Avaliação do Conhecimento Adquirido por Algoritmos de Aprendizado de Máquina Utilizando Exemplos[1]

Paulo Sergio Horst

Orientação:

Profa. Dra. Maria Carolina Monard

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
Agosto de 1999

[1]Trabalho realizado com auxílio do PICDT-CAPES.

Este documento foi preparado com o formatador de textos LaTeX. O sistema de citações de referências bibliográficas utiliza o padrão *Apalike* do sistema BibTeX.

Dedicatória

Aos meus pais, José Arnaldo e Alzira,
meu irmão Matheus e minhas irmãs Josiane e Clarice.

Agradecimentos

À Deus, que ilumina os meus passos e está sempre presente.

À Profa. Maria Carolina Monard, pela excelente orientação e por ter me dado a oportunidade de aprender muito. Pela sua amizade, dedicação, paciência, compreensão e pelos conselhos neste período.

À Profa. Solange pelo apoio, conselhos e atenção. Ao Prof. André, pelo incentivo e auxílio. Ao prof. Ricardo e prof. Alex pelas sugestões iniciais. Ao Augusto e à Huei pela colaboração e sugestões na fase final deste trabalho.

Em especial à Aletéia uma grande amiga que sempre me incentivou e esteve presente nas horas boas e também nas difíceis.

À Thereza e Alex pelo apoio, amizade e pelos bons momentos que compartilhamos.

Aos amigos (Chandler, Gentil, Jaque, Selma, Maju, Sílvio, Fábio, Luis Carlos, Gustavo, Maristela, Hélio, Edmar, Paulo, Flávia, Jorge, Andreza, Rudinei e muitos outros) com os quais convivi estes anos.

Ao Renato e Regiane. À uma turma muito legal (Cris, Marco, Omar, Elaine, Andrea, Jorge) e à todos os amigos do LABIC e dos outros laboratórios pelo companherismo e amizade.

À toda minha família, que sempre me incentivou, me passou confiança, amor e compreensão em todos os momentos.

Às funcionárias da Pós-Graduação (Beth, Laura e Marília), da biblioteca, ao Paulinho e a todos os funcionários do ICMC por me atenderem sempre bem.

A CAPES e UEPG pelo apoio financeiro. Aos amigos da UEPG, prof. Merhy, Amaury, prof. Bráulio que me incentivou e me mostrou a direção a seguir. Aos amigos da DIMAPA, e o pessoal da PROPESP e DRH pela atenção.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

O avanço de algumas áreas como computação e comunicação de dados, bem como a busca incessante pelo domínio das informações, contribui para aumentar cada vez mais as pesquisas relacionadas com aquisição de conhecimento, tema central da área de Inteligência Artificial.

A aquisição implícita de conhecimento é realizada utilizando-se algoritmos de Aprendizado de Máquina. No caso de algoritmos simbólicos supervisionados, o conhecimento adquirido é representado em estruturas lógicas, tais como regras do tipo **se então**, que são entendíveis pelo ser humano.

Quando o número de regras é elevado, ou as regras consideram muitas condições no seu corpo, torna-se difícil, ao ser humano, a análise desse conhecimento. Uma solução para esta questão é o desenvolvimento de boas medidas de avaliação de regras. Independentemente da quantidade de regras, essas medidas ajudam a selecionar aquelas que são mais úteis e interessantes, pois parte do conhecimento adquirido dos exemplos pode ser muito óbvio ou irrelevante.

Neste trabalho são discutidas algumas medidas propostas na literatura, com a finalidade de auxiliar o usuário no entendimento e utilização proveitosa do conhecimento adquirido. Com base nos estudos realizados foi projetado e implementado um sistema computacional, denominado $\mathcal{RQ}_{system}$, para auxiliar na avaliação dessas regras de conhecimento.

O $\mathcal{RQ}_{system}$ foi desenvolvido na linguagem de programação lógica Prolog e consiste de dois módulos principais. O primeiro é responsável pelo pré-processamento dos dados de entrada. O segundo módulo é responsável por fornecer diversar informações pré-definidas no sistema ou construídas e formuladas pelo usuário.

O $\mathcal{RQ}_{system}$ está descrito neste trabalho utilizando um pequeno conjunto de dados do mundo real e as regras geradas pelos algoritmos de Aprendizado de Máquina $\mathcal{CN}2$ e $\mathcal{C}4.5$. Esse sistema tem características interessantes que lhe conferem uma boa utilidade tanto na avaliação de regras quanto no estudo de outras questões relacionadas com as regras. Extensões futuras do sistema poderão ser particularmente úteis em Data Mining.

Abstract

The field of Machine Learning (ML) is concerned with the development of computational methods to implement various forms of learning, in particular methods capable of inducing knowledge from examples, *i.e.* determining a concept description from a set of provided concept examples.

Learning algorithms can generally be classified into one of two major categories: *black-box* methods and *knowledge-oriented* methods. The description produced by the black-box approach cannot be easily interpreted by the user and does not provide explanation of the recognition process. On the other hand, knowledge-oriented methods aim at creating symbolic knowledge structures that satisfy the principle of comprehensibility, providing explanation of the recognition process. In this work we consider knowledge-oriented ML algorithms that express the discovered concept in the form of if-then rules.

An important problem is related to the reliability, quality and interestingness of the rules generated by these algorithms. Still, when the quantity of rule generated is large, the selection of *good* rules can became a serious problem for the human user.

In this work we present and discuss several measures that can provide useful support in interpreting and ranking the rules generated by Machine Learning algorithms. These measures were implemented in a computational system called $\mathcal{RQ}_{system}$. The system uses as input a common file format for data sets and features description which is independent of the ML algorithm used to generate the if-then rules. The file format for the rules generated is algorithm dependent.

The $\mathcal{RQ}_{system}$ has been implemented in Prolog and it is query-centered, permitting the user to specify any constraints on the desired result of a query. The user can either specify the constraints in terms of procedures already implemented in the system or can define his/her new procedures to be considered as new constraints.

The $\mathcal{RQ}_{system}$ is described in this work using a small real world data set and the rules generated by $\mathcal{CN}2$ and $\mathcal{C}4.5$ Machine Learning algorithms. Future extensions to the system that we consider will be particularly useful in Data Mining are also discussed.

Capítulo 1

Introdução

Vive-se hoje na chamada *era da informação*, na qual a busca pelo domínio das informações mais importantes é incessante. Nesse sentindo, as áreas de comunicação de dados e computação têm sido as maiores contribuintes, principalmente por fornecerem alta tecnologia em suporte de armazenamento, processamento e acesso rápido e remoto às mais diversas fontes de informação.

Considerando que o conhecimento pode ser visto como uma abstração ou um nível de informação acima dos dados, existe a necessidade de áreas de pesquisa dentro da computação que tratem desse assunto, tais como a área de Inteligência Artificial (IA).

Em IA, o processo de aquisição de conhecimento pode ser realizado de forma explícita, ou seja, a aquisição é feita por intermédio de engenheiros de conhecimento utilizando técnicas convencionais (estudo de casos, entrevistas, etc.). Outra forma de aquisição de conhecimento é a implícita, isto é, de forma automatizada utilizando conjuntos de dados disponíveis no domínio considerado.

A subárea de IA, chamada de Aprendizado de Máquina, tem como objetivo principal realizar aquisição automática de conhecimento.

Na área de Aprendizado de Máquina são especificamente pesquisados, desenvolvidos e aplicados sistemas de aprendizado automatizados, os quais utilizam diferentes paradigmas, estratégias e modos de aprendizado. Nesse contexto, este trabalho se situa no estudo de sistemas de Aprendizado de Máquina por exemplos baseados no paradigma simbólico e modo de aprendizado supervisionado. Mais especificamente, neste trabalho tratamos da avaliação do conhecimento adquirido por esses sistemas (Horst, 1999).

A área de Aprendizado de Máquina tem sido bastante pesquisada ultimamente, no entanto, ainda existem vários problemas a serem resolvidos pois, em função da crescente aplicação de sistemas de aprendizado indutivos em problemas do mundo real, surge a necessidade de se ter maior confiabilidade no conhecimento adquirido.

Motivados por vários fatores, como a questão da confiabilidade, vários pesquisadores têm estudado o uso de técnicas estatísticas para solucionar, pelo menos em parte, os problemas relacionados com a avaliação da qualidade do conhecimento adquirido. Um outro importante ponto mais recentemente pesquisado é a avaliação da interessabilidade desse conhecimento a fim de procurar por conhecimento realmente interessante para o usuário.

Dessa forma, a investigação de várias técnicas, principalmente as técnicas estatísticas, para avaliar não somente a precisão de classificadores mas também o conhecimento adquirido por eles são de grande relevância. Essa avaliação deve ter também como objetivo a questão da qualidade e interessabilidade desse conhecimento, particularmente das regras geradas por sistemas de Aprendizado de Máquina baseados no paradigma simbólico.

Ambos os aspectos, qualidade e interessabilidade, são importantes para que o usuário (especialista ou analista do domínio) possa se focalizar nas regras de maior qualidade e interesse, em vez de examinar uma a uma todas as regras descobertas pelo sistema de aprendizado.

Isto deve-se ao fato de que vários dos métodos usualmente utilizados para adquirir conhecimento a partir de conjuntos de exemplos descobrem um grande número de regras. O número elevado de regras é um dos fatores que mais dificulta a análise das regras por parte do usuário. Outro fator está relacionado com a complexidade da regra, a qual refere-se, por exemplo, ao número de condições ou tipo de representação.

Independentemente da quantidade de regras, essas novas medidas de avaliação ajudam a selecionar aquelas regras que são mais úteis e interessantes, pois parte do conhecimento adquirido dos exemplos pode ser muito óbvio ou irrelevante.

Este trabalho aborda algumas medidas propostas na literatura, baseadas em estatística e teoria da informação, como uma maneira razoável de tentar medir a qualidade e interessabilidade do conhecimento adquirido, quando ele está expresso como regras. No entanto, analisar as regras manualmente pode ser impraticável devido, principalmente, a quantidade de dados associados a elas e também aos cálculos necessários para retornar tais informações. Assim, nós propomos um sistema computacional, denominado $\mathcal{RQ}_{system}$, com o objetivo de auxiliar na avaliação de regras. Esse sistema oferece facilidades para obter diversas informações relacionadas com as regras.

Este trabalho está organizado da seguinte forma.

No Capítulo 2 são apresentadas algumas características gerais dos sistemas de Aprendizado de Máquina, tais como paradigmas, estratégias, modos e tipos de aprendizado. Também é apresentado uma taxonomia das diversas questões estatísticas pesquisadas em Aprendizado de Máquina.

No Capítulo 3 é discutida a avaliação da precisão de hipóteses através de técnicas estatísticas. Os principais pontos abordados são a estimativa da precisão de hipóteses, comparação de hipóteses, teste de hipóteses e comparação de algoritmos de aprendizado.

No Capítulo 4 é mostrado que várias técnicas, com base na estatística e na teoria da informação, podem ser usadas como medidas de qualidade e interessabilidade para o conhecimento adquirido por sistemas de aprendizado simbólicos, utilizando exemplos, quando esse conhecimento está expresso como regras.

No Capítulo 5 é descrito em alto nível o $\mathcal{RQ}_{system}$ por nós proposto para auxiliar na avaliação do conhecimento adquirido por sistemas de Aprendizado de Máquina, mais especificamente, na avaliação de regras.

No Capítulo 6 são apresentadas as conclusões deste trabalho bem como trabalhos futuros a serem realizados.

Capítulo 2

Aprendizado de Máquina

2.1 Considerações Iniciais

A habilidade de aprender é um dos atributos mais importantes do comportamento inteligente. Assim, o estudo e a consequente modelagem computacional dos processos de aprendizado em suas múltiplas manifestações tem se constituído basicamente como o objetivo maior das inúmeras pesquisas em Aprendizado de Máquina (AM), dentro da área de Inteligência Artificial.

As pesquisas em Aprendizado de Máquina que inicialmente eram limitadas a estudos teóricos e posteriormente estudos experimentais recebem atualmente maior reconhecimento em virtude de sua aplicação prática. A aplicação de métodos de Aprendizado de Máquina na busca de soluções para problemas de domínios do mundo real tem sido marcante para a manutenção e maturidade dessas pesquisas. A medida que cresce o número de aplicações bem sucedidas de AM aumenta a aceitação de seu uso, bem como as exigências por mais qualidade, por mais abrangência de domínios de aplicação e, certamente, por mais confiabilidade nos resultados obtidos.

O Aprendizado de Máquina visto de uma maneira prática pode ser entendido como um conjunto de métodos, técnicas e ferramentas próprias para a aquisição automatizada de conhecimento a partir de conjuntos de dados (Rocha et al., 1997). Os métodos podem ser vistos como sistemas de aprendizado.

2.2 Sistemas de Aprendizado

Os sistemas de Aprendizado de Máquina ou simplesmente sistema de aprendizado podem ser classificados utilizando várias dimensões. Três dimensões particulares são (Michalski et al., 1986): *estratégias de aprendizado*, *representação do conhecimento adquirido* e *domínio da aplicação*.

Cada ponto no espaço definido pelas dimensões acima corresponde a uma estratégia particular de aprendizado, empregando uma determinada representação de conhecimento, e aplicada a um domínio específico. Considerando que existem sistemas de aprendizado que empregram múltiplas representações e processos, e também que muitos têm sido aplicados a vários domínios, esses sistemas podem ser caracterizados por vários pontos no espaço.

Como ilustrado na Figura 2.1, o objetivo fundamental de um sistema de aprendizado está em extrair conhecimento, por exemplo na forma de regras de decisão, de um conjunto de dados tal que essas regras possam ser aplicadas a novos dados. Assim, uma das tarefas principais desses sistemas é a chamada *classificação ou predição*.

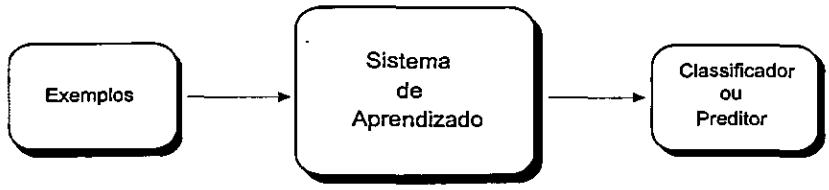

Figura 2.1: Sistema de Aprendizado (Batista, 1997)

Sob a ótica do problema de classificação, um sistema pode ser visto como um sistema de alto nível que ajuda a construir um sistema de tomada de decisão chamado de *classificador*. A Figura 2.2, ilustra a estrutura simples de um sistema de classificação que aceita um padrão de dados como entrada e produz uma decisão como saída.

Figura 2.2: Sistema de Classificação (Batista, 1997)

Neste trabalho os estudos estão concentrados particularmente nos sistemas de aprendizado supervisionado por exemplos. Esses sistemas podem ser vistos de uma maneira prática como aqueles que dispõem de um conjunto de exemplos para a aquisição de conhecimento. Nesse conjunto, cada exemplo corresponde a atributos que o descrevem e está relacionado a uma determinada classe. É comum que esses conjuntos sejam divididos em subconjuntos chamados de conjunto de exemplos de treinamento e conjunto de exemplos de teste, que são utilizados pelos algoritmos de aprendizado na fase de treinamento e na fase de teste, respectivamente.

Os sistemas de aprendizado por exemplos podem ser definidos de uma maneira mais formal como segue:

> Cada exemplo x_i é uma n-tupla extraída de algum conjunto X de acordo com alguma distribuição D de probabilidade desconhecida e fixa. Uma função f é aplicada em cada exemplo para produzir um rótulo $y_i = f(X_i)$. Estes rótulos podem ser valores contínuos — neste caso, o problema é referenciado como um problema de regressão — ou categóricos, isto é, símbolos discretos denominados de classe — caso em que é referenciado como um problema de classificação.
>
> O objetivo de algoritmos de Aprendizado de Máquina é construir uma aproximação h para a função desconhecida f de forma que, com alta probabilidade, um novo exemplo $x \in X$ extraído de acordo com D seja rotulado corretamente: $h(x) = f(x)$ (Dietterich, 1997).

Algoritmos de Aprendizado de Máquina trabalham buscando algum espaço de hipóteses H, para a hipótese h que seja "melhor" em algum sentido. Duas questões fundamentais de pesquisas em aprendizado de máquina sob este aspecto são:

> *Quais são os espaços de hipótese bons para busca?*
> *Quais definições de "melhor" devem ser usadas?*

Um exemplo de resposta para a primeira questão é o bem conhecido espaço de hipóteses de árvores de decisão e para a segunda questão, a definição de "melhor" é a hipótese que minimiza a chamada estimativa de erro pessimista.

Em termos práticos, a efetividade de um algoritmo de Aprendizado de Máquina depende em muito do espaço de hipóteses ser suficientemente pequeno e conter boas aproximações para a

função f. Um espaço de hipóteses pequeno demanda naturalmente menos tempo de busca, mas pode não conter uma boa aproximação de h para a função f.

Os sistemas de aprendizado possuem características particulares e comuns que possibilitam uma certa classificação quanto ao modo, paradigmas e estratégias de aprendizado, e também quanto a linguagem de descrição e forma de integração de novos exemplos.

2.2.1 Paradigmas de Aprendizado

Vários paradigmas já foram propostos para sistemas de Aprendizado de Máquina utilizando exemplos, os mais usualmente relatados são os paradigmas: *simbólico*, *estatístico*, *instance-based*, *conexionista* e *genético*.

Os sistemas de aprendizado que utilizam o *paradigma simbólico* buscam aprender analisando exemplos e contra-exemplos para construir representações simbólicas de um conceito. Essas representações geralmente são apresentadas na forma de alguma expressão lógica, árvore de decisão, regras de produção ou redes semânticas.

Dentre essas representações destacam-se as várias pesquisas feitas com árvores e regras de decisão. Os sistemas ID3 (Quinlan, 1986) e C4.5 (Quinlan, 1993) são exemplos de sistemas para indução de árvores de decisão que notadamente contribuíram nas pesquisas em Inteligência Artificial.

As pesquisas relacionadas com a indução de regras de decisão iniciaram a partir da tradução direta das árvores de decisão para regras, sobre as quais podiam-se aplicar em seguida técnicas de poda (Quinlan, 1987a; Quinlan, 1987b). Posteriormente, surgiram métodos para induzir regras diretamente dos dados, tal método é utilizado, por exemplo, pelo sistema CN2 (Clark & Niblett, 1989; Clark & Boswell, 1991).

O *paradigma estatístico* surgiu das pesquisas de estatísticos para criar métodos de classificação, que em vários casos se assemelham aos métodos empregados em Aprendizado de Máquina. Um exemplo de método criado por estatísticos para construir árvores de decisão é o CART (Breiman et al., 1984).

De uma forma geral as técnicas estatísticas tendem a focalizar tarefas em que todos os atributos têm valores contínuos ou ordinais, ou ainda alguns deles paramétricos. Os classificadores estatísticos geralmente assumem que valores de atributos estão normalmente distribuídos, e assim, usam os dados fornecidos para determinar média, variância e co-variância da distri-

buição.

Os sistemas que utilizam o *paradigma instance-based* podem ser exemplificados pela idéia de que classificar um caso é lembrar de um caso semelhante e sua classe correspondente, atribuindo ao novo caso a mesma classe. Desta forma, esses sistemas classificam casos nunca vistos através de casos similares conhecidos.

As características principais dos sistemas *instance-based* dizem respeito aos casos que devem ser lembrados, as similaridades entre os casos e a relação entre um novo caso e os casos existentes.

Saber quais os casos de treinamento que devem ser memorizados é importante para evitar dificuldades e lentidão de manuseio por parte do classificador. O ideal é reter apenas os casos com os quais seja possível resumir toda a informação. Estratégias a esse respeito são descritas em (Aha et al., 1991).

A medida de similaridade para casos nos quais todos os atributos são contínuos pode ser calculada pela raiz quadrada da soma dos quadrados da diferença dos atributos. Na presença de atributos ordinais esta medida se torna mais complicada, bem como na presença de atributos irrelevantes, que podem fazer com que dois casos similares sejam interpretados como muito diferentes. Métodos sensíveis ao contexto que alterem a escala dos atributos podem melhorar estas medidas (Stanfill & Waltz, 1986).

As redes neurais artificiais utilizam o *paradigma conexionista* e podem ser vistas como construções matemáticas relativamente simples. Essas redes foram inspiradas no modelo biológico do sistema nervoso e sua representação envolve unidades altamente interconectadas.

As redes neurais têm atualmente alcançado progresso e crescente interesse dos pesquisadores em vista de bons resultados provenientes tanto das pesquisas quanto das aplicações (Haykin, 1994).

O formalismo do *paradigma genético* é derivado do modelo evolucionário de aprendizado (Holland, 1986). Um classificador genético consiste de uma população de elementos de classificação que competem pela tarefa de predição. Tal como na teoria de Darwin onde sobrevivem os elementos mais adaptados ao meio ambiente, nesse paradigma os elementos mais fortes se reproduzem impedindo que o mesmo aconteça com aqueles de menor desempenho.

Para a geração de novos indivíduos os operadores genéticos básicos utilizados são *reprodução*, *cruzamento*, *mutação* e *inversão*. Esses operadores atuam no controle da quantidade de cópias produzidas de um indivíduo, na troca de material genético, na preservação de uma

espécie e na manutenção de uma certa diversidade da nova população.

2.2.2 Estratégias de Aprendizado

Considerando o aprendizado em geral sob o ponto de vista do aprendiz e do instrutor existem algumas estratégias básicas comumente relatadas que são aplicáveis tanto para aprendizado de conceitos quanto para qualquer modo de aquisição de conhecimento.

As estratégias de aprendizado por *hábito*, *instrução*, *dedução*, *analogia* e *indução* são apresentadas a seguir de acordo com a ordem crescente de complexidade da inferência que é desempenhada pelo aprendiz, como ilustra a Figura 2.3.

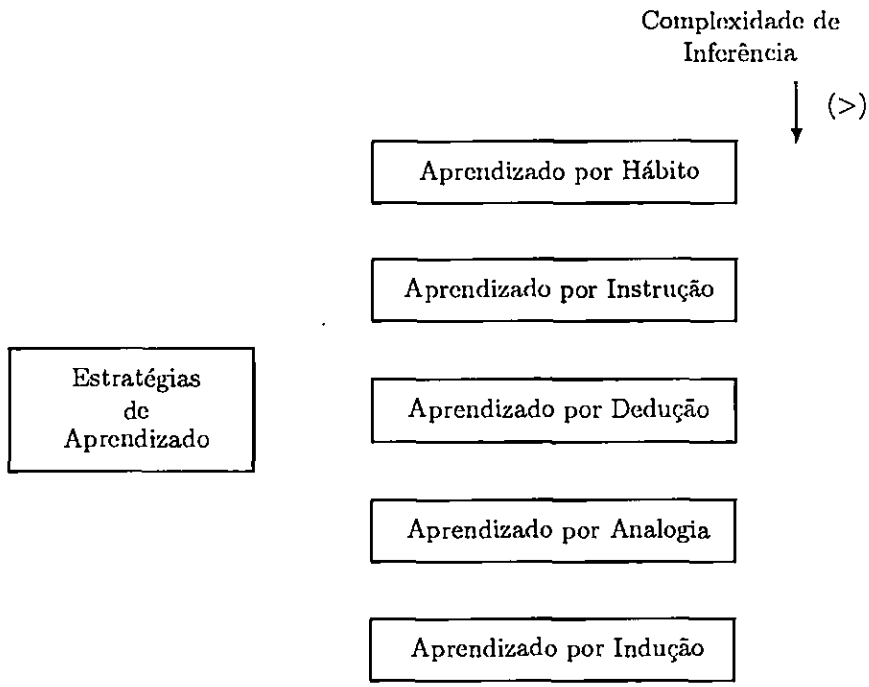

Figura 2.3: Classificação das Estratégias de AM (Monard et al., 1997)

No *aprendizado por hábito* o conhecimento é assimilado diretamente pelo aprendiz sem necessidade de aplicar alguma inferência. Em Aprendizado de Máquina, esta estratégia é empregada para reconhecer um conceito através de um algoritmo específico ou quando é contruída uma base de dados fatuais sobre o conceito.

Na estratégia de *aprendizado por instrução* o aprendiz adquire conceitos provenientes de uma fonte, mas não de forma direta. As informações fornecidas são analisadas a fim de

selecionar os fatos mais relevantes podendo ainda transformar tais informações em formas mais apropriadas para o aprendizado.

No *aprendizado por dedução* o conceito é adquirido através de dedução sobre o conceito já adquirido, o que significa dizer que esta estratégia inclui qualquer processo no qual o conhecimento aprendido é o resultado de uma transformação sobre um conhecimento já possuído que preserva a veracidade. Desta forma a inferência é fundamental atuando somente nas informações contidas no conhecimento (Raggett & Bains, 1992).

Na estratégia de *aprendizado por analogia* um novo conceito é adquirido ao modificar a definição de um conceito semelhante já conhecido. Desta forma, uma regra existente é modificada (por exemplo, observando as diferenças e similaridades entre o novo conceito e um conceito existente) de forma apropriada para que possa ser aplicada ao novo conceito. O aprendizado por analogia pode ser visto como a combinação do aprendizado indutivo com o dedutivo.

No *aprendizado por indução* o conceito é adquirido através de inferências indutivas sobre os fatos apresentados. No entanto, hipóteses geradas por essa inferência podem ou não preservar a veracidade. A indução é uma forma apropriada para permitir que conclusões gerais sejam obtidas de exemplos particulares.

2.2.3 Modos de Aprendizado

Sob o aspecto da classificação os sistemas de aprendizado podem trabalhar segundo dois modos: *supervisionado* e *não supervisionado*.

- No *aprendizado supervisionado*, dado um conjunto de observações onde as classes são conhecidas, o objetivo é encontrar regras capazes de classificar novas observações entre as classes já existentes.
- No *aprendizado não supervisionado* ou *clustering*, dado um conjunto de observações, o objetivo é tentar estabelecer a existência de classes ou *clusters* nesses dados.

2.2.4 Linguagens de Descrição

Em um sistema de aprendizado é necessário utilizar linguagens para representar exemplos ou instâncias, conceitos induzidos e conhecimento prévio sobre o domínio, denominado co-

nhecimento de fundo ou teoria de domínio. Essas linguagens são comumente referenciadas como *linguagens de descrição* denotadas por $\mathcal{L}_\mathcal{E}$, $\mathcal{L}_\mathcal{H}$ e $\mathcal{L}_\mathcal{D}$, ou seja

- $\mathcal{L}_\mathcal{E}$ é a linguagem de descrição de exemplos ou instâncias;
- $\mathcal{L}_\mathcal{H}$ é a linguagem de descrição de hipóteses ou conceitos aprendidos;
- $\mathcal{L}_\mathcal{D}$ é a linguagem de descrição da teoria de domínio ou conhecimento de fundo.

Exemplos de $\mathcal{L}_\mathcal{H}$ são *árvores de decisão* e *regras de decisão* e um exemplo de $\mathcal{L}_\mathcal{E}$ é *atributo-valor*. A Tabela 2.1 ilustra esse formato de representação onde a linha i refere-se ao i-ésimo exemplo e os elementos v_{ij} referem-se ao valor j-ésimo do atributo A_j do exemplo i, a coluna rotulada classe refere-se ao rótulo ou classe desse exemplo.

Exemplo	Atributos				Classe
	A_1	A_2	...	A_k	
1	v_{11}	v_{12}	...	v_{1k}	y_1
2	v_{21}	v_{22}	...	v_{2k}	y_2
...	...	...	...	...	...
n	v_{n1}	v_{n2}	...	v_{nk}	y_n

Tabela 2.1: Formato Atributo-Valor

A capacidade representacional das linguagens de descrição impoêm limites aos sistemas de aprendizado quanto aos conceitos que estes podem aprender. Por exemplo, na linguagem do tipo atributo-valor, relações entre objetos são difíceis de serem representadas, a solução é utilizar uma linguagem mais expressiva, tal como uma linguagem lógica de primeira ordem na qual é possível representar relações.

2.2.5 Formas de Aprendizado

Os sistemas de aprendizado indutivo podem ser classificados também segundo a forma *incremental* ou *não incremental* em que podem ser integrados os novos exemplos de treinamento.

- No aprendizado *incremental* é necessário que todos os exemplos de treinamento, simultaneamente, estejam disponíveis para que seja induzido um conceito. Geralmente essa forma é usada quando se sabe que os exemplos disponíveis nao sofrerão mudanças.
- No aprendizado *não incremental* a cada nova instância de treinamento, o conceito corrente pode ou não ser redefinido. O sistema considera os exemplos um a um durante

Modos de Aprendizado	Paradigmas de Aprendizado	Linguagens de Descrição	Formas de Aprendizado
Supervisionado	Simbólico	$\mathcal{L}_{\mathcal{E}}$ - Exemplos ou Instâncias	Incremental
Não Supervisionado	Estatístico	$\mathcal{L}_{\mathcal{H}}$ - Hipóteses ou Conceitos Aprendidos	Não Incremental
	Instance-based	$\mathcal{L}_{\mathcal{D}}$ - Teoria de Domínio ou Conhecimento de Fundo	
	Conexionista		
	Genético		

Tabela 2.2: Características Gerais de Sistemas de Aprendizado de Máquina (Félix, 1998)

o processo de aprendizado e de acordo com a sua classificação constrói ou altera a hipótese corrente.

Na Tabela 2.2 é apresentado um resumo de algumas das características gerais de sistemas de Aprendizado de Máquina.

2.3 Questões Fundamentais

Na pesquisa, desenvolvimento e aplicação de algoritmos de Aprendizado de Máquina para tarefas de classificação, muitas questões surgem e para entendê-las será utilizada uma taxonomia das diferentes classes de questões estatísticas. Na Figura 2.4, são consideradas nove dessas questões.

Iniciando na raiz da árvore que representa essa taxonomia, o primeiro ponto a ser considerado é se o estudo é feito sob um único domínio ou múltiplos domínios. Embora uma meta fundamental das pesquisas em Aprendizado de Máquina seja encontrar algoritmos de aprendizado que trabalhem bem em uma ampla variedade de domínios de aplicação, frequentemente as pesquisas estão aplicadas a um único domínio onde o objetivo é encontrar o melhor classificador ou o melhor algoritmo de aprendizado aplicado a este domínio.

Considerando o caso de um único domínio, existem dois conjuntos diferentes de questões dependendo se está sendo analisado os classificadores ou os algoritmos. Um classificador é uma função que, dado um exemplo de entrada, atribui este exemplo a uma das K classes. Um algoritmo de aprendizado é uma função que, dado um conjunto de exemplos e suas

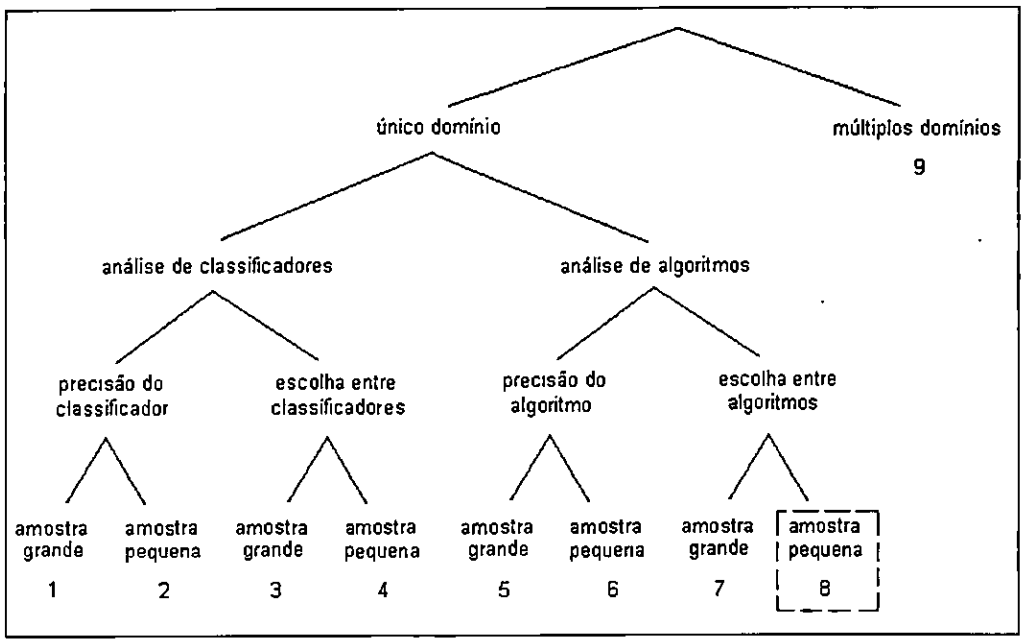

Figura 2.4: Taxonomia de Questões Estatísticas em AM (Dietterich, 1997)

classes, constrói um classificador.

Assim, em um determinado ambiente de aplicação o objetivo é encontrar o melhor classificador e estimar sua precisão sobre exemplos futuros. No entanto, em outras aplicações deve-se selecionar o melhor algoritmo de aprendizado em vez de encontrar o melhor classificador.

Seguindo a taxonomia mostrada na Figura 2.4, o próximo nível faz distinção entre duas tarefas fundamentais: estimar a precisão e escolher o melhor classificador (ou algoritmo) entre os classificadores disponíveis.

O nível mais inferior da taxonomia concentra-se na quantidade de dados disponíveis. Para dados abundantes não existem problemas em usá-los como conjunto de teste para avaliar classificadores, no entanto, quando os dados são limitados e deseja-se empregar todos eles como entrada para o algoritmo de aprendizado, deve-se usar alguma forma de *resampling* (por exemplo, *cross-validation* ou *bootstrap*) para executar a análise estatística (Weiss & Kulikowski, 1991).

Parte deste trabalho focaliza o tema da questão 8, onde dados dois algoritmos A e B e um pequeno conjunto de dados S deseja-se saber

Qual algoritmo produzirá o classificador mais preciso quando ambos são treinados

utilizando um conjunto de dados do mesmo tamanho que S?

Para responder esta questão deve-se usar vários métodos estatísticos para compará-los. Tendo em vista que S é pequeno, é necessário usar métodos de *holdout* e *resampling*, o que significa que a questão não pode ser respondida diretamente sem supor que o desempenho de dois algoritmos muda quando o tamanho do conjunto de treinamento também muda.

As questões 8 e 9 são as mais importantes para a pesquisa experimental em Aprendizado de Máquina. A questão 9 talvez seja a mais fundamental e difícil, pois trata exatamente sobre algoritmos produzirem classificadores mais precisos quando treinados com exemplos de novos domínios. No entanto, este trabalho se concentra em parte na questão 8 (porque normalmente se tem poucos exemplos de treinamento) que já requer um estudo minucioso o suficiente para uma pesquisa deste porte.

2.4 Considerações Finais

Neste Capítulo foram apresentadas algumas das características gerais de sistemas de Aprendizado de Máquina, tais como paradigmas de aprendizado, estratégias de aprendizado, formas e tipos de aprendizado e também as linguagens de descrição mais comuns.

Outro ponto abordado brevemente foi uma taxonomia de questões estatísticas, mostrando que muitas pesquisas em Aprendizado de Máquina podem ser feitas a partir de noves questões fundamentais. Dentro dessas nove questões foi caracterizado um dos pontos principais de investigação deste trabalho que é a questão do uso de algoritmos de aprendizado quando se tem poucos exemplos de treinamento e deseja-se conseguir classificadores que sejam os mais precisos possíveis.

Aprendizado de Máquina é uma área de pesquisa que já alcançou muitos resultados positivos, no entanto, ainda existem inúmeras questões importantes a serem investigadas. Uma destas questões é a avaliação do conhecimento adquirido, cujo tema é abordado no próximo capítulo.

Capítulo 3

Avaliação da Precisão de Hipóteses

3.1 Considerações Iniciais

Existem vários métodos estatísticos que podem ser aplicados para estimar a precisão de hipóteses. Três questões importantes a serem respondidas são (Mitchell, 1997):

1. Quando se conhece a precisão de uma hipótese sobre uma amostra limitada de dados, deseja-se saber *o quanto ela pode servir para estimar a precisão sobre novos exemplos?*
2. Quando uma hipótese tem melhor desempenho que outra sobre uma determinada amostra de exemplos, *qual a probabilidade que esta hipótese seja em geral mais precisa?*
3. Quando os dados disponíveis são limitados, *qual a melhor maneira de usá-los tanto para aprender uma hipótese quanto para estimar sua precisão?*

Estimar a precisão de uma hipótese quando se trabalha com uma amostra grande de dados não apresenta maiores problemas. No entanto, quando se trabalha com uma amostra limitada de dados, uma das dificuldades é que estas podem não representar bem a distribuição geral dos dados. Se este for o caso, a estimativa da precisão de uma hipótese utilizando essa amostra pode conduzir a falsos resultados.

O uso de métodos estatísticos junto com suposições referentes a distribuição dos dados permitem limitar a diferença que existe entre a precisão que é observada sobre a amostra dos dados disponíveis e a precisão verdadeira sobre a distribuição total dos dados (Horst et al., 1997; Horst et al., 1998).

Em geral, existem diversas dificuldades básicas para aprender uma hipótese e estimar sua precisão futura quando se trabalha com uma amostra limitada de dados. Essas dificuldades são, por exemplo, as relacionadas ao *bias na estimativa* e a *variância na estimativa.*

Quanto ao *bias*, a precisão observada de uma hipótese gerada com exemplos de treinamento é, frequentemente, o pior estimador da precisão sobre novos exemplos. Devido a que a hipótese gerada foi derivada daqueles exemplos, ela fornecerá uma estimativa otimistamente *biased* da precisão da hipótese sobre novos exemplos. Isso também acontece quando é considerado um espaço de hipóteses rico tornando possível o *overfitting* dos exemplos de treinamento. Por outro lado, uma estimativa geralmente será *unbiased* quando as hipóteses são testadas sobre conjunto de exemplos de teste escolhidos independentemente dos exemplos de treinamento e da hipótese.

A *variância* na estimativa pode aparecer mesmo quando a precisão de uma hipótese é medida sobre um conjunto *unbiased* de exemplos de teste, independentemente dos exemplos de treinamento. Esta precisão pode ainda variar em relação à precisão verdadeira em função da composição e do tamanho desse conjunto de exemplos de teste. Quanto menor for o conjunto de exemplos de treinamento, maior será a variância.

Neste capítulo são discutidos métodos para estimar a precisão de hipóteses, comparar a precisão de hipóteses, bem como comparar a precisão de dois algoritmos de aprendizado quando existem poucos dados disponíveis (Mitchell, 1997).

3.2 Estimativa da Precisão de Hipóteses

Ao analisar uma hipótese aprendida h sobre uma amostra de dados S que contém n exemplos extraídos aleatoriamente de acordo com a distribuição D, surgem duas questões de interesse:

1. *Qual é a melhor estimativa da precisão de h sobre futuras instâncias extraídas da mesma distribuição D?*
2. *Qual é o erro provável na estimativa dessa precisão?*

Estas questões podem ser respondidas e para isso é necessário distinguir entre duas noções de precisão ou, equivalentemente, de erro. Elas são o *erro amostral* e o *erro verdadeiro.*

O *erro amostral* é a taxa de erro de uma hipótese h sobre uma amostra de dados disponível S de instâncias extraídas da população X, ou seja, é a fração de S que h classifica erroneamente.

Definição 3.2.1 o *erro amostral* da hipótese h, denotado por $erro_S(h)$, em relação a função meta f e amostra de dados S é

$$erro_S(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x)) \tag{3.1}$$

onde n = número de exemplos em S e

$$\delta(f(x), h(x)) = \begin{cases} 1 & \text{se } f(x) \neq h(x) \\ 0 & \text{caso contrário} \end{cases}$$

Por outro lado, o *erro verdadeiro* é a taxa de erro de uma hipótese h sobre toda a distribuição desconhecida D de exemplos, isto é, a probabilidade que h classificará erroneamente uma única instância extraída aleatoriamente da distribuição D.

Definição 3.2.2 o *erro verdadeiro* da hipótese h, denotado por $erro_D(h)$ em relação a função meta f e distribuição D é

$$erro_D(h) \equiv \Pr_{x \in D}[f(x) \neq h(x)] \tag{3.2}$$

onde $\Pr_{x \in D}$ denota justamente que a probabilidade é considerada sobre a distribuição D.

Usualmente, procura-se pelo erro verdadeiro $erro_D(h)$ da hipótese, pois este é o erro esperado quando se aplica a hipótese a novos exemplos. No entanto, o que pode ser medido é apenas o erro amostral $erro_S(h)$. Em virtude disso, é preciso saber

Quão boa uma estimativa do $erro_D(h)$ *é fornecida pelo* $erro_S(h)$*?*

No caso de classificação, isto é, a hipótese h assume somente valores discretos, considerando que

1. a amostra S contém n instâncias retiradas uma a uma de acordo com a distribuição D, independente de h;
2. $n \geq 30$;
3. a hipótese h classifica erroneamente r desses n exemplos (ou instâncias), ou seja $erro_S(h) = \frac{r}{n}$;

então, se não existir outra informação, com base na teoria estatística é possível afirmar que o valor mais provável de $erro_D(h)$ é $erro_S(h)$ e, com aproximadamente 95% de probabilidade, o erro verdadeiro $erro_D(h)$ está no intervalo

$$erro_S(h) \pm 1.96\sqrt{\frac{erro_S(h)(1-erro_S(h))}{n}} \tag{3.3}$$

Por exemplo, se a amostra S contém $n = 50$ exemplos e a hipótese h classifica erroneamente $r = 10$ desses exemplos, então $erro_S(h) = \frac{10}{50} = 0.2$.

Se o experimento fosse repetido várias vezes retirando outras amostras S_1, S_2, etc., de 50 exemplos, espera-se que os erros dessas amostras $erro_{S_1}$, $erro_{S_2}$, etc., apresentem valores ligeiramente diferentes que $erro_S(h)$, mas será encontrado que para 95% desses experimentos o intervalo calculado contém o erro verdadeiro. Por isso, esse intervalo é denominado intervalo de confiança de 95% da estimativa para $erro_D(h)$. No exemplo considerado esse intervalo é dado por

$$0.2 \pm 1.96\sqrt{\frac{0.2 \cdot 0.8}{50}} = 0.2 \pm 0.110 \tag{3.4}$$

O intervalo de confiança de 95% definido pela Equação 3.3 pode ser generalizado para qualquer intervalo de confiança utilizando o valor da constante Z_N correspondente ao nível de confiança N% desejado, como mostra a Tabela 3.1

Nível de confiança N%:	50%	68%	80%	90%	95%	98%	99%
Constante Z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Tabela 3.1: Valores de Z_N para Intervalos de Confiança N% *Two-Sided*

Assim, a expressão geral para intervalo de confiança de N% da estimativa para $erro_D(h)$ é dado por

$$erro_S(h) \pm Z_N\sqrt{\frac{erro_S(h)(1-erro_S(h))}{n}} \tag{3.5}$$

Para o exemplo considerado, a Tabela 3.2 mostra os diferentes intervalos de confiança calculados com a Equação 3.5. Obviamente, intervalos com maior nível de confiança são maiores pois está se incrementando a probabilidade de $erro_D(h)$ estar nesse intervalo.

Pode ser observado que repetindo o experimento k vezes, isto é, retirando k amostras $S_1, S_2, ..., S_k$ de tamanho n, o erro dessas amostras $erro_{S_i}(h)$, $i = 1, ..., k$ é uma variável aleatória binomial — Apêndice A. Assim, a probabilidade da hipótese h classificar erronea-

$N\%$	$Z_N\sqrt{\frac{erro_S(h)(1-erro_S(h))}{n}}$	Intervalo de Confiança
50%	0.037	[0.163 , 0.237]
68%	0.056	[0.144 , 0.256]
80%	0.072	[0.128 , 0.272]
90%	0.092	[0.108 , 0.292]
95%	0.110	[0.090 , 0.310]
98%	0.131	[0.069 , 0.331]
99%	0.145	[0.055 , 0.345]

Tabela 3.2: Intervalos de Confiança para $n = 50$ e $r = 10$

mente r desses n exemplos está dada por

$$\Pr[\#erro_S(h) = r] = \binom{n}{r} p^r(1-p)^{n-r},\ r = 0, ..., n. \tag{3.6}$$

A probabilidade p, desconhecida, representa justamente o erro verdadeiro $erro_D(h)$ e a idéia é estimar p testando a hipótese h nas amostras S_i. Em outras palavras, sabendo que a variável aleatória $erro_S(h)$ obedece a distribuição Binomial, qual a provável diferença entre o erro amostral $erro_S(h) = \frac{r}{n}$ e o erro verdadeiro $erro_D(h) = p$? Em termos estatísticos $erro_S(h)$ é um estimador do erro verdadeiro $erro_D(h)$.

Nas próximas seções são apresentados os conceitos estatísticos básicos[1] que justificam os procedimentos aqui apresentados.

3.2.1 Estimadores, *Bias* e Variância

Em geral, um estimador é qualquer variável aleatória usada para estimar algum parâmetro da população da qual a amostra é extraída. Uma questão óbvia referente a qualquer estimador é tentar saber se na média ele fornece uma estimativa correta. Define-se o *bias de estimação* como sendo a diferença entre o valor esperado do estimador e o valor verdadeiro do parâmetro.

Definição 3.2.3 o *bias de estimação* de um estimador Y para um parâmetro arbitrário p é $E[Y] - p$.

Se o *bias* de estimação é zero, diz-se que Y é um estimador *unbiased* para p. Deve-se notar que isso ocorrerá se a média de muitos valores aleatórios de Y gerados por experimentos

[1]No Apêndice A encontram-se as definições dos termos utilizados

aleatórios repetidos (ou seja, $E[Y]$) convergem para p. Pode ser observado que $erro_S(h)$ é um estimador *unbiased* para $erro_D(h)$ tendo em vista que para uma distribuição Binomial o valor esperado de r é igual a np. Assim, dado que n é constante, o valor esperado de $\frac{r}{n}$ é p.

Quando testa-se uma hipótese utilizando os exemplos de treinamento, tem-se uma estimativa otimistamente *biased* de erro da hipótese. Portanto, para obter uma estimativa *unbiased* de $erro_D(h)$, a hipótese h e a amostra S devem ser escolhidas independentemente.

Outra propriedade importante de qualquer estimador é sua variância. Tendo aternativas de estimadores *unbiased* faz sentido escolher aquele que possui a menor variância, pois pela definição de variância, essa escolha produzirá o menor erro quadrado esperado entre o valor estimado e o valor verdadeiro do parâmetro.

Em geral, dado r erros em uma amostra de n exemplos de teste extraídos independentemente, o desvio padrão para $erro_S(h)$ é dado por

$$\sigma_{erro_S(h)} = \frac{\sigma_r}{n} = \frac{\sqrt{np(1-p)}}{n} = \sqrt{\frac{p(1-p)}{n}} \tag{3.7}$$

o qual pode ser aproximado, substituindo p por $\frac{r}{n} = erro_S(h)$

$$\sigma_{erro_S(h)} \approx \sqrt{\frac{erro_S(h)(1-erro_S(h))}{n}} \tag{3.8}$$

3.2.2 Intervalos de Confiança

Uma maneira comum de descrever a incerteza associada com uma estimativa é fornecer um intervalo dentro do qual é esperado que o valor verdadeiro esteja, juntamente com a sua probabilidade de estar nesse intervalo. Tais estimativas são chamadas de *estimativas de intervalo de confiança.*

Definição 3.2.4 um *intervalo de confiança* de $N\%$ para algum parâmetro p é o intervalo onde se espera conter p com probabilidade de $N\%$.

A primeira pergunta que surge é

Como derivar intervalos de confiança para $erro_D(h)$?

Derivar o intervalo de confiança para $erro_D(h)$ é simples, já que é conhecido que a distribuição de probabilidade Binomial governa o estimador $erro_S(h)$. A média dessa distribuição é

$erro_D(h)$ e o desvio padrão é dado pela Equação 3.8. Portanto, para derivar um intevalo de confiança de 95%, é somente preciso encontrar o intervalo centrado em torno do valor médio $erro_D(h)$, que seja amplo o suficiente para conter 95% da probabilidade total dessa distribuição. Isso fornece um intervalo contendo $erro_D(h)$, dentro do qual $erro_S(h)$ deve cair 95% das vezes. Equivalentemente, fornece o tamanho do intervalo que contém $erro_S(h)$ dentro do qual $erro_D(h)$ deve cair 95% das vezes.

Outra pergunta que surge é

> *Para um dado N, como encontrar o tamanho do intervalo que contém $N\%$ da massa de probabilidades?*

No caso da distribuição Binomial, este cálculo é trabalhoso. Entretanto, na maioria dos casos a distribuição Binomial é bem aproximada pela distribuição Normal $N(np, \sqrt{np(1-p)})$ — Apêndice A. A distribuição Normal padronizada, $N(0,1)$, encontra-se tabelada especificando o tamanho do intervalo, em torno da média, que contém $N\%$ da distribuição acumulada — Tabela 3.1, pg. 18. Esta é justamente a informação necessária para realizar o cálculo do intervalo de confiança $N\%$ como mostrado na Seção 3.2, pois dada uma variável aleatória X que obedece a distribuição Normal $N(\mu, \sigma)$ então:

- o valor aleatório medido y de Y estará $N\%$ das vezes no intervalo

$$\mu \pm Z_N \sigma \tag{3.9}$$

- a média μ estará $N\%$ das vezes no intervalo

$$y \pm Z_N \sigma \tag{3.10}$$

Utilizando na Equação 3.10, o valor da média e desvio padrão — Equação 3.8 — de $erro_S(h)$ obtém-se a expressão geral para intervalo de confiança $N\%$ da estimativa para $erro_D(h)$,

$$erro_S(h) \pm Z_N \sqrt{\frac{erro_S(h)(1 - erro_S(h))}{n}}$$

que é a Equação 3.5 utilizada na Seção 3.2.

É muito importante observar que esta expressão é válida somente no caso de classificação, ou seja, exemplos com classe categórica. Além disso, duas aproximações foram realizadas para obtê-la:

1. o erro verdadeiro $erro_D(h) = p$ na Equação 3.7 foi aproximado por $erro_S(h)$ para obter o valor aproximado do desvio padrão σ do $erro_S(h)$ na Equação 3.8.

2. a distribuição Binomial foi aproximada pela distribuição Normal.

Em termos estatísticos estas duas aproximações são consideradas muito boas sempre que $n \geq 30$ ou quando $np(1-p) \geq 5$. Para valores de n menores deve ser utilizada a própria distribuição Binomial.

O intervalo de confiança visto anteriormente está limitado *two-sided*, ou seja, ele limita a quantidade estimada tanto no seu limite superior quanto inferior. Em alguns casos o interesse é somente em limites *one-sided*, por exemplo, quando é necessário responder questões tais como

> *Qual a probabilidade que $erro_D(h)$ seja no máximo U (limite superior do intervalo de confiança)?*

Para encontrar tais limites de erro *one-sided* existe uma modifição simples do procedimento descrito na Secão 3.2.3. Isso vem do fato de que a distribuição Normal é simétrica em torno da sua média. Em função disto, qualquer intervalo de confiança *two-sided* baseado na distribuição Normal pode ser convertido para um intervalo de confiança *one-sided* com duas vezes a confiança.

3.2.3 Uma Metodologia para Derivar Intervalos de Confiança

Foi descrito anteriormente como derivar estimativas de intervalo de confiança quando se deseja estimar $erro_D(h)$ para uma hipótese h, de valores discretos, obtida utilizando uma amostra de n instâncias de exemplos de treinamento extraídos independentemente de uma distribuição D.

A descrição feita aqui ilustra uma metodologia geral aplicada a muitos dos problemas de estimativa, que pode ser visto como o problema de estimar a média (valor esperado) de uma população com base na média de uma amostra de tamanho n extraída aleatoriamente. Este processo geral inclui os seguintes passos (Mitchell, 1997):

1. Identificar qual o parâmetro p da população a ser estimado, por exemplo, $erro_D(h)$.

2. Definir o estimador Y, por exemplo $erro_S(h)$. Como já comentado, é desejável escolher uma variância mínima e estimador *unbiased*.

3. Determinar a distribuição de probabilidade D_Y que governa o estimador Y, incluindo sua média e variância.

4. Determinar o intervalo de confiança de $N\%$ encontrando os limites inferior e superior denominados L e U respectivamente, tal que $N\%$ da massa na distribuição de probabilidades D_Y encontre-se entre L e U.

A justificativa desta metodologia encontra-se no teorema do limite central — Apêndice A.

Na próxima seção é tratada a segunda questão considerada no início deste capítulo, isto é, quando uma hipótese tem melhor desempenho que outra, qual a probabilidade que esta hipótese seja em geral mais precisa?

3.3 Comparação de Hipóteses

Sejam duas hipóteses h_1 e h_2 para alguma função meta f discretamente valorada. Considerando que h_1 foi testada em uma amostra S_1 contendo n_1 exemplos extraídos aleatoriamente, e h_2 foi igualmente testada em uma amostra independente S_2 contendo n_2 exemplos extraídos da mesma distribuição D, então, a diferença d entre os erros verdadeiros dessas duas hipóteses, que é o parâmetro a ser estimado, é dado por

$$d \equiv erro_D(h_1) - erro_D(h_2) \tag{3.11}$$

Utilizando a metodologia descrita na Seção 3.2.3, após identificar esse parâmetro (passo 1) o passo seguinte (passo 2) é definir um estimador. Neste caso o estimador é definido como a diferença entre os erros amostrais[2], denotado por $\hat{d}$

$$\hat{d} \equiv erro_{S_1}(h_1) - erro_{S_2}(h_2) \tag{3.12}$$

O próximo passo (passo 3) é determinar a distribuição de probabilidade que governa a variável aleatória $\hat{d}$. Sabe-se que para n_1 e n_2 grandes (≥ 30), tanto $erro_{S_1}(h_1)$ quanto

[2]É possível provar que $\hat{d}$ fornece uma estimativa *unbiased* de d; ou seja $E[\hat{d}] = d$.

$erro_{S_2}(h_2)$ têm distribuições que são aproximadamente Normal. Devido ao fato que a diferença de suas distribuições Normais é também uma distribuição Normal, então $\hat{d}$ tem uma distribuição aproximadamente Normal com média d e sua variância é a soma das variâncias de $erro_{S_1}(h_1)$ e $erro_{S_2}(h_2)$.

Utilizando a Equação 3.8 que aproxima a variância de cada uma dessas distribuições obtém-se a variância aproximada das distribuições dos dois erros

$$\sigma_{\hat{d}}^2 \approx \frac{erro_{S_1}(h_1)(1 - erro_{S_1}(h_1))}{n_1} + \frac{erro_{S_2}(h_2)(1 - erro_{S_2}(h_2))}{n_2} \tag{3.13}$$

Determinada a distribuição de probabilidades que governa o estimador $\hat{d}$, é simples derivar intervalos de confiança que caracterizem o provável erro ao estimar d usando $\hat{d}$. Uma estimativa de intervalo de confiança $N\%$ para d é aproximada por:

$$\hat{d} \pm Z_N \sqrt{\frac{erro_{S_1}(h_1)(1 - erro_{S_1}(h_1))}{n_1} + \frac{erro_{S_2}(h_2)(1 - erro_{S_2}(h_2))}{n_2}} \tag{3.14}$$

onde Z_N é a constante descrita na Tabela 3.1.

Com a Equação 3.14 obtém-se o intervalo de confiança *two-sided* para estimar d. No entanto, para obter limites *one-sided* (limitando a maior ou a menor diferença possível em erros com algum nível de confiança) essa equação deve ser modificada, como comentado na Seção 3.2.2.

Apesar da análise anterior considerar o caso em que h_1 e h_2 são testadas em amostras de dados independentes, é geralmente aceitável usar o intervalo de confiança visto na Equação 3.14 no caso de h_1 e h_2 serem testadas em uma única amostra S (onde S é também independente de h_1 e h_2). Neste último caso $\hat{d}$ é redefinido como

$$\hat{d} \equiv erro_S(h_1) - erro_S(h_2) \tag{3.15}$$

A variância deste novo $\hat{d}$, considerando S_1 e S_2 como S, tende a ser menor que a variância dada pela Equação 3.13. Isto deve-se ao fato que usando uma única amostra S elimina-se a variância devido as diferenças aleatórias nas composições de S_1 e S_2. Neste caso, o intervalo de confiança dado pela Equação 3.14 geralmente será um intervalo bastante conservador, no entanto, ainda correto.

3.4 Testando Hipóteses

Em alguns casos, além de determinar o intervalo de confiança para algum parâmetro há o interesse de encontrar a probabilidade de alguma suposição ser verdadeira. Assim, surgem

questões como

Qual a probabilidade que $erro_D(h_1) > erro_D(h_2)$*?*

Por exemplo, supondo que são medidos os erros amostrais para h_1 e h_2 usando duas amostras independentes S_1 e S_2 de tamanho 100 e é encontrado que $erro_{S_1}(h_1) = 0.30$ e $erro_{S_2}(h_2) = 0.20$, então a diferença observada é $\hat{d} = 0.10$. Entretanto, devido a variações aleatórias nas amostras, é possível observar essa mesma diferença entre os erros das amostras, mesmo no caso de ser $erro_D(h_1) \leq erro_D(h_2)$. Assim, surgem as seguintes questões:

Qual a probabilidade que $erro_D(h_1) > erro_D(h_2)$*, dado que a diferença nos erros da amostra é* $\hat{d} = 0.10$*?*

e equivalentemente

Qual a probabilidade que $d > 0$*, dado que* $\hat{d} = 0.10$*?*

Nota-se que a probabilidade $\Pr(d > 0)$ é igual a probabilidade que $\hat{d}$ não tenha superestimado d para um valor maior que 0.10. Por outro lado, esta é a probabilidade que $\hat{d}$ caia dentro de um intervalo *one-sided* $\hat{d} < d + 0.10$. Desde que d seja a média da distribuição que governa $\hat{d}$ esse intervalo pode ser expresso como $\hat{d} < \mu_{\hat{d}} + 0.10$.

Para resumir, a probabilidade $\Pr(d > 0)$ é igual a probabilidade de $\hat{d}$ cair dentro do intervalo *one-sided* $\hat{d} < \mu_{\hat{d}} + 0.10$. Uma vez que já tenha sido calculada a distribuição aproximada que governa $\hat{d}$, pode-se determinar a probabilidade que $\hat{d}$ caia dentro deste intervalo *one-sided* ao calcular a massa da probabilidade da distribuição $\hat{d}$ dentro desse intervalo.

Para realizar esse cálculo, o intervalo pode ser re-expresso em termos do número de desvios padrões com que ele se afasta do meio. Usando a Equação 3.13 encontra-se que $\sigma_{\hat{d}} \approx 0.061$, então pode-se re-expressar o intervalo como, aproximadamente

$$\hat{d} < \mu_{\hat{d}} + 1.64\sigma_{\hat{d}} \tag{3.16}$$

Quando deseja-se saber

Qual o nível de confiança associado com esse intervalo one-sided para uma distribuição Normal?

deve ser consultada a Tabela 3.1, onde encontra-se que 1.64 desvios padrões sobre a média corresponde ao intervalo *two-sided* com nível de confiança de 90%, assim, o intervalo *one-sided* tem um nível de confiança associado de 95%.

Tendo em vista que $\hat{d} = 0.10$, a probabilidade que $erro_D(h_1) > erro_D(h_2)$ é aproximadamente 0.95, ou seja, em termos estatísticos diz-se que é aceita a hipótese que "$erro_D(h_1) > erro_D(h_2)$" com 0.95 de confiança ou, alternativamente, diz-se que é rejeitado a hipótese oposta (também chamada de hipótese nula) em um nível $(1 - 0.95) = 0.05$ de significância.

3.5 Comparação de Algoritmos de Aprendizado

No caso de se querer comparar o desempenho de dois algoritmos de aprendizado L_A e L_B em vez de duas hipóteses específicas, surge a seguinte questão

> *Qual o teste apropriado para comparar algoritmos de aprendizado e como determinar quando uma diferença entre os algoritmos é estatisticamente significante ou não?*

O primeiro passo é especificar o parâmetro a ser estimado — Seção 3.2.3 — para determinar qual entre os dois algoritmos, L_A e L_B, é melhor, na média, para aprender a função meta f.

Uma maneira razoável de definir o que está "na média" é considerar o desempenho relativo desses dois algoritmos considerando a média do desempenho de ambos algoritmos sobre todos os conjuntos de treinamento de tamanho n que podem ser extraídos de acordo com a distribuição D. Ou seja, deseja-se estimar o valor esperado da diferença dos erros entre os dois algoritmos

$$E_{S \subset D}[erro_D(L_A(S)) - erro_D(L_B(S))] \tag{3.17}$$

onde $L(S)$ representa a hipótese encontrada pelo algoritmo L utilizando os exemplos de treinamento da amostra S. O subscrito $S \subset D$ indica que o valor esperado E é calculado sobre amostras S extraídas de acordo com a distribuição D.

No entanto, na prática, quando se deseja comparar dois algoritmos de aprendizado, geralmente o que se tem é apenas uma amostra limitada D_0 dos dados. Nesse caso, uma maneira de estimar o valor da Equação 3.17 é dividir D_0 em um conjunto de treinamento S_0 e um conjunto disjunto de teste T_0. Assim, S_0 pode ser usado para treinar tanto L_A quanto L_B e T_0 pode ser usado para comparar a precisão das duas hipóteses aprendidas através da

seguinte equação:

$$erro_{T_0}(L_A(S_0)) - erro_{T_0}(L_B(S_0)) \tag{3.18}$$

Deve ser observado que existem duas diferenças fundamentais entre esse estimador e o estimador definido pela Equação 3.17, elas são:

1. $erro_{T_0}(h)$ é usado para aproximar $erro_D(h)$;
2. o valor medido é a diferença entre os erros apenas para um conjunto de treinamento S_0 ao invés de medir o valor esperado dessa diferença sobre todas as amostras S que podem ser extraídas de acordo com a distribuição D.

A estimativa obtida pela Equação 3.18 pode ser melhorada particionando repetidamente os dados D_0 em conjuntos disjuntos de treinamento e teste calculando a média dos erros sobre os conjuntos de teste para esses diferentes experimentos. Essa idéia é melhor descrita pelo procedimento na Tabela 3.3, o qual pode ser utilizado para estimar a diferença entre os erros de dois algoritmos de aprendizado baseado na amostra D_0 de dados disponíveis.

Esse procedimento primeiro particiona os dados em k subconjuntos disjuntos de igual tamanho, sendo que esse tamanho deve ser pelo menos 30. Depois treina e testa os algoritmos de aprendizado k vezes utilizando cada vez um dos k subconjuntos como conjunto de teste e o subconjunto restante como conjunto de treinamento. Dessa forma, os algoritmos de aprendizado são testados sobre k conjuntos de testes independentes e a diferença média dos erros, denotada por $\overline{\delta}$, é retornada como uma estimativa do valor esperado da diferença dos erros entre os dois algoritmos de aprendizado. Ou seja, $\overline{\delta}$ é uma estimativa de

$$E_{S \subset D_0}[erro_D(L_A(S)) - erro_D(L_B(S))] \tag{3.19}$$

onde S representa uma amostra aleatória de tamanho $\frac{k-1}{k}|D_0|$ extraída aleatoriamente de D_0.

A única diferença entre a Equação original 3.17 e a Equação 3.19 é que, nesta última, o valor esperado é calculado sobre subconjuntos do conjunto de dados disponíveis D_0, e não sobre subconjuntos retirados da população total de acordo com a distribuição D.

O intervalo de confiança $N\%$ para estimar o valor definido pela Equação 3.19 é dado por

$$\overline{\delta} \pm t_{N,k-1} s_{\overline{\delta}} \tag{3.20}$$

1. Particionar os dados disponíveis D_0 em k subconjuntos disjuntos $T_1, T_2, \ldots, T_k$ de igual tamanho — pelo menos 30 exemplos em cada T_i
2. Para i de 1 até k,
   utilizar T_i como conjunto de teste e o conjunto restante como conjunto de treinamento S_i
   - $S_i \leftarrow \{D_0 - T_i\}$
   - $h_A \leftarrow L_A(S_i)$
   - $h_B \leftarrow L_B(S_i)$
   - $\delta_i \leftarrow erro_{T_i}(h_A) - erro_{T_i}(h_B)$
3. Retornar o valor $\overline{\delta}$, onde

$$\overline{\delta} \equiv \frac{1}{k} \sum_{i=1}^{k} \delta_i$$

Tabela 3.3: Procedimento para Estimar a Diferença do Erro entre dois Algoritmos de Aprendizado L_A e L_B

onde $t_{N,k-1}$ é uma constante análoga a Z_N mas para a distribuição t e $s_{\overline{\delta}}$ é definida por

$$s_{\overline{\delta}} \equiv \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (\delta_i - \overline{\delta})^2} \tag{3.21}$$

é uma estimativa do desvio padrão da distribuição t que governa a variável aleatória δ.

O primeiro subscrito N na constante $t_{N,k-1}$ da Equação 3.20 representa o nível de confiança. O segundo subscrito, usualmente denotado v, é o número de graus de liberdade. Esse número está relacionado com o número de testes independentes realizados para encontrar o valor da variável aleatória δ que, neste caso, é $v = k - 1$.

A Tabela 3.4 mostra alguns valores de $t_{N,v}$, que se aproximam a Z_N para $v \rightarrow \infty$.

	N% Intervalo de Confiança			
	90%	95%	98%	99%
$v = 2$	2.92	4.30	6.96	9.92
$v = 5$	2.02	2.57	3.36	4.03
$v = 10$	1.81	2.23	2.76	3.17
$v = 20$	1.72	2.09	2.53	2.84
$v = 30$	1.70	2.04	2.46	2.75
$v = 120$	1.66	1.98	2.36	2.62
$v = \infty$	1.64	1.96	2.33	2.58

Tabela 3.4: Valores de $t_{N,v}$ para Intervalos de Confiança *Two-Sided*

Deve ser observado que o procedimento utilizado para estimar $\overline{\delta}$, descrito na Tabela 3.3, considera amostras idênticas — denominado de testes pareados — enquanto que o método descrito na Seção 3.3 utiliza amostras independentes.

Em geral, testes pareados produzem intervalos de confiança menores já que qualquer diferença nos erros observados deve-se às diferenças entre as hipóteses. Por outro lado, quando as hipóteses são testadas em amostras diferentes, a diferença do erro entre duas amostras pode ser atribuído não somente às diferenças entre as hipóteses mas também devido as diferenças nessas duas amostras.

Um outro teste estatístico simples está baseado na diferença entre a taxa de erro de cada um dos dois algoritmos L_A e L_B, ou seja, os erros não são pareados e as partições T_1, T_2,...,T_k na Tabela 3.3 não são, necessariamente, idênticas para ambos algoritmos. Neste caso, são conhecidos os valores $\overline{\delta}_A = \frac{1}{k}\sum_{i=1}^{k}\delta_{i_A}$ e $\overline{\delta}_B = \frac{1}{k}\sum_{i=1}^{k}\delta_{i_B}$ onde $\delta_{i_A} = erro_{T_{i_A}}(h_A)$ e $\delta_{i_B} = erro_{T_{i_B}}(h_B)$, bem como os desvios padrões $\sigma_{\overline{\delta}_A}$ e $\sigma_{\overline{\delta}_B}$. Após realizar diversas aproximações e no caso de se verificar que

$$\frac{|\overline{\delta}_A - \overline{\delta}_B|}{\sqrt{\frac{\sigma^2_{\overline{\delta}_A} + \sigma^2_{\overline{\delta}_B}}{2}}} > 2 \tag{3.22}$$

pode-se considerar que existe uma diferença, com 95% de nível de confiança, de um algoritmo superar o outro. Se $\overline{\delta}_A > \overline{\delta}_B$ então o algoritmo L_B supera o algoritmo L_A, caso contrário L_A supera L_B.

A fim de exemplificar a comparação de dois algoritmos de aprendizado utilizando este último teste, foram considerados um subconjunto de algoritmos, dados e medidas realizadas e publicadas em (Baranauskas & Monard, 1999). Dentre os diversos algoritmos analisados nesse trabalho, foram escolhidos os algoritmos simbólicos $C4.5$ (Quinlan, 1993) e $CN2$ (Clark & Niblett, 1989), disponíveis na biblioteca $MLC++$ (Félix et al., 1998; Kohavi et al., 1994), os quais são bastante conhecidos pela comunidade de Aprendizado de Máquina.

A Tabela 3.5 mostra o subconjunto escolhido do conjunto de dados analisados nesse trabalho. Nessa tabela, para cada conjunto de dados é mostrado em cada coluna: o número de exemplos (#E), número de atributos (#A) conforme os tipos contínuo (c) e discreto (d), número de classes (#C), erro majoritário e se o conjunto de dados possui valores desconhecidos. Os conjuntos de dados são apresentados em ordem crescente do número de atributos.

A Figura 3.1 mostra a dimensionalidade dos conjuntos de dados, isto é, número de atributos e número de instâncias de cada conjunto de dados. Devido a grande variação, o número de instâncias na Figura 3.1 é representado como $\log_{10}$(#Instâncias).

Cada um dos conjuntos de dados está relacionado com algum domínio específico conforme a descrição resumida a seguir:

Conjunto de Dados	#E	#A(c,d)	#C	Erro Majoritário	Valores Desconhecidos
bupa	345	6 (6,0)	2	42.03%	N
pima	769	8 (8,0)	2	34.98%	N
hungaria	294	13 (13,0)	2	36.05%	S
crx	690	15 (6,9)	2	44.49%	S
letter	15000	16 (16,0)	26	95.92%	S
hepatitis	155	19 (6,13)	2	20.65%	S
sonar	208	60 (60,0)	2	46.63%	N
dna	3186	180 (0,180)	3	48.09%	N

Tabela 3.5: Descrição dos Conjuntos de Dados

- bupa: para predizer quando um paciente masculino terá ou não desordens hepáticas baseado em vários testes de sangue e na quantia de consumo de álcool;
- pima: para predizer quando um paciente terá ou não teste positivo para diabetes de acordo com critérios da Organização Mundial de Saúde;
- hungaria: está relacionado com o diagnóstico de doenças do coração;
- crx: está relacionado com aplicações de cartões de crédito;
- letter: para reconhecer uma das 26 letras maiúsculas do alfabeto inglês identificadas como pontos em preto e branco dentro de um retângulo;
- hepatitis: está relacionado com a expectativa de vida de pacientes com hepatite;
- sonar: para predizer diferenças entre sinais de sonar transmitidos de metais e sinais transmitidos de rocha;
- dna: relacionado com a área de biologia molecular trata sobre características de DNA.

Desses conjuntos de dados, dna foi obtido do Projeto StatLog (Taylor et al., 1994) e os restantes são do repositório UCI Irvine (Blake et al., 1998).

Para cada um dos conjuntos de dados foram aplicados os algoritmos de aprendizado $C4.5$ e $CN2$ executados com seus parâmetros *default* utilizando a biblioteca MLC++. Na Tabela 3.6 são mostradas as taxas de erro (média e desvio padrão das 10-*fold stratified cross-validation*[3]) de $C4.5$ e $CN2$ para cada um dos conjuntos de dados (Baranauskas & Monard, 1999). Nessa

[3]A diferença entre *k-fold* e *k-fold stratified cross-validation* é que neste último caso procura-se manter a distribuição das classes em cada uma das k partições.

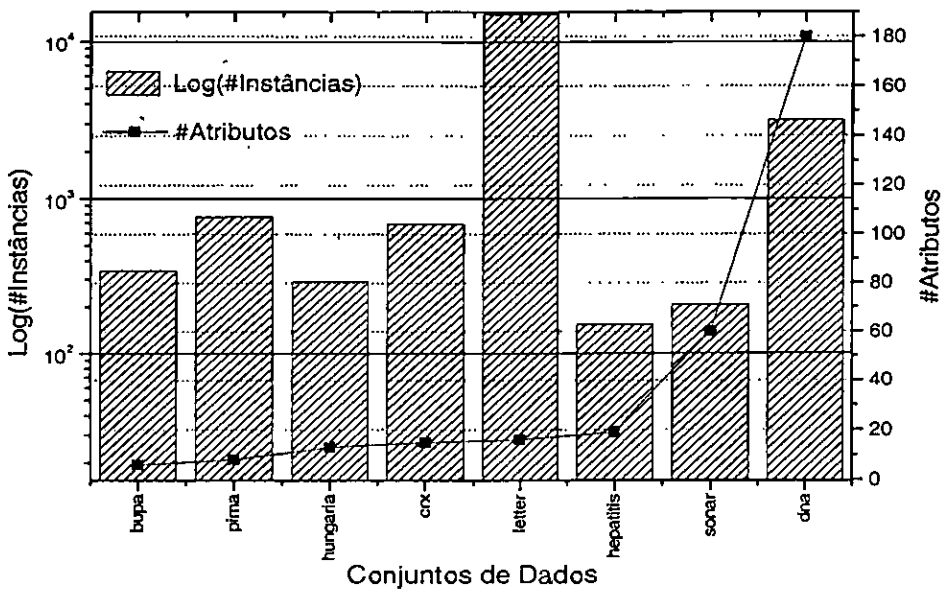

Figura 3.1: Dimensionalidade dos Conjuntos de Dados

tabela também é mostrado, na última coluna, o valor da diferença entre as duas taxas de erro calculado conforme a Equação 3.22.

Para determinar quando a diferença entre os dois algoritmos (L_A=$\mathcal{C}4.5$ e L_B=$\mathcal{CN}2$) é significativa ou não, é mostrado na Figura 3.2 um gráfico de barras, no qual cada barra representa a medida definida pela Equação 3.22.

Interpretando a Figura 3.2, qualquer barra que tenha comprimento maior que 2 indica que esse resultado é significante com nível de confiança de 95%. Quando a barra está abaixo de zero indica que o primeiro algoritmo ($\mathcal{C}4.5$) supera o segundo algoritmo ($\mathcal{CN}2$); se a barra está acima do zero indica o contrário, isto é, que o segundo algoritmo supera o primeiro. Assim, quando o comprimento da barra está abaixo de 2 indica que o primeiro algoritmo supera significantemente o segundo algoritmo, no caso contrário, o segundo supera significantemente o primeiro.

Pode-se observar na Figura 3.2 que apenas para os conjuntos de dados dna e letter o algoritmo $\mathcal{C}4.5$ supera significantemente o algoritmo $\mathcal{CN}2$ com nível de confiança de pelo menos 95%, nos demais conjuntos de dados a diferença entre os dois algoritmos não é significante. Outro fato, bem conhecido na comunidade de Aprendizado de Máquina, e também constatado na Figura 3.2, refere-se a impossibilidade de afirmar que um algoritmo de aprendizado será sempre melhor que outro em todas as situações, pois os conjuntos de dados influenciam

Conjunto de Dados	$C4.5$	$CN2$	Diferença entre Taxas de Erro
bupa	32.29 ± 1.73	32.18 ± 2.11	0.06 (+)
pima	25.74 ± 1.13	25.38 ± 1.38	0.29 (+)
hungaria	22.48 ± 4.20	22.07 ± 3.06	0.11 (+)
crx	15.65 ± 1.18	16.80 ± 1.21	0.96 (−)
letter	13.41 ± 0.34	29.66 ± 0.30	50.68 (−)
hepatitis	20.62 ± 2.27	18.25 ± 3.83	0.75 (+)
sonar	30.26 ± 1.97	28.81 ± 3.30	0.53 (+)
dna	7.60 ± 0.46	11.85 ± 0.62	7.79 (−)

Tabela 3.6: Taxas de Erro Obtidas para $C4.5$ e $CN2$ Utilizando 10-*fold stratified cross-validation*

bastante no desempenho dos algoritmos de aprendizado.

Deve ser observado que existem vários outros métodos, propostos na literatura, para comparar algoritmos de aprendizado. Nos últimos anos, têm aparecido debates ativos na comunidade de Aprendizado de Máquina, relacionados à eficácia de vários desses métodos, especialmente quando aplicados a conjuntos de exemplos pequenos.

Um dos problemas é que a maioria das aproximações estatísticas realizadas para encontrar uma estimativa do valor $E_{S \subset D}$, Equação 3.17, pg. 26, são válidas sempre que as medidas sejam realizadas sobre conjuntos independentes. Entretanto, se considerarmos a partição dos dados disponíveis, D_0, em k subconjuntos disjuntos $T_1, T_2, ..., T_k$, como proposto no procedimento descrito na Tabela 3.3, página 28, é possível observar que, ainda que cada subconjunto de teste seja independente do outro, isso não se verifica para os conjuntos de treinamento, os quais não são independentes. Na realidade, todo par de conjunto de treinamento compartilha 80% dos exemplos em D_0.

Um outro problema é que o tamanho dos conjuntos de teste é $\frac{1}{k}|D_0|$. Assim, o número de exemplos de teste pode não ser suficientemente grande para aproximar a distribuição Binomial pela Normal. No excelente trabalho de (Dietterich, 1996) é apresentada uma discussão sobre testes estatísticos para determinar quando um algoritmo é significantemente melhor que outro numa determinada tarefa de aprendizado. Esses testes estatísticos são comparados experimentalmente a fim de medir a probabilidade de cometer um erro de Tipo I — Apêndice A. Um erro de Tipo I acontece quando a hipótese nula é verdadeira, isto é, não existe diferença entre os algoritmos L_A e L_B, mas a hipótese nula é rejeitada. Também

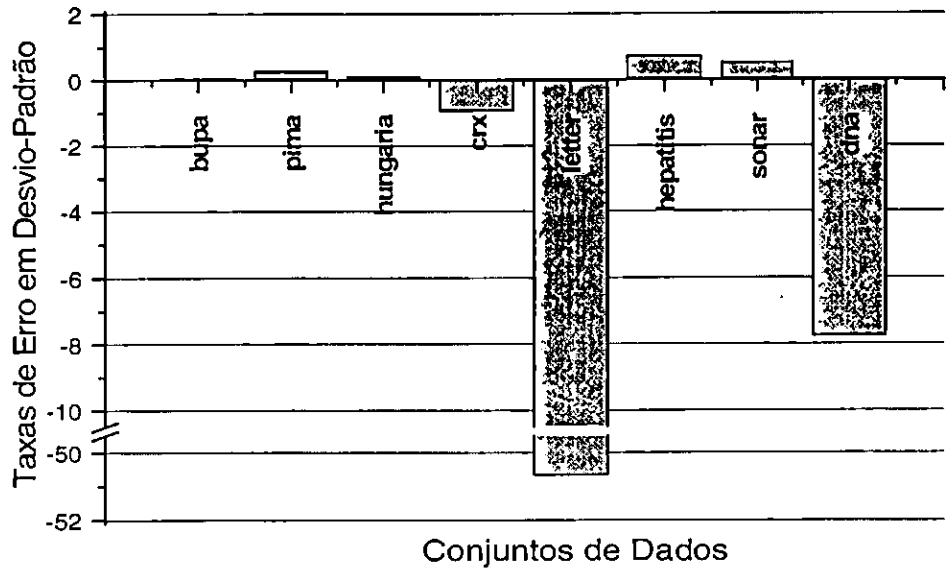

Figura 3.2: Diferença Absoluta em Desvios Padrões das Taxas de Erro para $C4.5$ e $CN2$

em (Salzberg, 1997) são discutidos alguns testes estatísticos para comparar algoritmos de aprendizado.

Segundo (Dietterich, 1996) o teste pareado descrito na Tabela 3.3 apresenta geralmente um bom desempenho, enquanto que o teste da Equação 3.22 pode apresentar, em algumas situações, uma alta probabilidade de erro do Tipo I.

3.6 Considerações Finais

Neste capítulo foram discutidos alguns dos métodos utilizados para estimar a precisão de hipóteses, comparar a precisão de hipóteses e de algoritmos de aprendizado quando existem poucos dados disponíveis. Parte do desenvolvimento deste capítulo bem como a notação utilizada estão baseadas no Capítulo 5 de (Mitchell, 1997).

Como mencionado anteriomente, existem debates ativos na comunidade relacionados aos estimadores estatísticos utilizados para comparar corretamente o desempenho de algoritmos de aprendizado. Consideramos esse um tema muito importante que merece a atenção dos pesquisadores. Entretanto, um tratamento mais aprofundado foge aos objetivos deste trabalho.

Deve ser observado que tão importante quanto a avaliação de hipóteses objetivando calcular e estimar sua precisão é a avaliação de hipóteses com objetivos de medir a sua qualidade e interessabilidade. Esse tema é discutido no próximo capítulo.

Capítulo 4

Medidas de Qualidade e Interessabilidade do Conhecimento Adquirido

4.1 Considerações Iniciais

Em função da crescente aplicação de sistemas de aprendizado indutivos em problemas do mundo real, muitos deles envolvendo risco, surge a necessidade de se ter maior confiabilidade no conhecimento adquirido, não somente em termos da precisão de hipóteses como considerado no capítulo anterior mas também em termos da qualidade e interessabilidade que o conhecimento adquirido apresenta.

Com a aplicação dos sistemas de aprendizado indutivos existentes, aparecem dois problemas comuns:

- dificuldade de fornecer conhecimento confiável sob todas as circunstâncias (por exemplo, quando há variações nos dados de origem do conhecimento adquirido, no tipo de representação desse conhecimento e outros);
- falta de medidas de qualidade sempre confiáveis (por exemplo, que possam ser utilizadas em todos os domínios de aplicação) para servir de suporte aos usuários, tais como engenheiros de conhecimento ou engenheiros de processos.

Motivados por vários fatores, como por exemplo os citados acima, pesquisadores têm es-

tudado o uso de técnicas estatísticas para solucionar, pelo menos em parte, os problemas relacionados com a avaliação da qualidade do conhecimento adquirido. Um outro importante ponto que tem sido mais recentemente pesquisado é a avaliação da interessabilidade desse conhecimento a fim de procurar por conhecimento realmente interessante para o usuário.

A avaliação da qualidade bem como da interessabilidade está relacionada com a interpretação do conhecimento adquirido.

4.2 Interpretação do Conhecimento Adquirido

Há duas maneiras particulares para ver o conhecimento que foi adquirido por um determinado sistema de aprendizado por exemplos. A primeira é uma visão genérica, como a sugerida na Figura 4.1, onde a hipótese h representa o conhecimento adquirido visto como uma "caixa preta". Nesse caso não está sendo considerado o paradigma do sistema de aprendizado nem a linguagem de descrição da hipótese gerada, ou seja, h é vista simplesmente como um classificador no qual é estimada a precisão do classificador, isto é, da hipótese h aprendida como um todo.

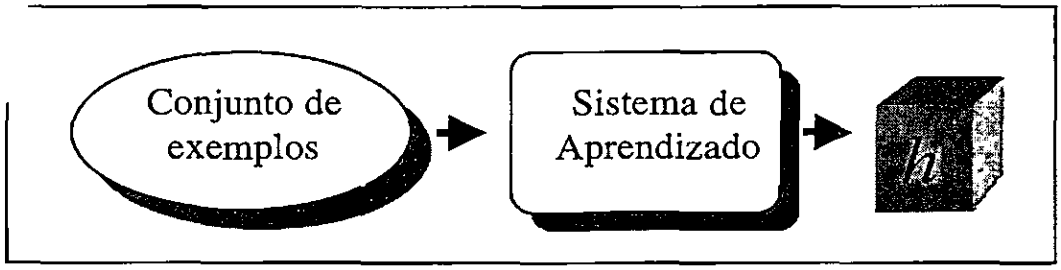

Figura 4.1: h vista como Classificador

A segunda maneira é uma visão mais apurada, onde é levada em consideração a linguagem de descrição na qual h está expressa, conforme ilustrado na Figura 4.2. Nesse caso a "caixa preta" é aberta e cada uma das regras R_i, bem como subconjuntos de regras nas quais h está expressa, podem ser analisadas.

Deve ser notado que algumas vezes, mesmo quando a precisão global do classificador não é considerada boa, podem existir algumas regras boas em termos de qualidade e interessabilidade. Este é um dos fatores que motivam várias pesquisas tanto na área de Aprendizado de Máquina quanto em Data Mining.

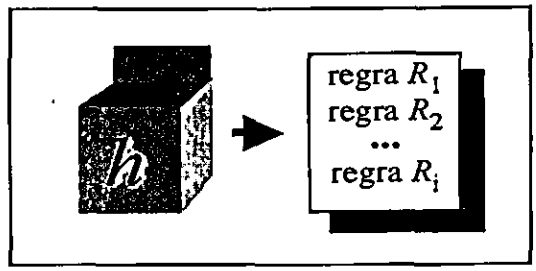

Figura 4.2: h vista como Conjunto de Regras

Neste trabalho, h é tratada sob estas duas interpretações, ou seja, de forma genérica considerando a precisão de h como tratado no Capítulo 3, e de forma particular, como regras geradas por um sistema de aprendizado que expressam esse conhecimento.

Em geral, as medidas de avaliação consideram, entre outros critérios, a qualidade das regras em termos de precisão, cobertura, completeza e consistência. Quanto a avaliação da interessabilidade de regras, as medidas tentam refletir o quanto de conhecimento interessante uma regra apresenta para o usuário, utilizando-se por exemplo de métodos estatísticos e da teoria da informação.

Ambos os aspectos, qualidade e interessabilidade, são importantes para que o usuário (especialista ou analista do domínio) possa se focalizar nas regras de maior qualidade e interesse em vez de examinar uma a uma todas as regras descobertas pelo sistema de aprendizado.

Isto deve-se ao fato de que vários dos métodos usualmente utilizados para adquirir conhecimento a partir de conjuntos de exemplos descobrem um grande número de regras. O número elevado de regras é um dos fatores que mais dificulta a análise das regras por parte do usuário. Outro fator está relacionado com a complexidade da regra, a qual refere-se por exemplo ao número de condições ou tipo de representação.

Independentemente da quantidade de regras, essas medidas de avaliação ajudam a selecionar aquelas regras que são mais úteis e interessantes, pois parte do conhecimento adquirido dos exemplos pode ser muito óbvio ou irrelevante.

Várias medidas já foram pesquisadas com a finalidade de auxiliar o usuário no entendimento e utilização do conhecimento adquirido por sistemas de aprendizado por exemplos. Embora as medidas de qualidade e interessabilidade sejam apresentadas neste trabalho separadamente (Seção 4.3 e Seção 4.5 respectivamente) algumas delas servem para os dois propósitos, ou seja, avaliar tanto a qualidade quanto a interessabilidade de regras.

4.3 Medidas de Qualidade de Regras

Existem vários trabalhos sobre técnicas estatísticas e propostas de medidas referentes a qualidade de regras que são encontrados na literatura. Um desses trabalhos é (Dean & Famili, 1997) no qual é apresentado um estudo comparativo entre várias dessas técnicas aplicadas a um domínio particular.

As medidas apresentadas nesta seção utilizam como base uma tabela de contingência, como a mostrada na Tabela 4.1 juntamente com a definição de cada um dos valores. A tabela de contingência relaciona uma regra R (ou h) com a classe C.

Tabela de Contingência

	classe C	classe não C	
regra R (ou h) cobre	rc	$r\bar{c}$	r
regra R (ou h) não cobre	$\bar{r}c$	$\bar{r}\bar{c}$	$\bar{r}$
	c	$\bar{c}$	n

rc = número de exemplos cobertos por R (ou h) que pertencem a classe C
$r\bar{c}$ = número de exemplos cobertos por R (ou h) que não pertencem a classe C
$\bar{r}c$ = número de exemplos não cobertos por R (ou h) que pertencem a classe C
$\bar{r}\bar{c}$ = número de exemplos não cobertos por R (ou h) e que não pertencem a classe C
r = número total de exemplos cobertos por R (ou h)
$\bar{r}$ = número total de exemplos não cobertos por R (ou h)
c = número total de exemplos na classe C
$\bar{c}$ = número total de exemplos que não pertencem a classe C
n = número total de exemplos

Tabela 4.1: Tabela de Contingência 2 x 2

Algumas medidas de qualidade de regras são brevemente apresentadas a seguir. A fim de exemplificar cada uma dessas medidas, serão consideradas duas regras R_1 e R_2 que pertencem a um conjunto hipotético de regras, relativas ao mesmo conjunto de $n = 600$ exemplos de treinamento com duas classes possíveis, tal que $c = 350$ e $\bar{c} = 250$. A Tabela 4.2 mostra a cobertura de cada uma dessas duas regras.

4.3.1 Consistência e Completeza

Duas medidas bastante úteis para calcular a qualidade de regras são a consistência e a completeza. Consistência é uma medida do quanto uma regra é especifica para o problema.

	classe C	classe não C	
regra R_1 cobre	$rc_1 = 160$	$r\bar{c}_1 = 40$	$r_1 = 200$
regra R_1 não cobre	$\bar{r}c_1 = 190$	$\bar{r}\bar{c}_1 = 210$	$\bar{r}_1 = 400$
regra R_2 cobre	$rc_2 = 60$	$r\bar{c}_2 = 15$	$r_2 = 75$
regra R_2 não cobre	$\bar{r}c_2 = 290$	$\bar{r}\bar{c}_2 = 235$	$\bar{r}_2 = 525$

Tabela 4.2: Cobertura das Regras R_1 e R_2

Quanto mais alta a consistência mais precisamente a regra cobre a classe em questão. Ela está no seu nível máximo quando a regra cobre somente as instâncias da classe e nenhuma instância fora desta classe, isto é, quando não existem erros positivos. A consistência de uma regra R é definida por

$$Cons(R) = \frac{rc}{r} \tag{4.1}$$

Considerando as duas regras R_1 e R_2 da Tabela 4.2, a consistência de ambas as regras resulta respectivamente em

$$\begin{aligned} Cons(R_1) &= \frac{160}{200} \\ &= 0.800 \end{aligned}$$

$$\begin{aligned} Cons(R_2) &= \frac{60}{75} \\ &= 0.800 \end{aligned}$$

Deve ser observado que esta medida não reflete o número de exemplos cobertos pela regra. Por exemplo, as regras R_1 e R_2 possuem a mesma medida de consistência, isto é, $Cons(R_1) = Cons(R_2) = 0.800$, ainda que R_1 cobre mais exemplos que R_2. A completeza, definida a seguir, considera esse aspecto.

Completeza é uma medida do quanto do domínio do problema é coberto pela regra. Quanto mais alta a completeza, mais instâncias são cobertas pela regra. A completeza está em seu nível máximo quando cada instância da classe C está coberta pela regra, isto é, quando não existem erros negativos. A completeza de uma regra R é definida por

$$Comp(R) = \frac{rc}{c} \tag{4.2}$$

Ambas as Equações 4.1 e 4.2 retornam um valor real entre 0 e 1.

Considerando as duas regras R_1 e R_2 da Tabela 4.2, a completeza de ambas as regras resulta

respectivamente em

$$\begin{aligned} Comp(R_1) &= \frac{160}{350} \\ &= 0.457 \end{aligned}$$

$$\begin{aligned} Comp(R_2) &= \frac{60}{350} \\ &= 0.171 \end{aligned}$$

refletindo assim o maior número de exemplos cobertos por R_1. Tanto a hipótese h, quanto cada uma (ou um subconjunto) das regras que a constituem, pode ser classificada conforme visto na Figura 4.3 quanto a cobertura de exemplos em:

- h *consistente* e *completa* não cobre os exemplos negativos e cobre todos os exemplos positivos.
- h *consistente* e *incompleta* não cobre exemplos negativos e não cobre todos os exemplos positivos.
- h *inconsistente* e *completa* cobre alguns exemplos negativos e cobre todos os exemplos positivos.
- h *inconsistente* e *incompleta* cobre alguns exemplos negativos e não cobre todos os exemplos positivos.

4.3.2 Estatística de Cohen

A estatística de Cohen busca encontrar um *nível de associação* na diagonal principal da tabela de contingência, para servir como medida da qualidade de uma regra R. O cálculo é feito combinando os valores de consistência e completeza de uma regra (Equações 4.1 e 4.2 respectivamente) com o número total de exemplos e o número de exemplos de uma classe cobertos pela regra R, através da seguinte equação:

$$Q_{Cohen}(R) = \frac{n \cdot Cons(R) \cdot Comp(R) - rc}{n \cdot \frac{Cons(R)+Comp(R)}{2} - rc} \tag{4.3}$$

O valor retornado pela Equação 4.3 está entre -1 e 1. Os valores negativos representam um relacionamento inverso entre uma regra e a classe por ela predita.

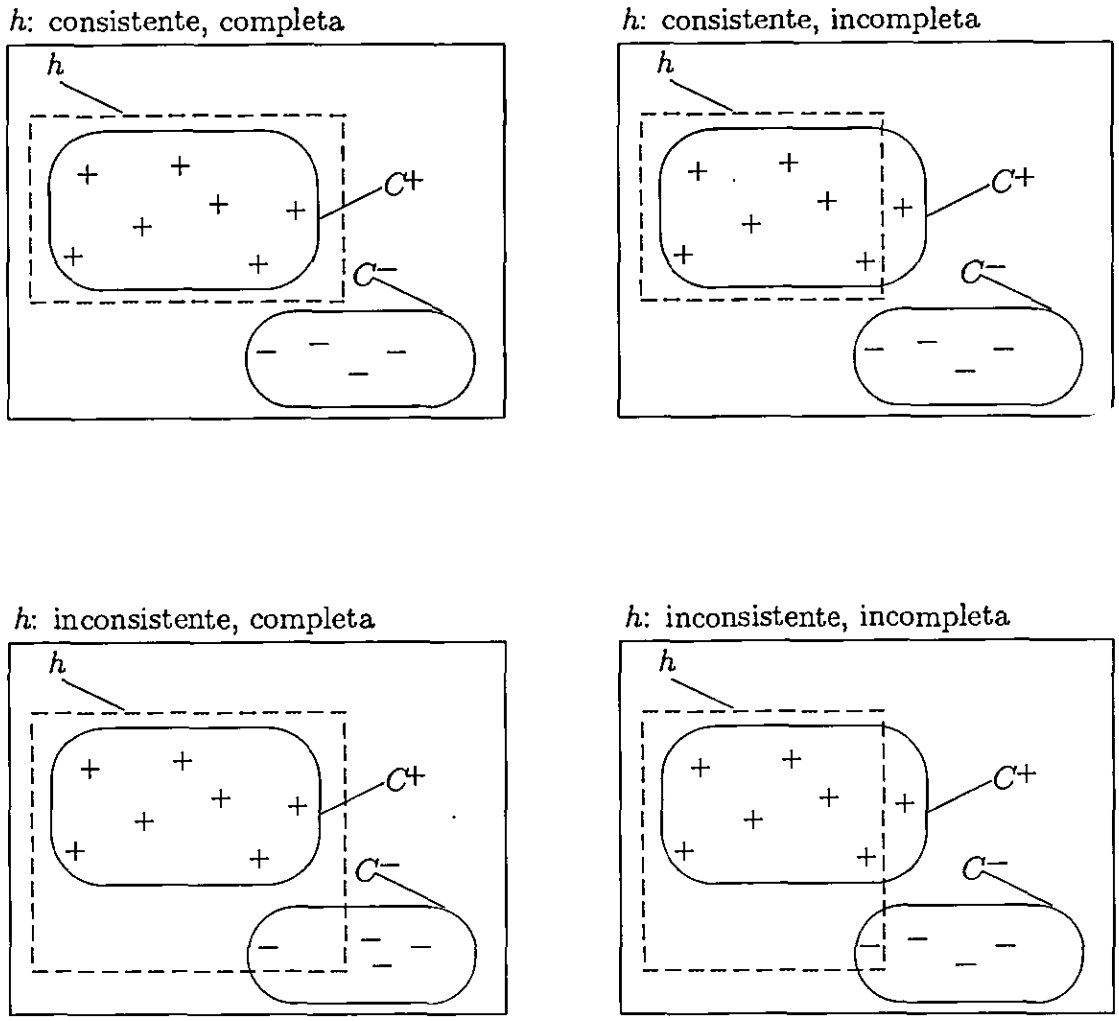

Figura 4.3: Consistência e Completeza de uma Hipótese h (Lavrač & Džeroski, 1994)

Considerando as duas regras R_1 e R_2 da Tabela 4.2, a estatística de Cohen de ambas as regras resulta respectivamente em

$$\begin{aligned} Q_{Cohen}(R_1) &= \frac{600 \cdot 0.800 \cdot 0.457 - 160}{600 \cdot \frac{0.800+0.457}{2} - 160} \\ &= 0.273 \end{aligned}$$

$$\begin{aligned} Q_{Cohen}(R_2) &= \frac{600 \cdot 0.800 \cdot 0.171 - 60}{600 \cdot \frac{0.800+0.171}{2} - 60} \\ &= 0.095 \end{aligned}$$

Segundo esta estatística, a regra R_1 é melhor que a regra R_2.

4.3.3 Estatística IMAFO

A estatística IMAFO (Torgo, 1993) apresenta um valor real entre 0 e 10 para medir a qualidade de uma regra R. O valor é calculado pela seguinte equação:

$$Q_{IMAFO}(R) = (AC_R \cdot E_C) \cdot 10 \tag{4.4}$$

onde AC_R é a precisão da regra R, calculada como segue:

$$AC_R = \frac{rc + \overline{rc}}{n} \tag{4.5}$$

e E_C é a estimativa de cobertura da regra R calculada pela seguinte equação:

$$E_C = \exp\left(\frac{rc}{c} - 1\right) \tag{4.6}$$

onde exp retorna $e^{(\frac{rc}{c}-1)}$, sendo $e \cong 2.718282$ o número de Euler.

Embora sejam esperados bons resultados dessa estatística na prática, existem dificuldades quanto a sua interpretação.

Considerando as duas regras R_1 e R_2 da Tabela 4.2, a estatística IMAFO de ambas as regras resulta respectivamente em

$$\begin{aligned} Q_{IMAFO}(R_1) &= \left(\frac{160+210}{600} \cdot \exp\left(\frac{160}{350} - 1\right)\right) \cdot 10 \\ &= 3.583 \end{aligned}$$

$$\begin{aligned} Q_{IMAFO}(R_2) &= \left(\frac{60+235}{600} \cdot \exp\left(\frac{60}{250} - 1\right)\right) \cdot 10 \\ &= 2.146 \end{aligned}$$

Segundo esta estatística, a regra R_1 é melhor que a regra R_2.

4.3.4 Estatística de Coleman

A estatística de Coleman é uma medida da "combinação" entre a primeira coluna e cada linha da tabela de contingência. No caso de uma tabela de contingência, tal como a Tabela 4.1, a relação correta é entre a primeira coluna e a primeira linha. Esta medida é calculada de forma similar a estatística de Cohen com a seguinte equação:

$$Q_{Coleman}(R) = \frac{n \cdot Cons(R) \cdot Comp(R) - rc}{n \cdot Comp(R) - rc} \tag{4.7}$$

Da mesma forma que a estatística de Cohen, o valor retornado está entre -1 e 1, com valores negativos significando também um relacionamento inverso entre uma regra e a classe por ela predita. No entanto, existe um problema com a estatística de Coleman, pois esta não extrai a completeza de uma regra. Isso a torna incapaz de medir o efeito de falsos negativos na qualidade da regra, pois o *bias* no sistema de indução favorecerá os falsos negativos sobre os falsos positivos.

Considerando as duas regras R_1 e R_2 da Tabela 4.2, a estatística de Coleman resulta no mesmo valor para ambas as regras, como mostrado a seguir

$$\begin{aligned} Q_{Coleman}(R_1) &= \frac{600 \cdot 0.800 \cdot 0.457 - 160}{600 \cdot 0.457 - 160} \\ &= 0.520 \end{aligned}$$

$$\begin{aligned} Q_{Coleman}(R_2) &= \frac{600 \cdot 0.800 \cdot 0.171 - 60}{600 \cdot 0.171 - 60} \\ &= 0.520 \end{aligned}$$

4.3.5 SKIB1 e SKIB2

As estatísticas SKIB1 e SKIB2 são combinações das estatísticas de Coleman e Cohen e têm sido sugeridas como medidas que aproveitam as melhores características de ambas. As respectivas equações são:

$$Q_{SKIB1}(R) = Q_{Coleman}(R) \cdot \frac{2 + Q_{Cohen}(R)}{3} \tag{4.8}$$

$$Q_{SKIB2}(R) = Q_{Coleman}(R) \cdot \frac{1 + Comp(R)}{2} \tag{4.9}$$

Estas são medidas da combinação e associação, isto é, elas extraem valores de toda a tabela de contingência para retornar uma qualidade de regra. Deve ser observado que essas estatísticas lidam bem com muitos dos problemas de distribuição de classes que causam confusão em outras medidas. Desta forma, parecem ser medidas de qualidade bastante úteis.

Considerando as duas regras R_1 e R_2 da Tabela 4.2, as estatísticas SKIB1 e SKIB2 de ambas as regras resultam respectivamente em

$$\begin{aligned} Q_{SKIB1}(R_1) &= 0.520 \cdot \frac{2 + 0.273}{3} \\ &= 0.394 \end{aligned}$$

$$
\begin{aligned}
Q_{SKIB1}(R_2) &= 0.520 \cdot \frac{2 + 0.095}{3} \\
&= 0.363
\end{aligned}
$$

$$
\begin{aligned}
Q_{SKIB2}(R_1) &= 0.520 \cdot \frac{1 + 0.457}{2} \\
&= 0.379
\end{aligned}
$$

$$
\begin{aligned}
Q_{SKIB2}(R_2) &= 0.520 \cdot \frac{1 + 0.171}{2} \\
&= 0.304
\end{aligned}
$$

Segundo estas estatísticas, a regra R_1 é melhor que a regra R_2.

4.3.6 Uso da Informação

A teoria da informação é outra área que está estritamente relacionada à estatística, e pode oferecer alguma ajuda para medicões de qualidade. A seguinte equação tenta calcular o ganho de informação resultante de uma regra particular:

$$Q_{IKIB}(R) = -\log \frac{c}{n} + \log Cons(R) \tag{4.10}$$

onde todos os logs estão na base 2. No entanto, somente é válida se

$$Cons(R) \geq \frac{c}{n} \tag{4.11}$$

Assim, o resultado deve ser usado como um limite mínimo de qualidade da regra. Tal como a estatística de Coleman, essa medida falha ao incorporar a completeza da regra gerando as mesmas consequências indesejáveis.

Considerando as duas regras R_1 e R_2 da Tabela 4.2, essa medida resulta no mesmo valor para ambas as regras, como mostrado a seguir

$$
\begin{aligned}
Q_{IKIB}(R_1) &= -\log \frac{350}{600} + \log 0.800 \\
&= 0.456
\end{aligned}
$$

$$
\begin{aligned}
Q_{IKIB}(R_2) &= -\log \frac{350}{600} + \log 0.800 \\
&= 0.456
\end{aligned}
$$

A Tabela 4.3 mostra o intervalo de valores — mínimo e máximo — para cada uma das medidas de qualidade descritas e mostra também os valores obtidos para as regras R_1 e R_2 utilizando cada uma dessas medidas. Deve ser observado que o limite máximo da medida $Q_{IKIB}(R)$ é dado por $-\log \frac{c}{n}$ na base 2, e portanto, no caso das regras R_1 e R_2 o valor é $-\log \frac{350}{600} = 0.778$.

Medida	Min	Max	Regras	
			R_1	R_2
$Cons(R)$	0	1	0.800	0.800
$Comp(R)$	0	1	0.457	0.171
$Q_{Cohen}(R)$	-1	1	0.273	0.095
$Q_{IMAFO}(R)$	0	10	3.583	2.146
$Q_{Coleman}(R)$	-1	1	0.520	0.520
$Q_{SKIB1}(R)$	-1	1	0.394	0.363
$Q_{SKIB2}(R)$	-1	1	0.379	0.304
$Q_{IKIB}(R)$	0	$-\log \frac{c}{n}$	0.456	0.456

Tabela 4.3: Algumas Medidas de Qualidade das Regras R_1 e R_2

4.4 Interessabilidade de Regras

A interessabilidade é uma maneira de avaliar regras tentando capturar o quanto de conhecimento interessante (ou inesperado) elas apresentam. As medidas de interessabilidade estão baseadas em vários aspectos, principalmente na utilidade que as regras representam para o usuário.

4.4.1 Aspectos da Interessabilidade

Dois aspectos de medidas de interessabilidade de regras são:

1. **acionabilidade**: regras são consideradas interessantes se o usuário pode tirar alguma vantagem utilizando elas (Piatetsky-Shapiro & Matheus, 1994);

2. **inesperabilidade**: regras são consideradas interessantes se elas são inesperadas, ou seja, causam "surpresa" ao usuário (Silberschatz & Tuzhilin, 1996).

A acionabilidade é uma medida importante, principalmente porque os usuários frequentemente estão interessados nas vantagens que podem conseguir tomando ações com base no

novo conhecimento adquirido. No entanto, não é fácil decidir quais regras são acionáveis; uma maneira de saber isso é verificar seu uso na prática.

A inesperabilidade pode ser tanto subjetiva quanto objetiva. Como medida subjetiva, ela está estritamente relacionada com as crenças ou impressões gerais do usuário. As crenças podem ser classificadas em dois tipos: crenças rígidas e crenças flexíveis. As crenças rígidas são as restrições que não podem ser mudadas com novas evidências, enquanto que as crenças flexíveis são aquelas nas quais se espera que alguma mudança ocorra frente a novas evidências.

Como medida objetiva, a inesperabilidade está estritamente relacionada com os dados da regra, assim, ela é autônoma e pode ser computada.

Esses dois aspectos, acionabilidade e inesperabilidade, das medidas de interessabilidade não são mutuamente exclusivos e podem dar lugar as seguintes três categorias (Silberschatz & Tuzhilin, 1996):

1. regras que são tanto inesperadas quanto acionáveis;
2. regras que são inesperadas mas não acionáveis;
3. regras que são acionáveis mas esperadas.

As regras da primeira e segunda categoria podem ser obtidas pela busca de regras inesperadas, enquanto que as regras da terceira categoria podem ser obtidas pela busca de regras que estejam em conformidade com conceitos prévios do usuário.

No que se segue é considerado que dado o conjunto de n exemplos na forma atributo-valor, como descrito na Tabela 2.1, página 11, o sistema de aprendizado gera regras de classificação com a seguinte representação: $A \rightarrow B$, isto é, **if** A **then** B, mais especificamente:

$$\textbf{if } \underbrace{cond_1 \textbf{ and } cond_2 \ \ldots \ \textbf{and } cond_m}_{A} \textbf{ then } \underbrace{\text{classe } C_k}_{B} \tag{4.12}$$

onde:

C_k representa um dos possíveis j valores da classe, isto é, $C = \{C_1, C_2, \ldots, C_j\}$;
$cond_i$, $i = 1 \ldots m$, é uma tripla da forma $< At \ op \ Val >$ onde:
At é um atributo;
op é um operador relacional que pertence ao conjunto $\{<, >, \leq, \geq, =\}$;
Val é um dos possíveis valores do domínio de At.

Em geral, se o atributo At for contínuo então op pertence ao conjunto $\{<, >, \leq, \geq\}$ enquanto que se o atributo At for discreto op é o operador "=".

Como visto anteriormente na Seção 4.3, três fatores que são comumente considerados para avaliar regras são: cobertura, completeza e consistência da regra. A consistência de uma regra é também denominado fator de confiança. Na notação de regras esses fatores podem ser expressos da seguinte maneira:

1. cobertura: $|A|$, ou seja, é o número de exemplos cobertos pelo antecedente de uma regra;
2. completeza: $\frac{|A\&B|}{|B|}$ onde $|A\&B|$ significa o número de exemplos cobertos pelo antecedente e consequente da regra e $|B|$ é o número de exemplos cobertos apenas pelo consequente da regra;
3. consistência: $\frac{|A\&B|}{|A|}$.

4.4.2 Princípios para a Interessabilidade

Encontrar as regras mais interessantes geradas por algum sistema de aprendizado por exemplos, significa procurar por aquelas regras com maior valor para alguma(s) medida(s) de interessabilidade.

Com base nos seguintes fatores de qualidade de uma regra: cobertura, completeza e fator de confiança, foram propostos por (Piatetsky-Shapiro, 1991) três princípios básicos desejáveis para uma medida de Interessabilidade de Regras (IR). Ele conjecturou que todas as medidas satisfazendo esses princípios atribuiriam alto valor para regras fortes e baixo valor para regras fracas. Uma regra forte pode ser definida como uma descrição de uma regularidade para muitos exemplos com alta confiança e uma regra fraca representa para um número menor de exemplos, uma regularidade com alta confiança (Suzuki & Kodratoff, 1998).

Os princípios são os seguintes:

1. $IR = 0$ se $|A\&B| = \frac{|A||B|}{n}$. Se A e B são estatisticamente independentes, a regra não é interessante.
2. IR aumenta monotonicamente com $|A\&B|$ quando outros parâmetros permanecem fixos;

3. IR diminui monotonicamente com $|A|$ ou $|B|$ quando outros parâmetros permanecem fixos;

Esses princípios apresentam algumas implicações interessantes. Primeiro, sobre o princípio 1, $IR > 0$ se $|A\&B| > \frac{|A||B|}{n}$ (indica que A está positivamente correlacionado a B) e $IR < 0$ se $|A\&B| < \frac{|A||B|}{n}$ (indica que A está negativamente correlacionado a B). A esse respeito, IR é similar ao coeficiente de correlação entre A e B. Segundo, pelo fato de se ter sempre $|A\&B| \leq |A| \leq |B|$ os princípios 2 e 3 implicam que IR tem um máximo local quando $|A\&B| = |A|$ ou $|A| = |B|$ e um máximo global quando $|A\&B| = |A| = |B|$.

Esses princípios podem ser estendidos para outros parâmetros da regra, por exemplo, complexidade da regra (IR diminui monotonicamente) ou tamanho do conjunto de exemplos (IR aumenta monotonicamente).

Além dos três princípios já citados, outros podem ser encontrados na literatura, por exemplo, um quarto princípio foi sugerido por (Major & Mangano, 1993) no qual IR aumenta monotonicamente com $|A|$ (cobertura da regra), dado um fator de confiança maior que o fator de confiança *default* (por exemplo, a probabilidade inicial das classes).

A seguir são apresentadas algumas medidas de interessabilidade propostas na literatura.

4.5 Medidas de Interessabilidade de Regras

Existem várias medidas de interessabilidade de regras já pesquisadas. A maior parte utiliza a abordagem subjetiva, no entanto, as medidas mostradas nesta seção utilizam apenas a abordagem objetiva, ou seja, podem ser computadas a partir dos dados relacionados a cada regra. Usualmente, quanto maior o valor apresentado por essas medidas, maior será a interessabilidade que a regra pode apresentar.

4.5.1 Medida PS

Uma medida introduzida por (Piatetsky-Shapiro, 1991) que satisfaz os três princípios básicos de interessabilidade é definida como

$$PS(R) = rc - \frac{r \cdot c}{n} \qquad (4.13)$$

A medida $PS(R)$ leva em conta a questão do desbalanceamento da distribuição das classes

favorecendo regras que predizem a classe minoritária. Uma distribuição de classes está desbalanceada se os exemplos de uma classe são muito mais frequentes, ou muito mais raros, que exemplos de outras classes.

Entretanto, a medida $PS(R)$ tem a propriedade indesejável de ser simétrica com relação ao antecedente e consequente da regra. Algumas modificações podem ser feitas na medida $PS(R)$ a fim de incorporar outros critérios, tais como interessabilidade dos atributos, custo de classificação incorreta e tamanho do disjunto. Esses três critérios propostos em (Freitas, 1998a) tem o efeito positivo de torná-la assimétrica e são apresentados a seguir.

4.5.2 Interessabilidade dos Atributos

Em domínios de aplicação onde o custo de atributos deve ser considerado, o termo $IntAtr(R)$ pode ser simplesmente definido como o inverso da soma dos custos de todos os atributos que ocorrem no antecedente da regra, isto é

$$IntAtr(R) = \frac{1}{\sum_{i=1}^{m} Custo(A_i)} \tag{4.14}$$

onde $Custo(A_i)$ é o custo do i-ésimo atributo que ocorre no antecedente da regra, e m é o número de atributos que ocorrem no antecedente da regra.

Para evitar que uma regra com muitas condições seja prejudicada, ou seja, que a regra atribua um baixo valor para $IntAtr(R)$ ao contrário do que acontece com uma regra que tem poucas condições, $IntAtr(R)$ pode ser simplesmente definido como o inverso da média aritmética dos custos de todos os atributos que ocorrem no antecedente da regra, ou seja

$$IntAtr(R) = \frac{1}{\sum_{i=1}^{m} \frac{Custo(A_i)}{m}} \tag{4.15}$$

Em domínios onde os custos de atributos são considerados irrelevantes, $IntAtr(R)$ pode ser definido da mesma forma que $SurpAtr(R)$ definido pela Equação 4.26 na Sessão 4.5.6 página 56, isto é

$$IntAtr(R) = \frac{1}{\sum_{i=1}^{m} \frac{GInfo(A_i)}{m}} \tag{4.16}$$

4.5.3 Custo de Classificação Incorreta e Tamanho do Disjunto

Em domínios onde os custos de classificação variam de acordo com a classe, a medida $PS(R)$ deve ser modificada. Uma maneira simples de fazer isso é multiplicar a Equação 4.13 por

um novo termo chamado $Custo_{CI}$, definido como

$$Custo_{CI}(R) = \frac{1}{\sum_{k=1}^{j} \Pr(C_k)\ Custo(C_i, C_k)} \tag{4.17}$$

onde

$\Pr(C_k)$ é a probabilidade de que a classe de um exemplo coberto pela regra seja C_k;
C_i é a classe predita pela regra;
$Custo(C_i, C_k)$ é o custo de classificar incorretamente um exemplo da classe C_k como sendo da classe C_i;
j é o número de classes.

Assumindo um problema de duas classes, C_1 e C_2, e considerando que a classe predita pela regra R é C_1, a Equação 4.17 resulta em

$$\begin{aligned} Custo_{CI}(R) &= \frac{1}{\Pr(C_1)\ Custo(C_1, C_1) + \Pr(C_2)\ Custo(C_1, C_2)} \\ &= \frac{1}{\Pr(C_2)\ Custo(C_1, C_2)} \end{aligned}$$

pois $Custo(C_1, C_1) = 0$. Uma estimativa natural para $\Pr(C_2)$, ou seja, a probabilidade de que a classe do exemplo coberto pela regra que prediz a classe C_1 seja C_2 é dado por

$$\Pr(C_2) = \frac{r\bar{c}}{r} \tag{4.18}$$

Quando a regra é um pequeno disjunto, isto é, cobre um número pequeno de exemplos, a estimativa da probabilidade pode ser melhorada usando a correção de Laplace.

$$\Pr(C_2) = \frac{1 + r\bar{c}}{2 + r} \tag{4.19}$$

Essa correção pode ser facilmente generalizada para um problema de j classes, substituindo na Equação 4.19 o 2 no denominador por j.

4.5.4 Extensão da Medida PS

Considerando os três critérios comentados anteriormente (interessabilidade dos atributos, tamanho do disjunto e custo de classificação incorreta) a medida $PS(R)$ de interessabilidade de regras definida na Seção 4.5.1, página 48 pode ser estendida da seguinte forma

$$PS_E(R) = \left(rc - \frac{r \cdot c}{n}\right) \cdot IntAtr(R) \cdot \left(\frac{1}{Custo_{CI}(R)}\right) \tag{4.20}$$

onde $IntAtr(R)$ pode ser calculado por uma das Equações 4.14, 4.15 ou 4.16, e $Custo_{CI}$(R) pode ser calculado pela Equação 4.17 e uma das Equações 4.18 ou 4.19.

Segundo (Freitas, 1998a), a escolha correta da equação a ser usada para os termos $IntAtr(R)$ e $Custo_{CI}(R)$ depende tanto do problema em questão quanto da interação desses termos com outros fatores de qualidade de regras, especialmente o tamanho do disjunto. Por exemplo, o uso da Equação 4.19 tem a vantagem de melhorar a confiabilidade de uma estimativa de probabilidade para pequenos disjuntos, enquanto que a Equação 4.18 não tem a mesma vantagem. Outro problema é que a Equação 4.13 é simétrica, ao passo que regras de classificação devem ser assimétricas, mas isso é resolvido adicionando os termos assimétricos $IntAtr(R)$ e $Custo_{CI}(R)$ nessa equação para obter a medida $PS_E(R)$.

4.5.5 Grau de Surpresa de Pequenos Disjuntos

Pequenos disjuntos são regras que cobrem um número pequeno de exemplos. Saber se uma regra é ou não um pequeno disjunto depende de algum critério específico relacionado com o número de exemplos cobertos pela regra (tal como um *threshold* ou outro critério mais flexível).

Em geral, os pequenos disjuntos são indesejáveis, porque apresentam pouca generalidade e são propensos a erro. Isso leva alguns algoritmos a utilizar um *bias* que favorece a descoberta de grandes disjuntos. No entanto, esses pequenos disjuntos tem o potencial para capturar conhecimento inesperado dos dados. (Freitas, 1998b) menciona o trabalho de (Provost & Aronis, 1996) que relata uma aplicação na qual um pequeno disjunto descoberto por um algoritmo revelou um conhecimento verdadeiramente interessante que levou a novas e importantes pesquisas no domínio onde foi aplicado.

A questão dos pequenos disjuntos tem sido tratada principalmente sob o ponto de vista da precisão de classificação. Sob o ponto de vista do grau de surpresa (ou mais genericamente interessabilidade) do conhecimento descoberto, (Freitas, 1998b) apresenta algumas sugestões, mostradas nesta seção, que podem ser utilizadas para uma medida do grau de surpresa de pequenos disjuntos.

A idéia geral dessa medida é que ela deve considerar um pequeno disjunto como conhecimento surpresa quando esse disjunto prediz uma classe diferente das classes preditas pelas suas generalizações mínimas. Um pequeno disjunto terá tantas generalizações quantas forem as suas condições, isto é, cada uma das i generalizações ($i = 1...m$) está associada com uma

das m condições $cond_i$ do disjunto original.

Usualmente, a maneira de se obter as generalizações depende do tipo dos atributos. Se o atributo At presente em $cond_i$ é discreto então a i-ésima generalização é obtida ao remover $cond_i$. Por outro lado, se At for contínuo a i-ésima generalização pode ser obtida de duas formas.

1. remover $cond_i$ do disjunto (semelhante ao caso de At ser discreto);
2. adicionar um pequeno valor α a Val em $cond_i$ quando op pertence a $\{<, \leq\}$, ou subtrair α de Val em $cond_i$ quando op pertence a $\{>, \geq\}$.

O valor de α deve ser especificado pelo usuário, no entanto, para casos onde seja necessário definir diferentes valores de α para vários atributos contínuos uma alternativa é utilizar um valor relativo de α (por exemplo um valor percentual) ou aplicar outros métodos (por exemplo uma abordagem baseada em percentil). A escolha do valor de α e da maneira de lidar com condições que tem atributos contínuos são dependentes do domínio.

Uma maneira de medir objetivamente o grau de surpresa de pequenos disjuntos que foram descobertos por um sistema de aprendizado por exemplos utilizando a remoção de $cond_i$ para obter as m geneneralizações pode ser definido da seguinte maneira. Seja a regra

$$\textbf{if } cond_1 \textbf{ and } cond_2 \textbf{ and } cond_3 \textbf{ and } \cdots \ cond_m \textbf{ then classe } C_k$$

onde $C_k \in \{C_1, C_2, \cdots C_j\}$ é a classe predita pelo pequeno disjunto. As possíveis generalizações são:

1 if $cond_2$ and $cond_3$ and $\cdots$ $cond_m$ then classe C_{k_1}
2 if $cond_1$ and $cond_3$ and $\cdots$ $cond_m$ then classe C_{k_2}
3 if $cond_1$ and $cond_2$ and $\cdots$ $cond_m$ then classe C_{k_3}
$\cdots$
m if $cond_1$ and $cond_2$ and $\cdots$ $cond_{m-1}$ then classe C_{k_m}

onde C_{k_i} é a classe predita pela i-ésima generalização da regra, ou seja, a classe $C_k \in \{C_1, C_2, \cdots C_j\}$ predita pelo pequeno disjunto é comparada com cada C_{k_i}, $i = 1, \ldots, m$, contando o número de vezes que C_{k_i} é diferente de C_k, isto é

$$SurpDisj(R) = \sum_{i=1}^{m} diferente(C_{k_i}, C_k) \tag{4.21}$$

onde

$$diferente = \begin{cases} 1 & \text{se } C_{k_i} \neq C_k \\ 0 & \text{caso contrário} \end{cases}$$

O resultado é um número inteiro no intervalo $0 \ldots m$, o qual define o grau de surpresa do pequeno disjunto. O valor de $SurpDisj(R)$ é influenciado pelo número de condições m no disjunto, portanto para evitar confusão entre medidas de complexidade sintática e medidas do grau de surpresa de disjuntos, pode ser definida a seguinte medida normalizada

$$SurpDisj_{norm}(R) = \frac{SurpDisj(R)}{m} \tag{4.22}$$

$SurpDisj_{norm}(R)$ assume valores no intervalo [0..1] e quanto maior esse valor mais surpresa o pequeno disjunto apresenta. No entanto, essa medida normalizada tem um *bias* que se direciona para regras com poucas condições, assim, provavelmente terá dificuldades para assegurar um alto valor normalizado para regras com muitas condições.

A fim de exemplificar o uso das mínimas generalizações para determinar o grau de surpresa de um pequeno disjunto será considerado o conjunto de exemplos da Tabela 4.4.

Exemplo	Atributos				Classe
	Cabeça	Corpo	Sorri?	Segura	
1	quadrada	quadrada	sim	balão	amigo
2	quadrada	triangular	não	espada	inimigo
3	redonda	redondo	sim	bandeira	amigo
4	triangular	redondo	sim	espada	inimigo
5	triangular	triangular	sim	balão	amigo
6	redonda	quadrada	não	bandeira	inimigo
7	triangular	quadrada	sim	bandeira	amigo
8	redonda	quadrada	sim	espada	inimigo
9	quadrada	redonda	sim	balão	amigo
10	redonda	triangular	não	bandeira	inimigo

Tabela 4.4: Exemplos de Robôs Amigos e Inimigos

Esse conjunto de exemplos, usualmente utilizado pela comunidade de Aprendizado de Máquina para ilustrar o processo de construção de novas *features* (Lee, 1999), está relacionado à observação de robôs amigos e inimigos com base nos seguintes atributos discretos:

Cabeça = {quadrada, redonda, triangular}
Corpo = {quadrada, redonda, triangular}

Sorri? = {sim, não}
Segura = {balão, espada, bandeira}

e a classe

classe = {amigo, inimigo}.

A seguinte regra R_1 foi extraída do conjunto de exemplos da Tabela 4.4.

if Sorri? = sim and Segura = espada then classe = inimigo [0 2]

Nessa regra, os valores que estão entre os sinais '[' e ']' indicam o número de exemplos cobertos pelo pequeno disjunto para as classes "amigo" e "inimigo" respectivamente.

As possíveis generalizações de R_1 são:

1 if Segura = espada then classe = inimigo [0 3]
2 if Sorri? = sim then classe = amigo [5 2]

A forma de determinar a classe predita por cada generalização deve ser a mesma utilizada pelo algoritmo que gerou o pequeno disjunto. Tipicamente assume-se a classe com maior frequência relativa entre os exemplos cobertos pela generalização. Dessa forma, na primeira generalização a classe predita é "inimigo" e na segunda a classe predita é "amigo".

Contando o número de classes preditas pelas duas generalizações mínimas que diferem da classe predita pelo pequeno disjunto o resultado é 1, ou seja $DisjSurp(R) = 1$. Esse resultado é aplicado a Equação 4.22 para obter o grau de surpresa normalizado.

$$\begin{aligned} SurpDisj_{norm}(R) &= \frac{1}{2} \\ &= 0.5 \end{aligned}$$

4.5.6 Grau de Surpresa dos Atributos Individuais de Regras

Várias das medidas do grau de surpresa (ou mais genericamente, medidas de interessabilidade) usualmente consideram o antecedente de uma regra como um todo, sem levar em conta individualmente os atributos que ocorrem no antecedente da regra. Algumas idéias baseadas na teoria da informação, considerando atributos com valores discretos, foram sugeridas no

excelente trabalho de (Freitas, 1998b) para serem utilizadas em uma medida do grau de surpresa de regras que considera individualmente os atributos que ocorrem no antecedente da regra.

Primeiramente é calculado o ganho de informação definido como $GInfo(At_p)$ de cada atributo preditivo At_p presente no antecedente da regra.

$$GInfo(At_p) = Info(C) - Info(C|At_p) \tag{4.23}$$

com

$$Info(C) = -\sum_{i=1}^{j} \Pr(C_i) \log \Pr(C_i) \tag{4.24}$$

e

$$Info(C|At_p) = \sum_{q=1}^{s} \Pr(At_{pq})(-\sum_{i=1}^{j} \Pr(C_i|At_{pq}) \log \Pr(C_i|At_{pq})) \tag{4.25}$$

onde:

$Info(C)$ é a informação da classe;
$Info(C|At_p)$ é a informação do atributo classe C dado o atributo At_p;
At_{pq} denota o q-ésimo valor do atributo At_p;
C_i denota o i-ésimo valor da classe C;
j é o número de classes;
s representa o número de valores possíveis do atributo At_p;
todos os log estão na base 2.

Atributos com alto ganho de informação geralmente são bons preditores de classe quando considerados individualmente. No entanto, do ponto de vista da interessabilidade de regras, existe a possibilidade que o usuário já conheceça quais os atributos que individualmente são melhores preditores e, assim, as regras contendo esses atributos tendem a ter baixo grau de surpresa para esse usuário. De outro lado estão os atributos com baixo ganho de informação os quais muitas vezes, são vistos como sem interesse pelo usuário e irrelevantes para classificação quando considerados individualmente.

A interação dos atributos pode tornar um atributo individualmente irrelevante em um atributo relevante, e esse fenômeno está intuitivamente associado com o grau de surpresa de uma regra. Considerando que outros fatores (tais como precisão da predição, cobertura e completeza) permaneçam iguais, pode se argumentar que uma regra cujo antecedente contém somente atributos com baixo ganho de informação pode ser apresentada como mais surpreendente que regras cujo antecedente contém atributos com alto ganho de informação. Essa

idéia pode ser expressa matematicamente ao definir $SurpAtr(R)$ como:

$$SurpAtr(R) = \frac{1}{\sum_{p=1}^{m} \frac{GInfo(At_p)}{m}} \tag{4.26}$$

A fim de exemplificar o uso do ganho de informação de atributos para determinar o grau de surpresa de uma regra será considerado a mesma regra R_1 da subseção anterior, página 54, relacionada com os exemplos na Tabela 4.4.

O ganho de informação do atributo "Cabeça" da Tabela 4.4 é obtido pela Equação 4.23

$$\begin{aligned} GInfo(\text{Cabeca}) &= Info(C) - Info(C|Cabeca) \\ &= 1 - 0.874 \\ &= 0.126 \end{aligned}$$

onde

$$\begin{aligned} Info(C) &= -\sum_{i=1}^{j=2} \Pr(C_i) \log \Pr(C_i) \\ &= -(-0.5 + (-0.5)) \\ &= 1 \end{aligned}$$

e

$$\begin{aligned} Info(C|\text{Cabeca}) &= \sum_{q=1}^{s=3} \Pr(At_{pq})(-\sum_{i=1}^{j=2} \Pr(C_i|At_{pq}) \log \Pr(C_i|At_{pq})) \\ &= 0.3(-(-0.918)) + 0.4(-(-0.811)) + 0.3(-(-0.918)) \\ &= 0.275 + 0.324 + 0.275 \\ &= 0.874 \end{aligned}$$

O ganho de informação dos outros atributos ("Corpo", "Sorri?" e "Segura") são obtidos da mesma forma e seus resultados são, respectivamente:

$$\begin{aligned} GInfo(\text{Corpo}) &= 0.951 \\ GInfo(\text{Sorri?}) &= 0.396 \\ GInfo(\text{Segura}) &= 0.600 \end{aligned}$$

Para calcular o grau de surpresa da regra R_1 considerando seus atributos "Sorri?" e "Segura"

utiliza-se a Equação 4.26 que resulta em

$$\begin{aligned} SurpAtr(R_1) &= \frac{1}{\frac{0.396}{2} + \frac{0.600}{2}} \\ &= \frac{1}{0.498} \\ &= 2.008 \end{aligned}$$

4.6 Considerações Finais

A utilização de medidas de qualidade e interessabilidade são úteis como mecanismo de avaliação de um conjunto de regras, principalmente com o objetivo de indentificar as regras que se destacam por ter boa qualidade e/ou por serem interessantes para o usuário. Deve se notar que uma regra com boa qualidade não é necessariamente interessante e por outro lado algumas vezes uma regra com baixa qualidade pode revelar um conhecimento muito interessante para o usuário.

As informações fornecidas por essas medidas podem auxiliar no desenvolvimento de novas e melhores medidas e, portanto, em mecanismos de pós análise de regras mais eficientes. Também podem auxiliar, eventualmente, em alguma fase do processo indutivo dos sistemas de aprendizado.

As medidas de interessabilidade apresentadas neste capítulo são objetivas e têm a vantagem de serem genéricas (independentes do domínio de aplicação). As medidas objetivas podem ser utilizadas junto com as medidas subjetivas. Por exemplo, numa estratégia de refinamento de um conjunto de regras, as medidas objetivas podem funcionar como um primeiro filtro para selecionar regras potencialmente interessantes enquanto que as medidas subjetivas podem ser usadas como um filtro final para selecionar as regras verdadeiramente interessantes para o usuário.

Deve ser observado que não é possível afirmar que uma determinada medida, tanto de qualidade quanto de interessabilidade, é melhor para qualquer domínio de aplicação. Cada pesquisador ou usuário deve estudar as medidas para, se necessário, adaptar ou até mesmo criar uma medida apropriada ao seu domínio particular de aplicação.

Um problema que permanece é em relação a distribuição dos dados. Por exemplo, no caso do número de exemplos que definem cada classe não estar uniformemente distribuído — exemplos desbalanceados (Batista et al., 1999; Batista & Monard, 1998). Neste caso, os

valores de rc e $\overline{r}c$ tendem a ser muito pequenos, pois existem poucos exemplos a serem cobertos, enquanto que os valores de $\overline{rc}$ e $r\overline{c}$ tendem a ser muito grandes, resultando em medidas não significativas.

Com base nos estudos apresentados neste capítulo e nos anteriores, foi por nós proposto um sistema computacional que fornece várias informações com o objetivo de auxiliar o processo de avaliação do conhecimento adquirido por sistemas de aprendizado utilizando exemplos, quando este conhecimento está representado por regras. Esse sistema, chamado $\mathcal{RQ}_{system}$ — *Rule Quality-system* — está descrito no capítulo seguinte.

Capítulo 5

Um Sistema para Auxiliar na Avaliação de Regras — $\mathcal{RQ}_{system}$

5.1 Considerações Iniciais

O conhecimento adquirido por sistemas de aprendizado pode ser avaliado pelo usuário de diferentes maneiras. Quando esse conhecimento está representado por regras, uma maneira mais simples é a própria interpretação da regra, isto é, analisar as informações nela contida. Isso pode até ser suficiente numa análise superficial, no entanto, há muitas outras informações relacionadas com as regras que podem ser obtidas utilizando-se por exemplo estatística e teoria da informação, conforme discutido no capítulo anterior.

As informações relacionadas com as regras permitem ao usuário avaliá-las mais criteriosamente, por exemplo, identificando aquelas que apresentam maior qualidade e sejam mais interessantes. Analisar as muitas informações sobre as regras, manualmente, pode ser impraticável devido, principalmente, a quantidade de dados associados a elas e também aos cálculos necessários para retornar tais informações.

Com o objetivo de auxiliar na avaliação de regras é por nós proposto um sistema computacional, descrito a seguir, que oferece facilidades para obter diversas informações relacionadas com a avaliação de regras.

5.2 Visão Geral do $\mathcal{RQ}_{system}$

O sistema está implementado na linguagem de programação lógica Prolog (Bratko, 1990; Clocksin & Mellish, 1994; Flach, 1994; O'Keefe, 1990; Ross, 1989; Rowe, 1988; Sterling & Shapiro, 1986), mais especificamente em LPA-Prolog (Westwood, 1997b; Westwood, 1997a), a qual oferece oportunidades de solucionar alguns dos problemas fundamentais de Engenharia de Software. Assim, ao explorar de maneira apropriada certas características do Prolog, é possível melhorar a clareza, robustez e confiabilidade de programas, bem como melhorar a comunicação entre programadores. A idéia principal é aproveitar a vantagem da concisão do Prolog para escrever programas legíveis. Todavia, como programas auto-documentáveis são um mito, é importante ter uma explicacão tão clara e concisa quanto possível da atual implementação. A implementação do $\mathcal{RQ}_{system}$ é aberta de forma a permitir que novas características possam ser incorporadas futuramente. Detalhes sobre a implementação desse sistema podem ser consultados nos Relatórios Técnicos (Monard & Horst, 1999b; Monard & Horst, 1999a).

O $\mathcal{RQ}_{system}$ consiste de dois módulos principais que executam tarefas bem definidas. O diagrama de contexto da Figura 5.1 proporciona uma visão geral do sistema. Pode ser observado que o primeiro módulo — Módulo 1 — desempenha a parte de pré-processamento dos dados de entrada e depois armazena tais dados em uma Base de Fatos, enquanto que o segundo módulo — Módulo 2 — processa os dados contidos na Base de Fatos para gerar uma série de informações que serão fornecidas para o usuário.

O método adotado para a modelagem e descrição do $\mathcal{RQ}_{system}$ está baseado na análise estruturada, a qual é amplamente utilizada pela comunidade de Engenharia de Software. Um dos recursos desse método é a utilização de DFDs — Diagramas de Fluxo de Dados — os quais permitem representar um sistema, ou *software*, de maneira simples e em qualquer nível de abstração (Pressman, 1997). Nas próximas seções são explicados cada um desses módulos.

5.3 Descrição dos Dados de Entrada

Os dados de entrada consistem dos seguintes arquivos

1. arquivo de *features*;

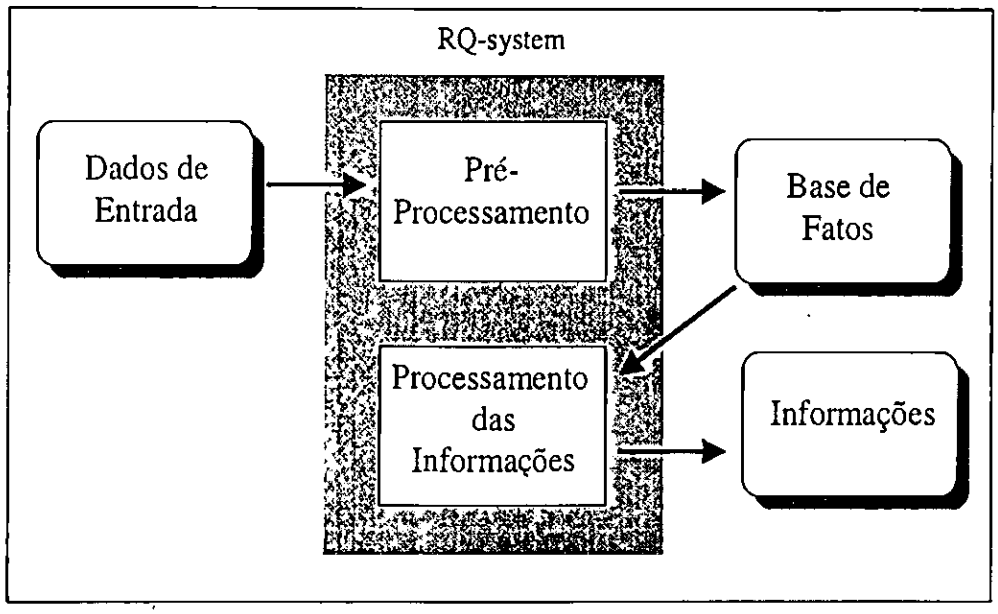

Figura 5.1: Visão Geral do $\mathcal{RQ}_{system}$

2. arquivo de exemplos;

3. arquivo de regras.

A sintaxe desses arquivos é descrita em detalhes nas próximas seções.

A nossa decisão de considerar esses três arquivos separadamente como dados de entrada para o $\mathcal{RQ}_{system}$ deve-se ao fato da maioria dos algoritmos de aprendizado simbólico possuir opções para utilizar como entrada um arquivo no qual são definidas as *features* e outro arquivo com os exemplos de treinamento, produzindo como saída um arquivo com as regras geradas, como ilustra a Figura 5.2

Deve ser observado que o arquivo de exemplos do $\mathcal{RQ}_{system}$ pode conter o mesmo conjunto de treinamento utilizado para gerar as regras no arquivo de regras, bm como outro conjunto de teste independente do utilizado para gerar essas regras.

A sintaxe dos dois primeiros arquivos é independente do algoritmo de aprendizado utilizado para gerar as regras de conhecimento, enquanto que a sintaxe do arquivo de regras depende do algoritmo utilizado para gerá-las. O $\mathcal{RQ}_{system}$ reconhece regras geradas pelos algoritmos de Aprendizado de Máquina $\mathcal{CN}2$ e $\mathcal{C}4.5$, mas pode ser facilmente estendido para reconhecer regras geradas por outros algoritmos proposicionais. Ambos os algoritmos trabalham com valores desconhecidos (indicados pelo símbolo ?), os quais são substituídos por um valor que

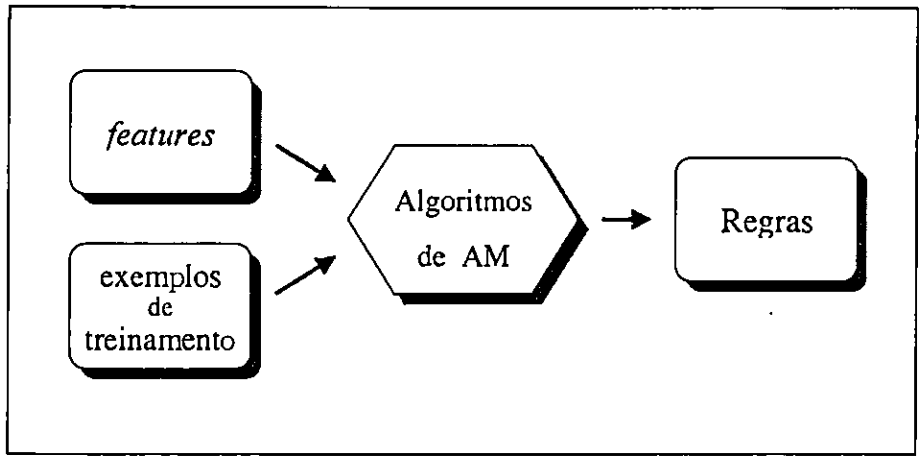

Figura 5.2: Esquema Geral de E/S de Algoritmos Simbólicos de Aprendizado de Máquina

depende dos valores do atributo, *i.e.* se são contínuos ou discretos.

$\mathcal{CN}2$ (Clark & Niblett, 1989) é um algoritmo de aprendizado que induz regras de classificação do tipo **if** <complex> **then** <class> em domínios onde podem existir ruídos. Cada <complex> é uma disjunção de conjuncões.

$\mathcal{C}4.5$ (Quinlan, 1993) pertence a uma família de algoritmos denominada *Top Down Induction by Decision Trees* — TDIDT. $\mathcal{C}4.5$ cria árvores para representar regras de classificação. Um nó em uma árvore de decisão representa um teste sobre um atributo particular. A árvore é construída utilizando o conjunto de exemplos de treinamento, através de um procedimento que escolhe inicialmente um atributo, de acordo com seus valores, para dividir esse conjunto. Outras divisões são feitas nos subconjuntos escolhendo-se outros atributos, até que cada subconjunto contenha apenas exemplos da mesma classe. Quando isso acontece, um nó folha é criado e rotulado com o mesmo nome da respectiva classe. O $\mathcal{C}4.5rules$, utilizado neste trabalho, após realizar uma poda na árvore de decisão, escreve a árvore resultante na notação de regras de decisão.

O conjunto de exemplos de treinamento no formato atributo-valor, extraído de (Quinlan, 1993) e listado na Tabela 5.1, será usado para ilustrar os três arquivos que constituem os dados de entrada do $\mathcal{RQ}_{system}$.

Outlook	Temp_F	Humidity_%	Windy	Class
sunny	75	70	true	Play
sunny	80	90	true	Don't_Play
sunny	85	85	false	Don't_Play
sunny	72	95	false	Don't_Play
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't_Play
rain	65	70	true	Don't_Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Tabela 5.1: Conjunto de Exemplos de Treinamento no Formato Atributo-Valor

Cada um desses exemplos está relacionado a uma observação que determina se deve-se jogar (Play) ou não jogar (Don't_Play) tênis levando em conta algumas características do tempo.

5.3.1 Arquivo de *Features*

Neste arquivo encontram-se definidos o atributo classe e todos os atributos (*features*) do conjunto de exemplos considerado. Deve ser observado que a ordem dessas definições é importante. Primeiramente, devem ser definidos os possíveis valores do atributo classe para depois serem definidos cada um dos atributos. Por exemplo, o primeiro atributo definido será considerado como aquele definido pela primeira coluna da Tabela 5.1, o segundo atributo pela segunda coluna e assim por diante.

Assim, a primeira linha do arquivo de *features* define os nomes das classes, sendo que os nomes são separados por vírgulas e a linha termina com um ponto. O resto do arquivo contém uma linha para cada *feature*. Cada linha começa com o nome da *feature* seguido por dois pontos e depois pela palavra **continuous**, indicando uma *feature* com valor real, ou por uma lista com todos os possíveis valores discretos que a *feature* pode assumir, separados por vírgulas.

A gramática, na notação BNF (Backus-Naur Form), que define a sintaxe do arquivo de *features* está descrita na Tabela 5.2. Deve ser observado que a sintaxe por nós definida é

muito semelhante a utilizada por vários algoritmos de aprendizado para definir as *features*.

<class_feature> ::= <class_name> { , <class_name>} .
<class_name> ::= *string* de caracteres não incluindo espaços em branco
<feature> ::= <feature_name> : <type>
<feature_name> ::= *string* de caracteres não incluindo espaços em branco
<type> ::= **continuous** | <listof_values>
<listof_values> :== <value> {,<value>}
<value> ::= *string* de caracteres não incluindo espaços em branco

Tabela 5.2: Sintaxe do Arquivo de *Features*

Na Tabela 5.3 é mostrado o arquivo de *features* `play-dontplay.names`, relacionado com o conjunto de exemplos da Tabela 5.1, na sintaxe reconhecida pelo $\mathcal{RQ}_{system}$.

```
Play, Don't_Play.

Outlook: sunny, overcast, rain.
Temp_F: continuous.
Humidity_%: continuous.
Windy: true, false.
```

Tabela 5.3: Exemplo de Arquivo de *Features* `play-dontplay.names`

5.3.2 Arquivo de Exemplos

Esse arquivo tem o mesmo formato que o arquivo de exemplos da $\mathcal{MLC}$++ onde cada linha representa um exemplo e os valores das *features* estão separados por vírgulas. O último valor corresponde ao valor da classe do exemplo. A gramática que define a sintaxe do arquivo de exemplos está descrita na Tabela 5.4.

<example> ::= <value> { , <value>}
<value> ::= <integer_number> | <real_number> | <string>
<string>::= *string* de caracteres não incluindo espaços em branco

Tabela 5.4: Sintaxe do Arquivo de Exemplos

O arquivo de exemplos `play-dontplay.data`, relacionado com o conjunto de treinamento da Tabela 5.1 é mostrado na Tabela 5.5.

sunny, 75, 70, true, Play
sunny, 80, 90, true, Don't_Play
sunny, 85, 85, false, Don't_Play
sunny, 72, 95, false, Don't_Play
sunny, 69, 70, false, Play
overcast, 72, 90, true, Play
overcast, 83, 78, false, Play
overcast, 64, 65, true, Play
overcast, 81, 75, false, Play
rain, 71, 80, true, Don't_Play
rain, 65, 70, true, Don't_Play
rain, 75, 80, false, Play
rain, 68, 80, false, Play
rain, 70, 96, false, Play

Tabela 5.5: Exemplo de Arquivo de Exemplos `play-dontplay.data`

5.3.3 Arquivo de Regras

Como mencionado anteriormente, o arquivo de regras é dependente do algoritmo. Na versão atual, o $\mathcal{RQ}_{system}$ reconhece as regras geradas pelos algoritmos $\mathcal{CN}2$ e $\mathcal{C}4.5$. A gramática que define a sintaxe do arquivo de regras, para ambos algoritmos, é descrita a seguir.

$\mathcal{CN}2$

A gramática que define a sintaxe do arquivo com regras não ordenadas do $\mathcal{CN}2$ está descrita na Tabela 5.6. Deve ser observado que a última regra desse arquivo, tanto para o $\mathcal{CN}2$ quanto para o $\mathcal{C}4.5$ deve ser a regra **DEFAULT**.

<rule> ::= **IF** <condition> <rest_conditions> <condition_class>
<listof_numbers>
<rule> ::= **(DEFAULT)** <condition_class> <listof_numbers>
<condition> ::= <attribute> <operator> <ident> |
<ident> <operator> <attribute> <operator> <ident>
<rest_conditions> ::= <condition> <rest_conditions> | **THEN**
<condition_class> ::= **class** = <ident>
<listof_numbers> ::= lista de números na sintaxe Prolog
<operator> ::= = | < | > | <= | >= | =< | =>
<ident> ::= <attribute> | <integer_number> | <real_number>
<attribute> ::= true atom na sintaxe Prolog diferente de class

Tabela 5.6: Sintaxe do Arquivo de Regras do $\mathcal{CN}2$

O arquivo de regras não ordenadas do $C\mathcal{N}2$ `play-dontplay.cn2.rules`, o qual foi induzido utilizando o conjunto de treinamento da Tabela 5.1, é mostrado na Tabela 5.7.

```
IF Humidity_% < 82.50
AND Windy = false
THEN class = Play [5 0]

IF Outlook = overcast
THEN class = Play [4 0]

IF Humidity_% > 95.50
THEN class = Play [1 0]

IF Outlook = sunny
AND Humidity_% < 77.50
THEN class = Play [2 0]

IF Outlook = sunny
AND Humidity_% > 77.50
THEN class = Don't_Play [0 3]

IF Outlook = rain
AND Windy = true
THEN class = Don't_Play [0 2]

(DEFAULT) class = Play [9 5]
```

Tabela 5.7: Exemplo de Arquivo de Regras do $C\mathcal{N}2$ `play-dontplay.cn2.rules`

Essas regras são não ordenadas no sentido de cada uma delas ser válida independentemente da ordem em que elas aparecem. A lista de números no fim de cada regra informa o número de exemplos de treinamento de cada classe, cobertos pela regra. Dessa forma, a ordem em que foram definidas as classes no arquivo de *features* determina a ordem de apresentação dos números dessa lista, assim como o número de elementos nessa lista é determinado pelo número de classes. No caso específico das regras da Tabela 5.7, o primeiro número refere-se a classe Play e o segundo a classe Don't_Play. Portanto, a lista [5 0] na primeira regra indica que ela cobre 5 exemplos de treinamnto da classe Play e nenhum exemplo da classe Don't_Play.

Pode também ser observado que o número total de exemplos cobertos pelas regras que definem a classe Play é 12 (5+4+1+2), enquanto que para a classe Don't_Play esse número

é 5 (3+2). Portanto, pode-se afirmar que um mesmo exemplo de treinamento é coberto por mais de uma das regras que definem a classe Play, pois o número total de exemplos de treinamento rotulados com a classe Play é 9 — Tabela 5.1, pg. 63. Para a classe Don't_Play, ainda que o número total de exemplos cobertos pelas duas regras é o mesmo que o total de exemplos de treinamento rotulados com essa classe, 5 em ambos os casos, nada pode ser afirmado pois existe a possibilidade de um mesmo exemplo ter sido coberto por ambas as regras e um outro exemplo dessa classe ter ficado sem cobertura. A identificação precisa dos exemplos cobertos por cada regra, bem como dos exemplos cobertos por mais de uma regra, pode ser realizada utilizando as facilidades implementadas no $\mathcal{RQ}_{system}$.

Deve ser observado que o arquivo de exemplos, aqui utilizado para ilustrar os dados de entrada do $\mathcal{RQ}_{system}$, é o mesmo que o utilizado para gerar o arquivo de regras. Neste caso, a informação contida na lista de números no fim de cada regra é redundante já que poderia ser facilmente calculada pelo $\mathcal{RQ}_{system}$. A nossa decisão de guardar todas as informações fornecidas pelo algoritmo $\mathcal{CN}2$, bem como as fornecidas pelo $\mathcal{C}4.5$ leva em consideração a possibilidade de poder utilizar outros conjuntos de exemplos do domínio. Neste último caso, a lista de números calculada pelo $\mathcal{RQ}_{system}$ não será necessariamente, idêntica a encontrada pelo algoritmo na fase de treinamento. Uma análise cuidadosa de ambos os resultados pode fornecer informações muito importantes relacionadas ao comportamento esperado das regras sobre exemplos nunca visto pelo algoritmo.

$\mathcal{C}4.5$

A gramática que define a sintaxe do arquivo de regras do $\mathcal{C}4.5$ está descrita na Tabela 5.8.

<rule> ::= **Rule** <integer_number> : <condition> <rest_condition>
<condition_class> <one_element_list>
<rule> ::= **Default** <default_condition_class>
<condition> ::= <attribute> <operator> <ident> | <attribute> **in** <setof_values>
<rest_condition> ::= <condition> <rest_condition> | ->
<condition_class> ::= **class** <ident>
<one_element_list> ::= [<real_number> %]
<default_condition_class> ::= **class** : <ident>
<attribute> ::= true atom diferente de **class**
<operator> ::= = | > | <=
<ident> ::= <attribute> | <integer_number> | <real_number>
<setof_values> ::= { <ident> <rest_setof_values>
<rest_setof_values> ::= <setof_values> | }

Tabela 5.8: Sintaxe do Arquivo de Regras do $\mathcal{C}4.5$

O arquivo de regras do $C4.5$ `play-dontplay.c45.rules`, o qual foi induzido do conjunto de treinamento da Tabela 5.1, é mostrado na Tabela 5.9.

```
Rule 2:
Outlook = overcast
-> class Play [70.7%]

Rule 4:
Outlook = rain
Windy = false
-> class Play [63.0%]

Rule 1:
Outlook = sunny
Humidity_% > 75
-> class Don't_Play [63.0%]

Rule 3:
Outlook = rain
Windy = true
-> class Don't_Play [50.0%]

Default class: Play
```

Tabela 5.9: Exemplo de Arquivo de Regras do $C4.5$ `play-dontplay.c45.rules`

Deve ser observado que essas regras tem uma certa ordem. Por exemplo, se um novo exemplo deve ser classificado, inicialmente devem ser verificadas a regras que aparecem primeiro definindo uma das classes, a classe Play no exemplo considerado. Somente no caso do exemplo não ser classificado por essas regras, é que o conjunto de regras da outra classe pode ser ativado. A lista com um número percentual no fim de cada regra informa a precisão de classificacão correta esperada para essa regra quando forem apresentados exemplos ainda não vistos.

Esses três arquivos, os quais constituem os dados de entrada do $\mathcal{RQ}_{system}$, são pré-processados pelo Módulo 1 descrito a seguir.

5.4 Módulo 1 - Pré-processamento dos Dados de Entrada

O Módulo 1 é responsável pelo pré-processamento dos dados de entrada do sistema, isto é, dos dados provenientes dos arquivos de entrada — arquivo de *features*, arquivo de exemplos e arquivo de regras — descritos na Seção 5.3. Existem processos específicos para ler, formatar, armazenar na memória e fazer a verificação desses dados. Os principais processos do Módulo 1 do $\mathcal{RQ}_{system}$, são descritos a seguir, em alto nível, conforme a representação do DFD na Figura 5.3.

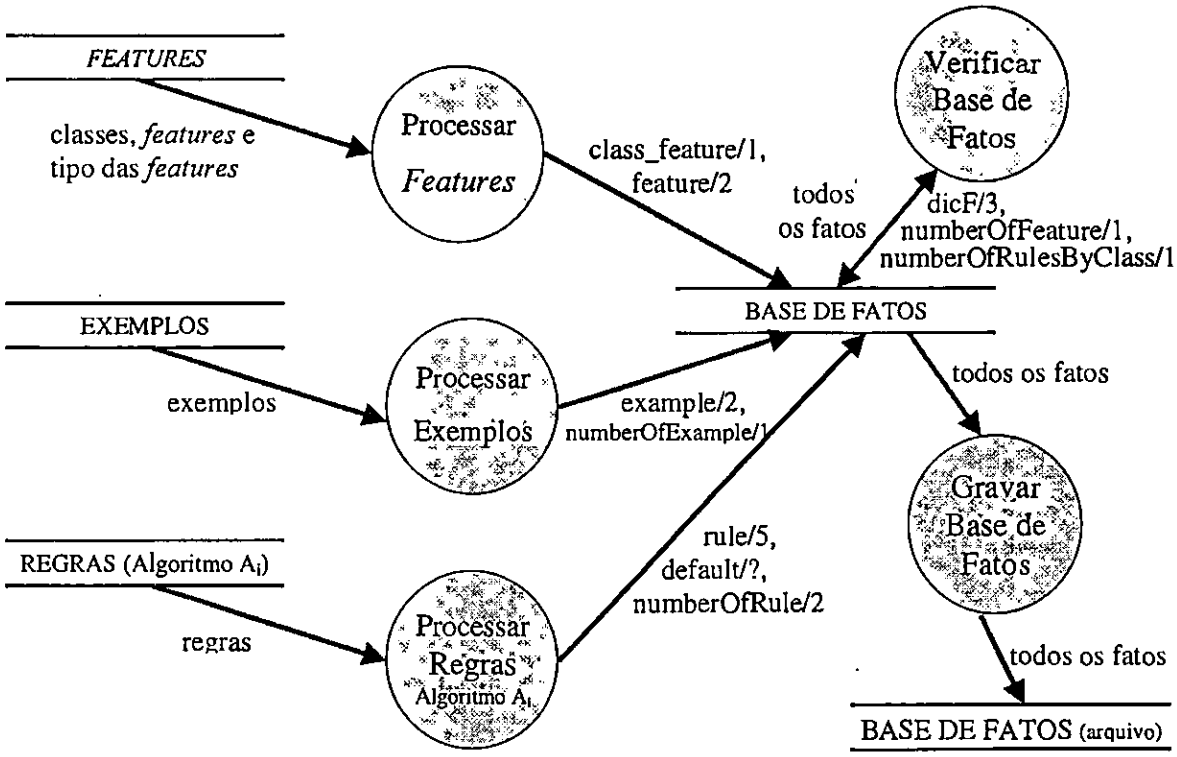

Figura 5.3: DFD do Módulo 1 do $\mathcal{RQ}_{system}$

5.4.1 Processar *Features*

O objetivo desse processo é ler e processar a informação contida no arquivo de *features*, a qual deve obedecer a sintaxe definida na Tabela 5.2, pg. 64, para o processo ser bem sucedido. Essa informação é lida sequencialmente e transformada para asserções (ou fatos) Prolog, as

quais são armazenadas na memória. As asserções são identificadas pelo seu nome e aridade, segundo a sintaxe descrita na Tabela 5.10,

class_feature/1
class_feature(<listof_names>).
<listof_names> ::= lista de true atoms na sintaxe Prolog
feature/2
feature(<feature_name>,<type>).
<feature_name> ::= true atom
<type> ::= **continuous** | <listof_values>
<listof_values> ::= lista de true atoms na sintaxe Prolog

Tabela 5.10: Sintaxe dos Fatos Prolog Gerados após Pré-processamento do Arquivo de *Features*

O fato `class_feature/1` é único, e refere-se a classe enquanto que o fato `feature/2` refere-se as *features*, sendo um fato para cada *feature*.

Por exemplo, após processar o arquivo de *features* `play-dontplay.names` — Tabela 5.3, pg. 64 — serão gravadas na memória as assercões mostradas na Tabela 5.11

```
class_feature(['Play','Don't_Play']).

feature('Outlook',[sunny,overcast,rain]).
feature('Temp_F',continuous).
feature('Humidity_%',continuous).
feature('Windy',[true,false]).
```

Tabela 5.11: Fatos Prolog Gerados após Pré-processamento do Arquivo de *Features*

5.4.2 Processar Exemplos

O objetivo desse processo é ler e processar a informação contida no arquivo de exemplos, a qual deve obedecer a sintaxe definida na Tabela 5.4, pg. 64, para o processo ser bem sucedido. Essa informação é lida sequencialmente e cada um dos exemplos é transformado em uma asserção Prolog `example/2`, segundo a sintaxe descrita na Tabela 5.12. Após armazenar na memória todos os exemplos, também é armazenada uma assercão `numberOfExample/1`, que informa o número total de exemplos processados.

example/2
example(<numberof_example>,<listof_values>).
<numberof_example> ::= número inteiro
<listof_values> ::= lista de true atoms na sintaxe Prolog
numberOfExample/1
numberOfExample(<totalof_example>).
<totalof_examples> ::= número inteiro

Tabela 5.12: Sintaxe dos Fatos Prolog Gerados após Pré-processamento do Arquivo de Exemplos

Por exemplo, após processar o arquivo de exemplos `play-dontplay.data` — Tabela 5.5, pg. 65 — cada exemplo será gravado na memória como mostra a Tabela 5.13.

```
example(1,[sunny,75,70,true,'Play']).
example(2,[sunny,80,90,true,'Don't_Play']).
example(3,[sunny,85,85,false,'Don't_Play']).
example(4,[sunny,72,95,false,'Don't_Play']).
example(5,[sunny,69,70,false,'Play']).
example(6,[overcast,72,90,true,'Play']).
example(7,[overcast,83,78,false,'Play']).
example(8,[overcast,64,65,true,'Play']).
example(9,[overcast,81,75,false,'Play']).
example(10,[rain,71,80,true,'Don't_Play']).
example(11,[rain,65,70,true,'Don't\_Play']).
example(12,[rain,75,80,false,'Play']).
example(13,[rain,68,80,false,'Play']).
example(14,[rain,70,96,false,'Play']).

numberOfExample(14).
```

Tabela 5.13: Fatos Prolog Gerados após Pré-processamento do Arquivo de Exemplos

5.4.3 Processar Regras

O objetivo desse processo é ler e processar a informação contida no arquivo de regras, a qual deve obedecer a sintaxe do algoritmo A_i utilizado para gerar essas regras, *i.e.* o processo é dependente do algoritmo de aprendizado. Na versão atual, o $\mathcal{RQ}_{system}$ reconhece a sintaxe para $A_i \in \{\mathcal{CN}2, C4.5\}$ — Tabelas 5.6 e 5.8. Essa informação é lida sequencialmente e

cada uma das regras é tranformada em uma asserção Prolog segundo a sintaxe descrita na Tabela 5.14 no caso de $A_i = \mathcal{CN}2$, e na Tabela 5.15 no caso de $A_i = \mathcal{C}4.5$.

rule/5
rule(<numberof_rule>,<listof_conditions>,<class>,<listof_numbers>,<algorithm_name>).
<numberof_rule> ::= número inteiro
<listof_condition> lista de estruturas Prolog da forma <condition>(<feature_name>,<value>)
<class> ::= estrutura Prolog da forma <class>(<class_name>)
<listof_number> ::= lista de números na sintaxe Prolog
<algorithm_name> ::= true atom
default/3
default(<class>,<listof_number>,<algorithm_name>).
<class> ::= estrutura Prolog da forma <class>(<class_name>)
<listof_number> ::= lista de números na sintaxe Prolog
<algorithm_name> ::= true atom

Tabela 5.14: Sintaxe dos Fatos Prolog Gerados após Pré-processamento do Arquivo de Regras do $\mathcal{CN}2$

rule/5
rule(<numberof_rule>,<listof_conditions>,<class>, <one_element_list>,<algorithm_name>).
<numberof_rule> ::= número inteiro
<listof_condition> lista de estruturas Prolog da forma <condition>(<feature_name>,<value>)
<class> ::= estrutura Prolog da forma <class>(<class_name>)
<one_element_list> ::= lista com um número na sintax Prolog
<algorithm_name> ::= true atom
default/2
default(<class>,<algorithm_name>).
<class> ::= estrutura Prolog da forma <class>(<class_name>)
<algorithm_name> ::= true atom

Tabela 5.15: Sintaxe dos Fatos Prolog Gerados após Pré-processamento do Arquivo de Regras do $\mathcal{C}4.5$

Em ambos os casos, após processar todas as regras, também é armazenada na memória uma asserção `numberOfRule/2`, que informa o número total de regras processadas, cuja sintaxe está definida na Tabela 5.16,

numberOfRule/2
numberOfRule(<totalof_rule>,<algorithm_name>).
<totalof_rule> ::= número inteiro
<algorithm_name> ::= true atom

Tabela 5.16: Sintaxe do Fato Prolog `numberOfRule/2`

Por exemplo, após processar o arquivo de regras `play-dontplay.cn2.rules` geradas por $\mathcal{CN}2$ — Tabela 5.7, pg. 66 — cada regra será armazenada na memória como mostra a Tabela 5.17.

```
rule(1,[less('Humidity_%',82.50),equal('Windy',false)],class('Play'),[5,0],cn2).
rule(2,[equal('Outlook',overcast)],class('Play'),[4,0],cn2).
rule(3,[grater('Humidity_%',95.50)],class('Play'),[1,0],cn2).
rule(4,[equal('Outlook',sunny),less('Humidity_%',77.50)],class('Play'),[2,0],cn2).
rule(5,[equal('Outlook',sunny),greater('Humidity_%',77.50)],class('Don't_Play'),
                                                              [0,3],cn2).
rule(6,[equal('Outlook',rain),equal('Windy',true)],class('Don't_Play'),[0,2],cn2).

default(class('Play'),[9,5],cn2).
```

Tabela 5.17: Fatos Prolog Gerados após Pré-processamento do Arquivo de Regras do $\mathcal{CN}2$

A Tabela 5.18 mostra cada regra armazenada na memória para o caso das regras geradas por $\mathcal{C}4.5$, após o processamento do arquivo de regras `play-dontplay.c45.rules` — Tabela 5.9, pg. 68.

```
rule(1,[equal('Outlook',overcast)],class('Play'),[70.7],c45).
rule(2,[equal('Outlook',rain),equal('Windy',false)],class('Play'),[63.0],c45).
rule(3,[equal('Outlook',sunny),greater('Humidity_%',75)],class('Don't_Play'),
                                                              [63.0],c45).
rule(4,[equal('Outlook',rain),equal('Windy',true)],class('Don't_Play'),[50.0],c45).

default(class('Play'),c45).
```

Tabela 5.18: Fatos Prolog Gerados após Pré-processamento do Arquivo de Regras do $\mathcal{C}4.5$

5.4.4 Verificar Base de Fatos

O objetivo principal desse processo é verificar massivamente a Base de Fatos procurando por possíveis inconsistências entre os dados armazenados na memória, já que o usuário pode,

eventualmente, ter fornecido arquivos de entrada de domínios diferentes.

Primeiramente é construído um dicionário com as *features* e depois é verificada a consistência dos exemplos e das regras. Em casos de erro uma mensagem é escrita no console do sistema informando o tipo de erro e o número do exemplo, ou da regra, onde foi encontrado esse erro. Essa verificação só termina após todos os exemplos e regras terem sido analisados.

A Figura 5.4 mostra os sub-processos responsáveis pela verificação da Base de Fatos. Tais sub-processos são:

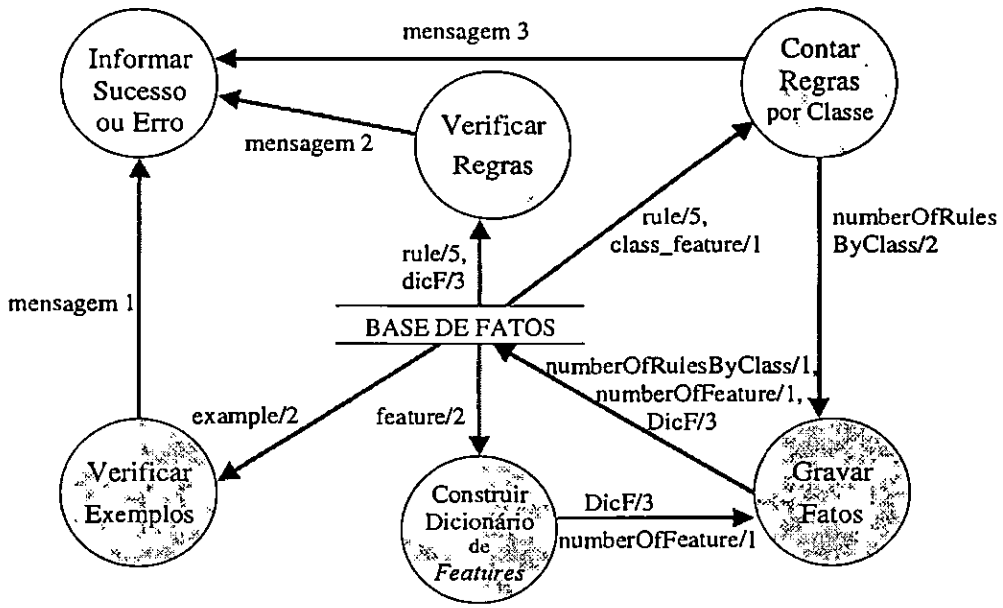

Figura 5.4: DFD do Processo Verificar Base de Fatos

Construir dicionário de *features*: cada fato `feature/2`, previamente armazenado na Base de Fatos é inserido em um dicionário de *features* balanceado onde a chave é o nome da *feature*. Usar um dicionário balanceado é vantajoso especialmente quando muitas buscas são requisitadas na estrutura (Monard & Nicoletti, 1993a; Monard & Nicoletti, 1993b). Cada elemento do dicionário é um termo composto

<feature_name>(<feature_number>,<type>).

Por exemplo, a Figura 5.5 ilustra o dicionário gerado pelas *features* da Tabela 5.11.

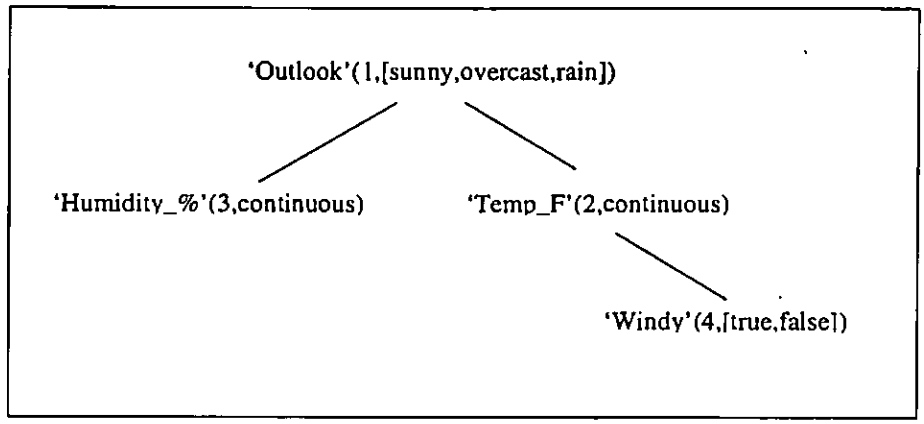

Figura 5.5: Dicionário de *Features*

Verificar exemplos: cada fato `example/2` da Base de Fatos é analisado. É verificado se o nome da classe é válido, *i.e.* se é um elemento da lista que é o argumento de `class_feature/1`. Os outros valores de cada exemplo (um valor para cada *feature*) são verificados com o auxílio do dicionário de *features*, *i.e.* se o tipo da *feature* é contínuo ou, no caso de ser discreto, se o valor da *feature* é um elemento da lista que difere dos possíveis valores que essa *feature* pode assumir. O número total de exemplos previamente processados `numberOfExample/1` — Seção 5.4.2, pg. 70 — também é conferido;

Verificar regras: cada fato `rule/5` da Base de Fatos é analisado. É verificado, entre outros, a validade dos seguintes itens: se as *features* presentes nas condições de cada regra estão presentes no dicionário; se o valor de cada condição está de acordo com o tipo da respectiva *feature*; se a classe da regra está presente em `class_feature/1`; se o numero de elementos da lista que informa o número de exemplos de treinamento de cada classe, cobertos pela regra (no caso de regras geradas pelo $C\mathcal{N}2$) corresponde ao número de classes;

Contar regras por classe: conta o número de regras existentes para cada classe, para posteriormente armazenar essa informação na Base de Fatos como `numberOfRulesByClass/1` cujo argumento é uma lista na qual cada elemento corresponde ao número de regras na classe segundo a informação contida em `class_feature/1`. Por exemplo, para as regras geradas por $C\mathcal{N}2$ — Tabela 5.7, pg. 66 — tem-se

```
numberOfRulesByClass([4,2],cn2).
```

e para as geradas por $C4.5$ — Tabela 5.9, pg. 68 — tem-se

```
numberOfRulesByClass([2,2],c45).
```

Também é verificada se a soma desses valores corresponde ao número total de regras `numberOfRules/2` anteriormente armazenada na Base de Fatos;

Gravar fatos: armazena na Base de Fatos o dicionário de *features* como um fato `DicF/3`, o número total de features `numberOfFeature/1` e o número de regras geradas para cada classe, `numberOfRulesByClass/2`;

Informar sucesso ou erro: avalia as mensagens recebidas dos processos de verificação da Base de Fatos. Quando uma dessas mensagens indica que foi encontrado algum problema de consistência entre os dados, tal ocorrência é informada ao usuário que deverá corrigir a origem do problema antes de continuar utilizando o sistema.

5.4.5 Gravar Base de Fatos

Esse processo do Módulo `1` é executado opcionalmente, mediante indicação do usuário, e possibilita gravar toda a Base de Fatos em arquivo. Essa é uma opção interessante quando se pretende trabalhar outras vezes com a mesma Base de Fatos, evitando assim a repetição dos processos de pré-processamento dos dados de entrada. Um outro processo, descrito na Seção 5.6.1, é capaz de ler posteriormente esse arquivo reconstituindo a Base de Fatos.

5.5 Descrição da Base de Fatos

A seguir é feita uma descrição resumida dos fatos gravados na Base de Fatos pelos processos do Módulo 1 do $\mathcal{RQ}_{system}$ — Figura 5.3, pg. 69.

- `class_feature/1`: tem como argumento uma lista com o nome das classes, na mesma ordem em que elas aparecem no arquivo de *features*;

- **feature/2**: é gravado um fato para cada *feature*, tem como argumentos o nome da *feature* e o respectivo tipo;
- **numberOfFeature/1**: tem como argumento o número total de *features* no arquivo de *features*;
- **example/2**: é gravado um fato para cada exemplo, tem como argumentos o número de ordem em que o exemplo aparece no arquivo de exemplos e uma lista com os valores dos atributos que descrevem esse exemplo;
- **numberOfExample/1**: tem como argumento o número total de exemplos lidos do arquivo de exemplos;
- **rule/5**: tem como argumentos os dados referentes a cada regra: o número da regra, uma lista com as condições no corpo da regra, o nome da classe predita, uma lista com o número de exemplos cobertos (no caso de regras do $CN2$) ou uma lista com um número que indica a precisão esperada para essa regra quando são apresentados exemplos ainda não vistos (no caso de regras do $C4.5$) e o nome do algoritmo que gerou a regra. Esses dados podem variar conforme o algoritmo de aprendizado;
- **numberOfRule/2**: tem como argumentos o número total de regras lidas do arquivo de regras e o nome do algoritmo que gerou essas regras;
- **default/<A_i>**: tem como argumentos uma lista com o número de exemplos para cada classe e pode conter outros dados. O número de argumentos pode variar conforme o algoritmo A_i que gerou as regras;
- **numberOfRulesByClass/2**: tem como argumentos uma lista cujos elementos indicam o número total de regras para cada uma das classes, na mesma ordem em que elas encontram-se no fato `class_feature/1`;
- **dicF/3**: dicionário de *features* balanceado; cada elemento do dicionário corresponde a uma *feature* contendo o nome da *feature* como chave e também o tipo e o número da *feature*.

5.6 Módulo 2 - Processamento de Informações

O Módulo 2, com base nas informações contidas na Base de Fatos, é responsável por fornecer diversas informações ao usuário. Ele é composto por dois processos gerais conforme a

representação do DFD na Figura 5.6.

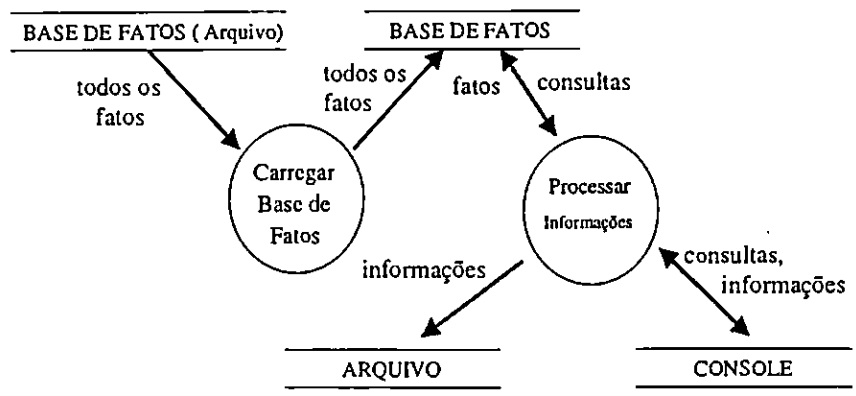

Figura 5.6: DFD do Módulo 2 do $\mathcal{RQ}_{system}$

5.6.1 Carregar Base de Fatos

Esse processo é executado opcionalmente mediante indicação do usuário. Ele possibilita que uma Base de Fatos, previamente gravada em arquivo, seja reconstituída na memoria. Em outras palavras, essa opção é válida quando já existe uma Base de Fatos gravada em arquivo e deseja-se utilizá-la para obter novas informações. De outro lado, quando não se tem uma Base de Fatos gravada em arquivo, ou não se deseja utilizá-la, todos os processos do Módulo 1 devem anteceder a execução do Módulo 2.

5.6.2 Processar Informações

O objetivo desse processo é gerar uma série de informações a partir de consultas à Base de Fatos ou ativando processos que geram informações previamente definidas no sistema. As informações retornadas são enviadas para o console mas podem, opcionalmente, serem gravadas em um arquivo determinado pelo usuário.

Os seguintes sub-processos, descritos conforme a representação do DFD na Figura 5.7, realizam o processamento de informações.

> **Processar Informações Pré-definidas:** Esse processo executa consultas à Base de Fatos utilizando procedimentos já definidos no $\mathcal{RQ}_{system}$ para retornar

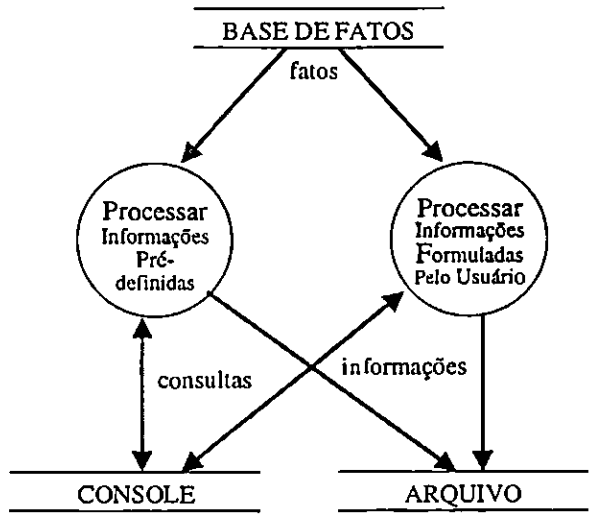

Figura 5.7: DFD do Processo Processar Informações

informações específicas. Algumas dessas informações estão relacionadas a cada regra, tais como

- número de condições no antecedente;
- número, e quais exemplos são cobertos na classe predita pela regra;
- número, e quais exemplos são cobertos fora da classe predita pela regra.

As informações retornadas podem também estar relacionadas a um conjunto de regras, por exemplo, para as regras que predizem uma determinada classe:

- total de exemplos cobertos pelas regras para a classe determinada;
- total de exemplos cobertos pelas regras fora da classe determinada.

Outras necessitam de alguns cálculos, por exemplo, as medidas de qualidade de cada regra:

- consistência $Cons(R)$;
- completeza $Comp(R)$;
- estatística de Cohen $Q_{Cohen}(R)$;
- estatística IMAFO $Q_{IMAFO}(R)$;
- estatística de Coleman $Q_{Coleman}(R)$;
- estatística SKIB1 $Q_{SKIB1}(R)$;

- estatística SKIB2 $Q_{SKIB2}(R)$;
- estatística IKIB $Q_{IKIB}(R)$.

Ainda há outras informações que necessitam maior computação, isto é, constantes consultas à Base de Fatos e muitos cálculos, como por exemplo as medidas de interessabilidade de cada regra:

- grau de surpresa de pequenos disjuntos $SurpDisj_{norm}(R)$;
- grau de surpresa dos atributos $SurpAtr(R)$.

Processar Informações Formuladas pelo Usuário: o usuário pode obter informações sobre a Base de Fatos formulando suas próprias consultas em Prolog com a possibilidade de combina'-las com os procedimentos pré-definidos no $\mathcal{RQ}_{system}$. Assim, devido a versatilidade da linguagem Prolog, o usuário tem a possibilidade de extrair todo tipo de informação dessa Base de Fatos, tais como:

Quais regras têm um determinado número de condições? ou maior ou menor que um valor específico?

Quais as regras nas quais aparece um determinado atributo?

Quais as regras nas quais um determinado atributo é comparado com um valor específico?

Quais os atributos utilizados no conjunto total de regras?

Quais os atributos que aparecem mais (e menos) frequentemente no conjunto total de regras?

Quais os exemplos cobertos por mais de uma regra?

Quais os exemplos não cobertos por alguma regra?

Quais regras cobrem exemplos em comum? Quais esses exemplos?

Qual o número médio de condições no conjunto total de regras?

Quais as maiores (ou menores) regras em termos de número de condições ou número de exemplos cobertos?

Dadas duas medidas de qualidade X e Y e $\mathrm{R} \in \{<, \leq, =, \geq, >\}$

Quais as regras que verificam X R Y*?*

Quais as regras que possuem máximo (ou mínimo) grau de surpresa de pequenos disjuntos?

Quais os atributos que possuem maior (ou menor) ganho de informação?

Uma descrição detalhada dos procedimentos pré-definidos implementados no Módulo 2 do $\mathcal{RQ}_{system}$ encontram-se em (Monard & Horst, 1999a).

5.7 Exemplo de Utilização do $\mathcal{RQ}_{system}$

A fim de exemplificar o uso do $\mathcal{RQ}_{system}$ foi utilizado o conjunto de dados "lenses", obtido do repositório UCI Irvine (Blake et al., 1998), descrito na Tabela 5.19.

Exemplo	Atributos				Classe
	age_of_the_patient	spectacle_prescription	astigmatic	tear_production_rate	
1	young	myope	no	reduced	no_contact_lenses
2	young	myope	no	normal	soft_contact_lenses
3	young	myope	yes	reduced	no_contact_lenses
4	young	myope	yes	normal	hard_contact_lenses
5	young	hypermetrope	no	reduced	no_contact_lenses
6	young	hypermetrope	no	normal	soft_contact_lenses
7	young	hypermetrope	yes	reduced	no_contact_lenses
8	young	hypermetrope	yes	normal	hard_contact_lenses
9	pre-presbyopic	myope	no	reduced	no_contact_lenses
10	pre-presbyopic	myope	no	normal	soft_contact_lenses
11	pre-presbyopic	myope	yes	reduced	no_contact_lenses
12	pre-presbyopic	myope	yes	normal	hard_contact_lenses
13	pre-presbyopic	hypermetrope	no	reduced	no_contact_lenses
14	pre-presbyopic	hypermetrope	no	normal	soft_contact_lenses
15	pre-presbyopic	hypermetrope	yes	reduced	no_contact_lenses
16	pre-presbyopic	hypermetrope	yes	normal	no_contact_lenses
17	presbyopic	myope	no	reduced	no_contact_lenses
18	presbyopic	myope	no	normal	no_contact_lenses
19	presbyopic	myope	yes	reduced	no_contact_lenses
20	presbyopic	myope	yes	normal	hard_contact_lenses
21	presbyopic	hypermetrope	no	reduced	no_contact_lenses
22	presbyopic	hypermetrope	no	normal	soft_contact_lenses
23	presbyopic	hypermetrope	yes	reduced	no_contact_lenses
24	presbyopic	hypermetrope	yes	normal	no_contact_lenses

Tabela 5.19: Conjunto de Exemplos de Treinamento lenses

O arquivo de features correspondente a esse conjunto de exemplos está descrito na Tabela 5.20.

hard_contact_lenses, soft_contact_lenses, no_contact_lenses.
age_of_the_patient: young, pre-presbyopic, presbyopic.
spectacle_prescription: myope, hypermetrope.
astigmatic: no, yes.
tear_production_rate: reduced, normal.

Tabela 5.20: Arquivo de *Features* do Conjunto de Dados lenses

Os algoritmos $C\mathcal{N}2$ e $C4.5$ foram executados com parâmetros *default* utilizando a biblioteca $\mathcal{MLC}$++. Primeiramente foi medida a precisão de ambos os algoritmos usando 10-*fold stratified cross-validation*. Os resultados obtidos para $C\mathcal{N}2$ foram: erro médio e desvio padrão de $20.00\% \pm 8.54\%$ com erro máximo de 66.70% e mínimo de 0%. Os resultados obtidos para $C4.5$ foram: erro médio e desvio padrão de $15.00\% \pm 8.03\%$ com erro máximo de 66.66% e mínimo de 0%.

Utilizando a Equação 3.22, pg. 29 para comparar ambos os algoritmos obtém-se

$$\frac{|20-15|}{\sqrt{\frac{8.54^2+8.03^2}{2}}} = 0.603$$

Assim, para esse conjunto de exemplos, como o erro do $C\mathcal{N}2$ é maior que o do $C4.5$, pode-se concluir, segundo esse teste estatístico para comparar algoritmos de aprendizado, que $C4.5$ supera $C\mathcal{N}2$.

Em seguida, todo o conjunto de exemplos foi fornecido para o $C\mathcal{N}2$ e o $C4.5$ que geraram, respectivamente, as regras listadas nas Tabelas 5.21 e 5.22.

Para ambos os algoritmos, o $\mathcal{RQ}_{system}$ foi executado utilizando o arquivo de *features* da Tabela 5.20. Para o arquivo de exemplos foram considerados, na sintaxe requerida pelo sistema, todos os exemplos de treinamento descritos na Tabela 5.19. Para $C\mathcal{N}2$ foi utilizado o arquivo de regras descrito na Tabela 5.21 e para o $C4.5$ o descrito na Tabela 5.22.

```
IF spectacle_prescription = myope
AND astigmatic = yes
AND tear_production_rate = normal
THEN class = hard_contact_lenses [3 0 0]

IF age_of_the_patient = young
AND astigmatic = yes
AND tear_production_rate = normal
THEN class = hard_contact_lenses [2 0 0]

IF astigmatic = no
AND tear_production_rate = normal
THEN class = soft_contact_lenses [0 5 1]

IF tear_production_rate = reduced
THEN class = no_contact_lenses [0 0 12]

IF spectacle_prescription = hypermetrope
AND astigmatic = yes
THEN class = no_contact_lenses [1 0 5]

IF age_of_the_patient = presbyopic
AND spectacle_prescription = myope
AND astigmatic = no
THEN class = no_contact_lenses [0 0 2]

(DEFAULT) class = no_contact_lenses [4 5 15]
```

Tabela 5.21: Arquivo de Regras do $\mathcal{CN}2$ `lenses.cn2.rules`

```
Rule 3:
spectacle_prescription = myope
astigmatic = yes
tear_production_rate = normal
-> class hard_contact_lenses [63.0%]

Rule 1:
tear_production_rate = reduced
-> class no_contact_lenses [89.1%]

Rule 2:
astigmatic = no
tear_production_rate = normal
-> class soft_contact_lenses [61.2%]

Default class: no_contact_lenses
```

Tabela 5.22: Arquivo de Regras do $\mathcal{C}4.5$ `lenses.c45.rules`

As diversas informações extraídas pelo $\mathcal{RQ}_{system}$ encontram-se resumidas a seguir.

A Tabela 5.23 mostra, para $\mathcal{CN}2$, o número de condições no antecedente de cada regra, e a cobertura de cada regra.

R	cond	class hard_contact_lenses	not class hard_contact_lenses
1	3	3	0
2	3	2	0
total		5	0
R	cond	class soft_contact_lenses	not class soft_contact_lenses
3	2	5	1
total		5	1
R	cond	class no_contact_lenses	not class no_contact_lenses
4	1	12	0
5	2	5	1
6	3	2	0
total		19	1

Tabela 5.23: Número de Condições e Cobertura das Regras Geradas por $\mathcal{CN}2$

Deve ser observado que essa cobertura é idêntica a contida no arquivo de regras, pois foram utilizados no arquivo de exemplos os mesmo exemplos utilizados para gerar as regras. No entanto, como mencionado anteriormente, qualquer outro conjunto de exemplos, do mesmo domínio, poderia ser considerado no arquivo de exemplos.

A Tabela 5.24 mostra informação semelhante para $C4.5$.

R	cond	class hard_contact_lenses	not class hard_contact_lenses
1	3	3	0
total		3	0
R	cond	class no_contact_lenses	not class no_contact_lenses
2	1	12	0
total		12	0
R	cond	class soft_contact_lenses	not class soft_contact_lenses
3	2	5	1
total		5	1

Tabela 5.24: Número de Condições e Cobertura das Regras Geradas por $C4.5$

As Tabelas 5.25 e 5.26 mostram para $CN2$ e $C4.5$, respectivamente, as medidas de qualidade de cada uma das regras induzidas.

R	$Comp(R)$	$Cons(R)$	$Q_{Cohen}(R)$	$Q_{IMAFO}(R)$	$Q_{Coleman}(R)$	$Q_{SKIB1}(R)$	$Q_{SKIB2}(R)$	$Q_{IKIB}(R)$
1	0.750	1	0.833	7.464	1	0.944	0.875	2.585
2	0.500	1	0.625	5.560	1	0.875	0.750	2.585
3	1	0.833	0.882	9.583	0.789	0.759	0.789	2.000
4	0.800	1	0.750	7.164	1	0.917	0.900	0.678
5	0.333	0.833	0.185	2.781	0.556	0.405	0.370	0.415
6	0.133	1	0.103	1.927	1	0.701	0.567	0.678

Tabela 5.25: Medidas de Qualidade de Regras Geradas por $CN2$

R	$Comp(R)$	$Cons(R)$	$Q_{Cohen}(R)$	$Q_{IMAFO}(R)$	$Q_{Coleman}(R)$	$Q_{SKIB1}(R)$	$Q_{SKIB2}(R)$	$Q_{IKIB}(R)$
1	0.750	1	0.833	7.464	1	0.944	0.875	2.585
2	0.800	1	0.750	7.164	1	0.917	0.900	0.678
3	1	0.833	0.882	9.583	0.789	0.759	0.789	2.000

Tabela 5.26: Medidas de Qualidade de Regras Geradas por $C4.5$

Pode ser observado que todas as medidas de qualidade, das regras R_1, R_2 e R_3, geradas por $C4.5$, são idênticas, respectivamente, as regras R_1, R_4 e R_3 geradas por $CN2$. Esse é um

forte indício de que essas regras são, respectivamente, iguais. Para o conjunto de exemplos aqui considerado, esse é o caso, como pode ser observado nas Tabelas 5.21 e 5.22.

Outra informação interessante obtida pelo $\mathcal{RQ}_{system}$ está relacionada com os atributos que aparecem nas regras, como mostram as Tabelas 5.27 e 5.28.

R	Atributos			
	age_of_the_patient	spectacle_prescription	astigmatic	tear_production_rate
1		•	•	•
2	•		•	•
3			•	•
4				•
5		•	•	
6	•	•	•	

Tabela 5.27: Atributos Presentes nas Regras Geradas por $\mathcal{CN}2$

R	Atributos			
	age_of_the_patient	spectacle_prescription	astigmatic	tear_production_rate
1		•	•	•
2				•
3			•	•

Tabela 5.28: Atributos Presentes nas Regras Geradas por $\mathcal{C}4.5$

Com base nesses resultados é possível observar que o atributo *age_of_the_patient*, que aparece nas regras geradas por $\mathcal{CN}2$, é considerado irrelevante por $\mathcal{C}4.5$. Também pode ser observado que para $\mathcal{C}4.5$ o atributo mais relevante, segundo o critério utilizado por $\mathcal{C}4.5$, *i.e.* aquele que é a raiz da árvore de decisão, é o atributo *tear_production_rate* seguido do atributo *astigmatic* e, finalmente do atributo *spectacle_prescription*. $\mathcal{CN}2$ utiliza um *bias* completamente diferente para induzir as regras. Ainda assim, se considerarmos a importância de um atributo como sendo o número de vezes que é referenciado no conjunto total de regras, poderiamos concluir que para $\mathcal{CN}2$ o atributo mais relevante é *astigmatic*, seguido de *tear_production_rate*, *spectacle_prescription* e por último, *age_of_the_patient*.

Também é possível obter informações referentes a quais exemplos são cobertos por cada regra, como mostram as Tabela 5.29 e 5.30, onde o símbolo o indica que esse exemplo está rotulado com uma classe diferente da predita pela regra que cobre esse exemplo.

R	Exemplos																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1				•								•								•				
2				•				•																
3		•				•				•				•				o				•		
4	•		•		•		•		•		•		•		•		•		•		•		•	
5							•	o							•	•							•	•
6																	•	•						

Tabela 5.29: Exemplos Cobertos pelas Regras Geradas por $\mathcal{CN}2$

R	Exemplos																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1				•								•								•				
2	•		•		•		•		•		•		•		•		•		•		•		•	
3		•				•				•				•				o				•		

Tabela 5.30: Exemplos Cobertos pelas Regras Geradas por $\mathcal{C}4.5$

Aqui também pode ser observado a diferença entre ambos os algoritmos. Em geral, $\mathcal{CN}2$ tende a gerar regras que cobrem os mesmos exemplos, como é o caso dos exemplos 4, 7, 8, 15, 17, 18 e 23 — Tabela 5.29. Isto não acontece com $\mathcal{C}4.5$, que utiliza hiperplanos para dividir, por classes, o espaço de exemplos. Por outro lado, isso faz com que $\mathcal{C}4.5$, em geral, não possa cubrir todos os exemplos, como é o caso dos exemplos 8, 16 e 24 — Tabela 5.30 — que não são cobertos por nenhuma das três regras geradas por $\mathcal{C}4.5$, os quais deveriam ser classificados utilizando a regra *default*. Essa regra classificaria esses exemplos com a classe *no_contact_lenses*. Nesse caso, somente o exemplo 8 seria incorretamente classificado, já que ele está rotulado com a classe *hard_contact_lenses*.

O $\mathcal{RQ}_{system}$ fornece ambas as opções, *i.e.* classificar exemplos utilizando somente as regras geradas, como aqui apresentado, ou considerando também a regra *default*. Essa opção é interessante pois permite diferenciar os exemplos erroneamente classificados por uma regra, como é o caso do exemplo 18 para $\mathcal{C}4.5$, e o exemplo 8 que seria erroneamente classificado, mas pela regra *default*.

Em geral, $\mathcal{CN}2$ tende a deixar menos exemplos sem cobrir do que $\mathcal{C}4.5$ mas, como já observado, as regras podem cobrir os mesmos exemplos. Nesses casos, pode acontecer que essas regras predizem classes diferentes para o mesmo exemplo. Isto pode ser observado nos exemplos 8 e 18 na Tabela 5.29, onde é mostrada a cobertura *individual* de cada regra. Qual é nesses casos a classificação dada por $\mathcal{CN}2$ considerando todo o conjunto de regras?

$\mathcal{CN}2$ considera todas as regras que cobrem o mesmo exemplo, regras 2 e 5 para o exemplo 8,

e calcula, com base na informação da lista de números que informa o número de exemplos de treinamento, de cada classe, cobertos pelas regras, a classificação final desse exemplo. Essa lista é [2 0 0] para a rgra 2, e [1 0 5] para a regra 5. $\mathcal{CN}2$ adiciona os elementos de ambas as listas obtendo [3 0 5], e classifica o exemplo como sendo da classe majoritária nessa lista, ou seja *no_contact_lenses*. Como o exemplo 8 está rotulado como *hard_contact_lenses*, ele será erroneamente classificado por $\mathcal{CN}2$. Analogamente para o exemplo 18 tem-se [0 5 2] para a regra 3 e [0 0 2] para a regra 6, obtendo assim [0 5 3] e o exemplo será incorretamente classificado como sendo da classe *soft_contact_lenses*.

As Tabelas 5.31 e 5.32 mostram para $\mathcal{CN}2$ e $\mathcal{C}4.5$ respectivamente, o grau de surpresa normalizado de pequenos disjuntos $SurpDisj_{norm}(R)$ — Equação 4.22, pg. 53 — e o grau de surpresa de uma regra em função do grau de surpresa dos atributos individuais da regra $SurpAtr(R)$ — Equação 4.26, pg. 56.

R	$SurpDisj_{norm}(R)$	$SurpAtr(R)$
1	0	3.108
2	0	3.108
3	0.5	2.160
4	0	1.822
5	0	4.802
6	0	6.580

Tabela 5.31: Medidas de Interessabilidade de Regras Geradas por $\mathcal{CN}2$

R	$SurpDisj_{norm}(R)$	$SurpAtr(R)$
1	0	3.108
2	0	1.822
3	0.5	2.160

Tabela 5.32: Medidas de Interessabilidade de Regras Geradas por $\mathcal{C}4.5$

Segundo a primeira medida, somente a regra 3, que é idêntica para ambos os algoritmos, mereceria alguma atenção. Considerando a segunda medida, as regras 6, 5 e 3 de $\mathcal{CN}2$ se destacam, bem como as regras 1 e 3 de $\mathcal{C}4.5$. Entretanto, como observado na Seção 4.5.6, pg. 54, é possível argumentar que uma regra cujo antecedente contém somente atributos com baixo ganho de informação pode ser considerada mais surpreendente que outra regra cujo antecedente contém atributos com maior ganho de informação, somente se outros fatores tais como precisão da predição, cobertura e completeza forem semelhantes para ambas as

regras. Segundo os resultados obtidos — Tabelas 5.25 e 5.26, pg. 85 — esse não é o caso para os dois subconjuntos de regras considerados.

O conjunto de regras utilizado para ilustrar o $\mathcal{RQ}_{system}$ é muito pequeno. Em situações reais, o número de regras a ser analisado é grande. Justamente nessas situações tanto as facilidades oferecidas pelo $\mathcal{RQ}_{system}$ como a possibilidade dada ao usuário de definir novos procedimentos podem ser melhor apreciadas (Baranauskas et al., 1999a; Baranauskas et al., 1999b). Por exemplo, para encontrar quais as regras mais interessantes, o usuário poderia utilizar (ou definir outro procedimento) o seguinte procedimento já definido no sistema

```
similar(Medida1,Medida2,Diferenca):-
    Z is Medida1 - Medida2,
    Y is abs(Z),
    Y =< Diferenca.
```

o qual é bem sucedido se as diferenças, em valor absoluto, dos dois primeiros argumentos, é menor que o terceiro argumento. Dessa forma, é possível para o usuário procurar por todos os pares de regras com precisão da predição, cobertura, completeza (ou quaisquer outras medidas) semelhantes, onde o grau de semelhança `Diferenca` é definido pelo próprio usuário, tal que a interessabilidade de uma dessas regras seja superior a da outra.

5.8 Considerações Finais

O $\mathcal{RQ}_{system}$ por nós projetado e implementado, dispõe de recursos para auxiliar na avaliação de regras. Uma característica importante do sistema é que as consultas realizadas para obter tais informações não ficam restritas àquelas pré-definidas no sistema, várias outras podem ser construídas e formuladas livremente pelo usuário.

Outro fator a ser destacado no $\mathcal{RQ}_{system}$ é a possibilidade de se acrescentar facilmente novos recursos, tais como, procedimentos para reconhecer regras geradas por outros algoritmos e novos procedimentos que definem outras medidas de qualidade de regras.

As características do $\mathcal{RQ}_{system}$, o tornam um sistema útil para auxiliar na avaliação de regras bem como permite estudar outras importantes questões relacionadas a regras.

Capítulo 6

Conclusões

Muitas pesquisas têm sido conduzidas na área de Aprendizado de Máquina, no entanto, ainda há várias questões importantes em aberto. Neste trabalho foram abordadas algumas dessas questões relacionadas com a aquisição de conhecimento, principalmente sobre a avaliação do conhecimento adquirido por algoritmos de AM.

Diferentes maneiras podem ser usadas para avaliar o conhecimento, isso depende de vários fatores, como por exemplo o tipo de representação. Neste trabalho nos concentramos no conhecimento expresso por regras. O uso de estatística e teoria da informação têm sido utilizadas para fornecer medidas úteis para a avaliação de regras. Existem várias medidas objetivas e subjetivas já propostas, nós focalizamos o uso de medidas objetivas por serem autônomas e computáveis, mas ambas são interessantes e podem até ser utilizadas conjuntamente.

A qualidade e a interessabilidade das regras são dois aspectos muito importantes que podem ser avaliados utilizando-se essas medidas, no entanto, outras informações relacionadas com as regras podem ser utilizadas pelo usuário para avaliá-las. Todavia, obter e analisar essas informações pode não ser uma tarefa fácil de se realizar manualmente, pois usualmente o número de regras é elevado bem como a quantidade de dados associados as regras e os cálculos necessários para obter tais informações. Assim, foi decidido projetar e implementar um sistema computacional, denominado $\mathcal{RQ}_{system}$, com o intuito de auxiliar na avaliação da qualidade das regras induzidas por algoritmos de Aprendizado de Máquina.

Pode ser observado, através dos simples exemplos utilizados no capítulo anterior para ilustrar as facilidades por nós implementadas no $\mathcal{RQ}_{system}$, que as informações obtidas com esse sistema podem ser muito úteis, tanto para avaliar as regras permitindo que o usuário se

concentre naquelas mais interessantes, quanto para estudar outras importantes questões relacionadas com a qualidade dessas regras.

O $\mathcal{RQ}_{system}$ foi por nós projetado de forma a permitir que novas características sejam futuramente adicionadas ao sistema. Portanto, o sistema pode ser estendido para ampliar sua aplicação e utilidade.

A versão atual do sistema reconhece regras geradas pelos algoritmos $\mathcal{CN}2$ e $\mathcal{C}4.5$. No caso do $\mathcal{C}4.5$, é utilizado especificamente o $\mathcal{C}4.5rules$ o qual, após efetuar uma poda na árvore de decisão, escreve a árvore resultante em forma de regras de decisão. O $\mathcal{RQ}_{system}$ pode ser facilmente estendido para reconhecer regras do $\mathcal{C}4.5tree$, que induz árvores de decisão, utilizando o programa TreeToRules desenvolvido por (Neto, 1999) na PUC-PR.

Deve ser observado que o $\mathcal{RQ}_{system}$ está integrado a um projeto mais geral em andamento no Laboratório de Inteligência Computacional (LABIC) no ICMC-USP. Esse projeto visa incorporar vários recursos, tais como sistemas, ferramentas e informações, desenvolvidos e/ou utilizados nas diversas pesquisas de Aprendizado de Máquina e Data Mining realizadas no LABIC (Baranauskas, 1998; Batista et al., 1999; Caulkins, 1999; Nagai, 1999; Oliveira, 1999; Rocha, 1999).

Data Mining (DM) constitui um campo de pesquisa recente em IA, cujo objetivo é extrair conhecimento de grandes bases de dados. Um dos tópicos tratados em DM para extrair conhecimento é o uso de algoritmos de AM com grandes quantidades de dados. Em geral, algoritmos de AM se aplicam a conjuntos de dados bem menores que os encontrados em grandes bases de dados. Para ultrapassar essa limitação, outras abordagens, tais como diminuição da dimensão dos dados (Lee, 1999), e amostragens de dados devem ser investigadas.

Nesse contexto, consideramos que o $\mathcal{RQ}_{system}$ será muito útil para testar diversas idéias, antes de elas serem implementadas em outras ferramentas em desenvolvimento no LABIC para a área de Data Mining, devido as facilidades de prototipagem rápida oferecidas pela linguagem Prolog.

São vários os trabalhos futuros a serem realizados como resultado deste trabalho. O mais simples é reconhecer e incorporar regras geradas por outros algoritmos proposicionais. Um trabalho mais complexo é estender o sistema para processar regras relacionais, como as induzidas pelos algoritmos de AM baseados em Programação Lógica Indutiva (Caulkins, 1999).

O $\mathcal{RQ}_{system}$ pode ser estendido para permitir armazenar juntamente na memória regras

geradas por um algoritmo mas que foram obtidas utilizando conjuntos diferentes de exemplos de treinamento, bem como regras geradas por algoritmos diferentes. Isso acrescentaria uma importante característica da área de Data Mining ao $\mathcal{RQ}_{system}$.

Como já mencionado, em DM o conjunto de exemplos é muito grande. Assim, é necessário induzir regras utilizando várias amostras distintas de exemplos. Nesses casos o algoritmo de AM utilizado considera cada uma dessas amostras como um conjunto de treinamento e induz as regras correspondentes a esse conjunto. Assim, têm-se tantos conjuntos de regras induzidas quanto amostras retiradas da Base de Dados. Finalmente, para encontrar as regras que representam o conhecimento de toda essa grande base de dados é necessário procurar por um conjnto mínimo de regras em função das regras induzidas pelo algoritmo de AM utilizando cada amostra.

Também, alguns dos temas tratados neste trabalho poderiam ser mais aprofundados, por exemplo, estudos sobre estimadores de algoritmos, novas medidas de qualidade e interessabilidade de regras. Desses estudos podem surgir pontos importantes a serem acrescentados ao $\mathcal{RQ}_{system}$.

Apêndice A

Alguns Termos e Conceitos de Estatística

Neste trabalho são usados vários termos e conceitos estatísticos. Uma breve definição de alguns deles extraída de (Achcar & Rodrigues, 1997; Johnson & Bhattacharyya, 1992; Mitchell, 1997) é apresentada a seguir.

A.1 Conceitos Básicos

Uma *população* é o conjunto de medições (ou registro de alguma característica qualitativa) correspondente a uma coleção de unidades para as quais as inferências estão sendo feitas.

Uma *amostra de uma população* estatística é o conjunto de medições que são colecionadas para uma determinada investigação.

A.2 Medidas Descritivas

As medidas descritivas são usadas para sumarizar as informações contidas nos dados em termos de variabilidade e formas das distribuições de frequências.

A *média amostral* de um conjunto de n medições $x_1, x_2, \ldots, x_n$ é a soma destas medições dividida por n. A média amostral é denotada por $\bar{x}$ e pode ser expressa por

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

A *mediana de uma amostra* de um conjunto de n medições $x_1, \ldots, x_n$ é o valor do meio de uma sequência de medições que estejam arranjadas em ordem crescente.

A *variância de uma amostra* de n observações é a soma dos desvios quadrados dividida pelo número de observações menos um. Sua expressão é:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

O *desvio padrão de uma amostra* é a raiz quadrada da variância e serve como uma medida de variabilidade na mesma unidade em que estão os dados. É expresso por:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

A lei (teorema) de Chebyschev estabelece que para qualquer conjunto de dados, o intervalo $(\bar{x} - ks; \bar{x} + ks)$ contém pelo menos a proporção $(1 - \frac{1}{k^2})$ das observações. Em particular, o intervalo $(\bar{x} - 2s; \bar{x} + 2s)$ contém pelo menos $(1 - \frac{1}{2^2}) = \frac{3}{4}$ ou 75% dos dados.

Uma *variável aleatória* X associa um valor numérico para cada resultado de um experimento. Tal variável pode assumir valores discretos ou contínuos.

A *distribuição de probabilidade*, ou simplesmente distribuição de uma variável aleatória discreta X, é uma lista de valores numéricos distintos de X com suas respectivas probabilidades associadas. Pode ser descrita pela função:

$$f(x_i) = P[X = x_i]$$

que deve satisfazer as condições:

i) $f(x_i) \geq 0$, para todo valor x_i de X

ii) $\sum_{i=1}^{k} f(x_i) = 1$

A *média da variável aleatória* X ou *média da população*, ou ainda chamada de *valor esperado*, é denotada por $E[X]$. A média e valor esperado são na verdade o mesmo valor expressos por:

$$\begin{aligned} E[X] &= \mu \\ &= \sum x_i f(x_i) \end{aligned}$$

A variância de X também chamada de variância da população é a somatória do produto de cada desvio ao quadrado pela sua probabilidade. Ela é denotada por σ^2 ou simplesmente $var(X)$

$$\sigma^2 = \sum (x_i - \mu)^2 f(x_i)$$

O desvio padrão de X, denotado por σ, é a raiz quadrada positiva da variância, isto é,

$$\sigma = \sqrt{var(X)}$$

Para qualquer distribuição de probabilidade tem-se algumas interpretações úteis:

i) a probabilidade de X ter um valor contido no intervalo $(\mu - 2\sigma; \mu + 2\sigma)$ é maior ou igual a $1 - \frac{1}{2^2} = 0,75$;

ii) a probabilidade de X ter um valor contido no intervalo $(\mu - 3\sigma; \mu + 3\sigma)$ é maior ou igual a $1 - \frac{1}{3^2} = 0,89$;

iii) a probabilidade de X ter um valor contido no intervalo $(\mu - k\sigma; \mu + k\sigma)$ é maior ou igual a $1 - \frac{1}{k^2}$.

A.3 Distribuição Binomial

Na distribuição Binomial são utilizados os *Ensaios de Bernoulli* que são situações amostrais onde os elementos de uma população assumem somente dois resultados possíveis denominados usualmente sucesso e falha. Por exemplo, peças defeituosas ou não defeituosas numa linha de produção.

Um ensaio de Bernoulli consiste em selecionar um elemento da população. De forma resumida estes ensaios são definidos da seguinte forma:

a) cada ensaio determina um resultado entre dois valores possíveis: sucesso (S) e falha (F);

b) em cada ensaio, a probabilidade de sucesso $P(S)$ é a mesma e é denotada por $p = P(S)$. A probabilidade de falha é $P(F) = 1 - p = q$, tal que $p + q = 1$;

c) os ensaios são independentes, isto é, a probabiliade de sucesso num ensaio não muda pelo fato de se conhecer qualquer informação sobre o resultado de outros ensaios.

A *distribuição binomial* é denotada por:

n = número fixo de ensaios de Bernoulli

p = probabilidade de sucesso em cada ensaio

X = número (aleatório) de sucessos em n ensaios

A variável aleatória X é chamada variável aleatória binomial e sua distribuição é chamada de distribuição binomial. A distribuição binomial com n ensaios e probabilidade de sucesso p é descrita por:

$$f(x) = P[X = x] = \binom{n}{x} p^x (1-p)^{n-x}$$

para os valores possíveis $x = 0, 1, \cdots, n$.

A *média*, *variância* e *desvio padrão* da distribuição binomial para n ensaios e p probabilidade de sucesso são respectivamente:

$E[X] = np$

$\sigma^2 = npq$ (onde $q = 1 - p$)

$\sigma = \sqrt{npq}$

A.4 Distribuição Normal

A *distribuição normal* que tem uma média μ e um desvio padrão σ é denotada por $N(\mu, \sigma)$.

A *distribuição normal padrão* tem uma densidade em forma de sino com:

média $\mu = 0$ e

desvio padrão $\sigma = 1$.

Uma variável aleatória Z tem distribuição normal padronizada se $\mu = E(Z) = 0$ e $\sigma^2 = 1$. Isto é, $Z \sim N(0, 1)$.

A.5 Aproximação Normal à Distribuição Binomial

Se X é uma variável aleatória que representa o número de sucessos em n ensaios independentes, então X tem uma distribuição binomial.

Quando a probabilidade de sucesso p não está próxima de 0 ou de 1 e o número de ensaios é grande, a distribuição normal serve como uma boa aproximação para a distribuição binomial. Para manter uma boa aproximação à normalidade deve-se usar uma correção de continuidade (somar e subtrair $\frac{1}{2}$ para cada valor x de uma variável aleatória X com distribuição binomial).

Desta forma, quando np e $n(1-p)$ são grandes (≥ 20), a distribuição binomial é bem aproximada pela distribuição normal, sendo a média $\mu = np$ e desvio padrão $\sigma = \sqrt{np(1-p)}$.

A.6 Testes de Hipótese

A *hipótese nula* e a *hipótese alternativa* quando usadas em testes de hipótese para estabelecer asserções substanciadas de uma amostra são formuladas da seguinte forma:

i) a negação da asserção é tomada como hipótese nula H_0 e

ii) a própria asserção é tomada como hipótese alternativa H_1.

Associado a um teste de hipóteses têm-se dois *tipos de erro* possíveis:

erro do Tipo I: rejeição de H_0 quando H_0 é verdadeira e

erro do Tipo II: não rejeição de H_0 quando H_1 é verdadeira.

As probabilidades de erro do Tipo I e do Tipo II são denotadas por α e β respectivamente, e seus valores máximos são chamados de níveis de significância.

A.7 Teorema do Limite Central

Seja qual for a população, a distribuição de $\bar{x}$ é aproximadamente normal quando n é grande. Em uma amostra aleatória de uma determinada população com média μ e desvio padrão σ,

quando n é grande (geralmente $n > 30$), a distribuição de $\bar{x}$ é aproximadamente normal com média μ e desvio padrão $\sigma/\sqrt{n}$. Consequentemente,

$$Z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

é aproximadamente $N(0, 1)$.

Referências

Achcar, J. A. & Rodrigues, J. (1997). Introdução à estatística para ciências e tecnologia. Departamento de Ciências Matemáticas e de Computação, ICMC-USP.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 1(6):37–66.

Baranauskas, J. A. (1998). Extração de conhecimento de bases de dados. Exame de Qualificação de Doutorado, ICMC-USP.

Baranauskas, J. A. & Monard, M. C. (1999). The MLC^{++} wrapper for feature subset selection using decision tree, production rule, instance based and statistical inducers: Some experimental results. Technical Report 87, ICMC-USP, ftp://ftp.icmsc.sc.usp.br/pub/BIBLIOTECA/rel_tec/.

Baranauskas, J. A., Monard, M. C., & Horst, P. S. (1999b). Evaluation of CN2 induced rules using feature selection. In *The Argentine Symposium on Artificial Intelligence - ASAI'99*, Argentina. In print.

Baranauskas, J. A., Monard, M. C., & Horst, P. S. (1999a). Evaluation of feature selection by wrapping around the CN2 inducer. In *Anais do II Encontro Nacional de Inteligência Artificial - ENIA'99*, pages 315–326, Rio de Janeiro.

Batista, G. E. A. P. A. (1997). Um ambiente de avaliação de algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC-USP.

Batista, G. E. A. P. A., Carvalho, A., & Monard, M. C. (1999). Aplicando seleção unilateral em conjuntos de exemplos desbalanceados: Resultados iniciais. In *Anais do II Encontro Nacional de Inteligência Artificial - ENIA'99*, pages 327–340, Rio de Janeiro.

Batista, G. E. A. P. A. & Monard, M. C. (1998). Seleção unilateral para melhorar a classificação de conjuntos de exemplos desbalanceados. In *Proceedings (Student Session) XIII Simpósio Brasilero de Inteligência Artificial - SBIA'98*, Porto Alegre.

Blake, C., Keogh, E., & Merz, C. J. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html.

Bratko, I. (1990). *Prolog Programming for Artificial Intelligence.* Addison-Wesley.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Morgan Kaufmann Publishers, Wadsworth, CA, USA.

Caulkins, C. W. (1999). Aquisição de conhecimento utilizando aprendizado de máquina relacional. Exame de Qualificação de Mestrado, ICMC-USP.

Clark, P. & Boswell, P. (1991). Rule induction with CN2: Some recent improvements. In Springer-Verlag, editor, *Proceedings of 5th European Conference - EWSL'91*, pages 151–163.

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3(4):261–283.

Clocksin, W. F. & Mellish, C. S. (1994). *Programming in Prolog.* Springer-Verlag.

Dean, P. & Famili, A. (1997). Comparative performance of rule quality measures in a induction system. *Applied Intelligence*, 7:113–124.

Dietterich, T. G. (1996). Proper statistical tests for comparing supervised classification learning algorithms. Technical Report, Department of Computer Science, University of State Oregon.

Dietterich, T. G. (1997). Statistical tests for comparing supervised classification learning algorithms. ftp://ftp.cs.orst.edu/pub/tgd/papers/stats.ps.gz.

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.

Flach, P. (1994). *Simply Logical Intelligente Reasoning by Example.* John Willey & Sons.

Freitas, A. A. (1998a). A multi-criteria approach for the evaluation of rule interestingness. In *Proceedings of the International Conference on Data Mining*, pages 7–20, Rio de Janeiro.

Freitas, A. A. (1998b). On objective measures of rule surprisingness. In *Principles of Data Mining & Knowledge Discovery: Proceedings of the Second European Symp. Lecture Notes in Artificial Intelligence*, volume 1510, pages 1–9.

Haykin, S. (1994). *Neural Networks A Comprehensive Foundation.* Macmillan College Publishing Company.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. *Machine Learning: An Artificial Intelligence Approach*, 2(27).

Horst, P. S. (1999). Avaliação do conhecimento adquirido por algoritmos de aprendizado de máquina utilizando exemplos. In *Anais do IV Workshop de Teses e Dissertações em Andamento do ICMC-USP*, pages 55–56.

Horst, P. S., Padilha, T. P. P., Rocha, C. A. J., Rezende, S. O., & Carvalho, A. C. P. L. (1997). Usando algoritmos simbólicos e conexionistas para avaliação de crédito. In *Anais do IV Simpósio Brasileiro de Redes Neurais – SBRN'97*, pages 91–94, Goiânia.

Horst, P. S., Padilha, T. P. P., Rocha, C. A. J., Rezende, S. O., & Carvalho, A. C. P. L. (1998). Knowledge acquisition using symbolic and connectionist algorithms for credit evaluation. In *IEEE World Congress on Computational Intelligence – WCCI'98*, pages 277–282, USA.

Johnson, R. A. & Bhattacharyya, G. K. (1992). *Statistics: Principles and Methods.* John Willey & Sons.

Kohavi, R., Sommerfield, D., & Dougherty, J. (1994). *MLC^{++}: A Machine Learning Library in C^{++}.* IEEE Computer Society Press.

Lavrač, N. & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, Londres, Inglaterra.

Lee, H. D. (1999). Seleção e construção de features relevantes para o aprendizado de máquina. Exame de Qualificação de Mestrado, ICMC-USP.

-/> Major, J. A. & Mangano, J. J. (1993). Selecting among rules induced from a hurricane database. In *Knowledge Discovery in Databases Workshop*, pages 28–44.

Michalski, R. S., Mozetic, I., Hong, J., & Lavrač, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the Fifth Annual National Conference on Artificial Intelligence*, volume 1510, pages 1041–1045.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill Series in Computer Science.

Monard, M. C., Batista, G. E. A. P. A., Kawamoto, S., & Pugliesi, J. B. (1997). Uma introdução ao aprendizado simbólico de máquina. Notas Didáticas do ICMC-USP n°29.

http://labic.icmc.sc.usp.br/didatico/PostScript/ML.ps.

Monard, M. C. & Horst, P. S. (1999a). The $\mathcal{RQ}_{system}$: description of the information processing module. Technical Report, ICMC-USP. Em andamento.

Monard, M. C. & Horst, P. S. (1999b). The $\mathcal{RQ}_{system}$: description of the pre-processing module. Technical Report, ICMC-USP. Em andamento.

Monard, M. C. & Nicoletti, M. C. (1993a). Programas prolog para processamnto de listas e aplicações. Notas Didáticas do ICMC-USP nº 7. http://labic.icmc.sc.usp.br/didatico/PostScript/listas-prolog.ps.

Monard, M. C. & Nicoletti, M. C. (1993b). Técnicas avançadas de programação prolog para tratamento de árvores. Notas Didáticas do ICMC-USP nº 8. http://labic.icmc.sc.usp.br/didatico/PostScript/arvores-prolog.ps.

Nagai, W. A. (1999). Modelos de previsão no processo de data mining. Exame de Qualificação de Mestrado, ICMC-USP.

Neto, J. L. (1999). TreeToRules system manual.

O'Keefe, R. A. (1990). *The Craft of Prolog.* The MIT Press.

Oliveira, R. B. T. (1999). O processo de extração de conhecimento da base de dados apoiado por agentes de software. Exame de Qualificação de Mestrado, ICMC-USP.

Piatetsky-Shapiro, G. (1991). Discovery, analysis and presentation of strong rules. In Piatetsky-Shapiro, G. & Frawley, W. J., editors, *Knowledge Discovery in Databases*, pages 229–248.

Piatetsky-Shapiro, G. & Matheus, C. (1994). The interestingness of deviations. In *Knowledge Discovery in Databases*, pages 25–36.

Pressman, R. S. (1997). *Software engineering: a practitioner's approach.* McGraw-Hill, New York.

Provost, F. J. & Aronis, J. M. (1996). Scaling up inductive learning with massive parallelism. *Machine Learning*, 23(1):33–46.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, pages 81–106.

Quinlan, J. R. (1987a). Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 304–307, Itália.

Quinlan, J. R. (1987b). Simplifying decision trees. *International Journal of Man-Machine Studies*, pages 221–234.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, Los Altos, CA, USA.

Raggett, J. & Bains, W. (1992). *Artificial Intelligence from A to Z.* Chapman & Hall.

Rocha, C. A. J. (1999). Redes bayesianas para extração de conhecimento de bases de dados, considerando a incorporação de conhecimento de fundo e o tratamento de dados incompletos. Dissertação de Mestrado, ICMC-USP.

Rocha, C. A. J., Padilha, T. P. P., Horst, P. S., & Rezende, S. O. (1997). Processo de desenvolvimento de sistemas baseados em conhecimento considerando aspectos de qualidade. In *Anais da SEPAI - Semana Paraense de Informática - SUCESU*, pages 113–120, Belém.

Ross, P. (1989). *Advanced Prolog: Techniques, and Examples.* Addison-Wesley.

Rowe, N. C. (1988). *Artificial through Prolog Prolog.* Prentice Hall.

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. http://www.cs.jhu.edu/~salzberg/critique.ps.

Silberschatz, A. & Tuzhilin, A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Trans. on Knowledge and Data Eng.*, 8(6):970–974.

Stanfill, C. & Waltz, D. (1986). Instance-based learning algorithms. *Communications of the ACM*, 8(12):1213–1228.

Sterling, L. & Shapiro, E. (1986). *The Art of Prolog.* MIT Press Series in Logic Programming.

Suzuki, E. & Kodratoff, Y. (1998). Discovery of surprising exception rules based on intensity of implication. In *Principles of Data Mining & Knowledge Discovery: Proceedings of the Second European Symp. Lecture Notes in Artificial Intelligence*, volume 1510, pages 10–18.

Taylor, C., Mitchie, D., & Spiegelhater, D. (1994). *Machine Learning, Neural and Statistical Classification.* Paramount Publishing International.

Torgo, L. (1993). Controlled redundancy in incremental rule learning. In *Proceedings of European Workshop on Machine Learning*, pages 185–195.

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, USA.

Westwood, D. (1997a). LPA-PROLOG for windows programming guide. Logic Programming Associates Ltd, Inglaterra.

Westwood, D. (1997b). LPA-PROLOG technical reference. Logic Programming Associates Ltd, Inglaterra.