e outra eliminando o mesmo da descrição dos atributos (exceto para o algoritmo *Naive Bayes*). Na segunda avaliação foi utilizado o conjunto de dados de domicílios dos clientes de uma seguradora específica, no qual os atributos-classes são indefinidos e também deve-se ressaltar que esse conjunto possui valores ausentes para alguns atributos.

Através dessas avaliações experimentais realizadas foi possível verificar que o algoritmo supervisionado *Naive Bayes* foi eficaz nas avaliações realizadas. E os algoritmos de *clustering* *AutoClass*, *AutoClass Pro* e *K-Means* tiveram um melhor comportamento quan-

do o número de *clusters* encontrado foi inferior a 10 (dez), também foi observado que eles têm uma alta complexidade computacional. Além disso, foi notado que as ferramentas de visualização são de grande relevância na análise e compreensão do conhecimento adquirido.

A seguir são apresentadas algumas sugestões para trabalhos futuros:

- Investigar mais profundamente a área de estatística com a finalidade de discutir métodos que possam contribuir em análises experimentais dos algoritmos de Aprendizado de Máquina.
- Investigar outros algoritmos de *clustering* que trabalham com cálculos de probabilidades e medidas de distâncias.
- Propor uma linguagem de descrição (uma forma mais entendível) do conhecimento adquirido pelos algoritmos de Aprendizado de Máquina que realizam a descrição como tarefa de aprendizado.
- Investigar o comportamento de alguns algoritmos de *clustering* utilizando os vários métodos de particionamento dos dados, tais como: *holdout*, *n-fold cross-validation* e *bootstrap*.

Referências

- Achcar, J. A. & Rodrigues, J. (1998). Introdução à estatística para ciências e tecnologia. Technical Report 192, ICMC-USP.
- Aitkin, M., Anderson, D., & Hinde, J. (1981). Statistical modelling of data on teaching styles. *Journal of the Royal Statistical Society(A)*, 144:419–461.
- Andersen, S. K., Olesen, K. J., Jensen, F. V., & Jensen, F. (1989). F. hugin - a shell for building bayesian belief in expert systems. *Proc. 11th International Joint Conference on AI, Detroit*, pages 1080–1085.
- Baxter, R. & Oliver, J. (1997). Data mining consulting. <http://www.ultimode.com>.
- Castineira, M. I. (1996). Explicação em sistemas que utilizam diagramas de influências como formalismo de representação de conhecimento. Tese de Doutorado, IFSC-USP.
- Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1990). Autoclass: A bayesian classification system. *Readings in Machine Learning, Morgan Kaufmann*, pages 431–441.
- Cheeseman, P. & Stutz, J. (1996). Bayesian classification (autoclass): Theory and results. *Advances in Knowledge Discovery & Data Mining*, pages 153–180.
- Chinrungrueng, C. & Séquin, C. H. (1995). Optimal adaptive k-means algorithm with dynamic adjustment of learning rate. *IEEE Transactions on Neural Networks*, 6(1):157–169.
- Clark, P. & Niblett, T. (1989). The cn2 induction algorithm. *Machine Learning*, 3(4):261–283.
- Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. Cambridge, Mass: Mit Press.

- Cover, T. M. & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.
- Decker, K. M. & Focardi, S. (1995). Technology overview: A report on data mining. Technical report, CSCS-ETH, Swiss Scientific Computing Center.
- Dietterich, T. G. (1996). Machine learning. *Computing Surveys*, 28(4).
- Dietterich, T. G. (1997). Machine learning research: Four current directions. Technical report, Department of Computer Science, University of State Oregon.
- Domingos, P. & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple bayesian classifier. *Proceedings of the 13th International Conference of Machine Learning*, pages 105–112.
- Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Machine Learning: Proceedings of the Twelfth International Conference*, pages 194–202.
- Drakos, N. (1998). *ACPro Segmentation Tool — User's Guide*. Computer Based Learning Unit, University of Leeds.
- Duda, R. O. & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley-Interscience.
- Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings The Second International Conference on Knowledge Discovery & Data Mining (KDD-96)*, pages 226–231.
- Fayyad, U. M., Piatesky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). From data mining to knowledge discovery in databases. *American Association for Artificial Intelligence*, pages 37–54.
- Félix, L. C. M. (1998). Data mining no processo de extração de conhecimento de bases de dados. Dissertação de Mestrado, ICMC-USP.
- Félix, L. C. M., Rezende, S. O., Doi, C. Y., de Paula, M. F., & Romanato, M. J. (1998). *MLC++* biblioteca de aprendizado de máquina em *C++*. Technical Report 72, ICMC-USP.
- Filho, C. P. (1995). *Introdução à Simulação de Sistemas*. Editora UNICAMP.

- Francisco, W. D. (1993). *Estatística Básica*. Editora Básica.
- Goldberg, H. (1998). Restructuring databases by consolidation and link formation. *Symposium on Artificial and Link Analysis*, pages 47–52.
- Hanson, R., Stutz, J., & Cheeseman, P. (1991). Bayesian classification theory. Technical report, Technical Report FIA-90-12-7-01, NASA Ames Research Center, Artificial Intelligence Branch.
- Hartigan, J. A. & Wong, M. A. (1978). A k-means clustering algorithm. *Applied Statistics*, pages 100–108.
- Hastie, T. & Tibshirani, R. (1995). Discriminant adaptive nearest neighbor classification. *Proceedings The First International Conference on Knowledge Discovery Data Mining*, pages 142–149.
- Haykin, S. (1994). *Neural Networks A Comprehensive Foundation*. Macmillan College Publishing Company, Inc.
- Heckerman, D. (1997). Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1:79–119.
- Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction: Processes of Inference, Learning and Discovery*. MIT Press, Cambridge.
- Ismail, M. A. & Kamel, M. S. (1989). Multidimensional data clustering utilizing hybrid search strategies. *Pattern Recognition*, 22(1):75–89.
- Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. *Proceedings The Second International Conference on Knowledge Discovery & Data Mining*, pages 202–207.
- Kohavi, R. & Kunz, C. (1997). Option decision trees with majority votes. *Proceedings of the Fourteenth International Conference of Machine Learning*.
- Larson, R. C. & Odoni, A. R. (1943). *Urban Operations Research*. Prentice-Hall, Inc., Englewood Cliffs.
- Ling, C. X. & Wang, H. (1997). Computing optimal attribute weight setting for nearest neighbor algorithms. *Artificial Intelligence Review*, pages 255–272.

- McAuliffe, T. L. & Affi, A. A. (1984). Comparasion of a nearest neighbor and other approaches to the detection of space-time clustering. *Computational Statistics and Data Analysis*, 2:125–142.
- Meneses, C. J. & Grinstein, G. G. (1998). Categorization and evaluation of data mining techniques. *International Conference on Data Mining, Rio de Janeiro*, pages 53–80.
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Series in Computer Science.
- Mizoguchi, R. & Shimura, M. (1980). A nonparametric algorithm for detecting clusters using hierarchial structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(4):292–300.
- Morgan, J. & Messenger, R. (1973). *THAID*: A sequential search program for the analysis of nominal scale dependent variables. Technical report, Institute for Social Research, University of Michigan.
- Nilsson, N. J. (1996). *Introduction to Machine Learning*. Stanford University.
- Oliveira, P. R. (1997). Redes neurais artificiais para extração de características. Dissertação de Mestrado, ICMC-USP.
- Oliveira, R. B. T. & Rezende, S. O. (1998). Ferramentas de visualização de dados do mineset. Technical Report 71, ICMC-USP.
- Padilha, T. P. P. & Rezende, S. O. (1999). Algoritmos de aprendizado de máquina. Technical report, ICMC-USP (em andamento).
- Pazzani, M. J. (1995). Searching for dependencies in bayesian classifiers. Technical report, Departament of Information and Computer Science, University of California.
- Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*. (27):221–234.
- Quinlan, J. R. (1988). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers. Inc.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers. Inc., Los Altos, California, USA.
- Quinlan, J. R. & Carmeron-Jones, R. M. (1993). Foil: A midterm report. *Proceedings of the 6th European Conference on Machine Learning ECML-93*, pages 3–20.

- Ramoni, M. & Sebastiani, P. (1997). Discovering bayesian networks in incomplete database. Technical Report 46, Knowledge Media Institute - The Open University.
- Russell, S. J. & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.
- Weiss, S. M. & Kulikowski, C. A. (1991). *Computer Systems That Learn Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems*. Morgan Kaufmann Publishers, CA.
- Wettschereck, D. (1994). A study of distance-based machine learning algorithms. PhD thesis, Oregon State University.
- Wirth, R. & Reinartz, T. P. (1996). Detecting early indicator cars in an automotive database: A multi-strategy approach. *Proceedings The Second International Conference on Knowledge Discovery*, 3(4):76–81.
- Wong, M. A. & Lane, T. (1983). A kth nearest neighbour clustering procedure. *Journal of the Royal Statistical Society(A)*, 45(3):362–368.