Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Extração de Informações Padronizadas para a Avaliação de Regras Induzidas por Algoritmos de Aprendizado de Máquina Simbólico

Ronaldo Cristiano Prati
José Augusto Baranauskas
Maria Carolina Monard/ILTC

Nº 145

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Julho/2001

Extração de Informações Padronizadas para a Avaliação de Regras Induzidas por Algoritmos de Aprendizado de Máquina Simbólico*

Ronaldo Cristiano Prati
José Augusto Baranauskas
Maria Carolina Monard/ILTC

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
Laboratório de Inteligência Computacional
Caixa Postal 668, 13560-970 - São Carlos, SP, Brasil
e-mail: {prati, jaugusto, mcmonard}@icmc.sc.usp.br

Resumo

Dado um conjunto de exemplos (rotulados) de treinamento, é possível aplicar diversos algoritmos de Aprendizado de Máquina supervisionado, obtendo, para cada um desses algoritmos, uma descrição do conceito (classificador) que descreve o conhecimento implicito nesses exemplos de uma forma mais compacta. Em geral, é possível transformar um classificador, obtido através de um algoritmo de aprendizado simbólico, os quais descrevem o conhecimento induzido em uma forma diretamente interpretável por seres humanos, para um conjunto de regras.

Essas regras podem ser avaliados em conjunto, comparando o desempenho de cada classificador em relação a outros classificadores como uma "caixa preta", ou cada regra pode ser avaliada individualmente, quanto à qualidade, interessabilidade, novidade, entre outras medidas objetivas.

Para a avaliação das regras, são necessárias informações que permitam derivar facilmente medidas objetivas. Entretanto, somente alguns algoritmos de aprendizado apresentam essas informações, mas de uma forma não uniforme.

Neste trabalho é o proposta e implementada uma biblioteca de ferramentas que calcula um conjunto mínimo de informações para cada regra, de uma forma padronizada, para um conjunto de algoritmos de aprendizado de máquina supervisionado freqüentemente utilizados pela comunidade. Essa biblioteca está integrada a um sistema computacional de maior porte, que vem sendo desenvolvido em nosso Laboratório de Inteligência Computacional –LABIC –, para realizar, entre outros, extração automática e análise de conhecimento.

A biblioteca foi projetada de forma a admitir, futuramente, a inclusão de novos indutores.

Palavras-Chave: Aprendizado de Máquina, Avaliação de Regras.

Julho 2001

*Trabalho realizado com auxílio da FAPESP

Sumário

1 Introdução **1**

2 Avaliação de Regras Induzidas por Indutores Simbólicos **2**
2.1 Notação e Terminologia 2
2.1.1 Matriz de contingência 4
2.1.2 Formato dos Arquivos 5
2.2 Interpretação de um Conjunto de Regras 8
2.3 Atributos Com Valores Não Especificados 11
2.4 Medidas de Avaliação de Regras 12

3 Extensão do Formato Padrão de Regras **16**

4 A Biblioteca para Cálculo das Informações **18**

5 Exemplo **21**

6 Considerações Finais **27**

Lista de Figuras

1 Arquivo de Regras no Formato $\mathcal{PBM}$ 6
2 Gramática do Formato Padrão de Regras 7
3 O Arquivo `voyage.names` 8
4 Gramática do Arquivo de Nomes 9
5 O Arquivo `voyage.data` 10
6 Interpretação Geométrica para uma Árvore de Decisão 10
7 Interpretação Geométrica para um Conjunto de Regras Não Ordenadas 11
8 Arquivo de Regras no Formato $\mathcal{PBM}$ com Informações Padrão 17
9 Gramática do Formato Padrão de Regras Extendido 17
10 A Biblioteca para Cálculo das Informações Padrão 18
11 Regras não Ordenadas Geradas pelo Indutor $\mathcal{CN}2$ 23
12 Regras Avaliadas como Não Ordenadas 24
13 Regras Ordenadas Geradas pelo Indutor $\mathcal{CN}2$ 25
14 Regras Avaliadas como Ordenadas 26
15 Regras Ordenadas Entre as Classes Geradas pelo Indutor $\mathcal{C}4.5rules$ 26
16 Regras Avaliadas como Ordenadas Entre as Classes 27

Lista de Tabelas

1 Exemplo de um Conjunto de Dados 3
2 Cobertura de uma Regra $B \rightarrow H$ 4
3 Matriz de Contingência para uma Regra R 5
4 Matriz de Contingência com Freqüências Relativas para uma Regra R 5
5 Conjunto de Exemplos de Treinamento Voyage 22
6 Conjunto de Exemplos de Teste Voyage 22
7 Medidas Obtidas a Partir da Avalição das Regras Não Ordenadas 24
8 Medidas Obtidas a Partir da Avalição das Regras Ordenadas 25

9 Medidas Obtidas a Partir da Avalição das Regras Ordenadas Entre as Classes . . 27

Lista de Algoritmos

1 Cálculo das Informações Padrão para Regras Não Ordenadas 20
2 Cálculo das Informações Padrão para Regras Ordenadas 20
3 Cálculo das Informações Padrão para Regras Ordenadas Entre Classes 21

[1]Este documento foi elaborado com o LaTeX, sua bibliografia é mantida com o auxílio da ferramenta BibView (Prati, Baranauskas & Monard 1999) e gerada automaticamente pelo BibTeX

1 Introdução

> "Physicists ask what kind of place this universe is and seek to characterize its behavior systematically. Biologists ask what it means for a physical system to be living. We in AI wonder what kind of information-processing system can ask such questions."
>
> *Avron B. Barr e Edward A. Feigenbaum*
> *The Handbook of Artifical Inteligence*

Um sistema de Aprendizado de Máquina — AM — supervisionado é um programa (*indutor*) capaz de induzir uma descrição de um conceito a partir de um conjunto de exemplos conhecidos e previamente rotulados com as suas respectivas classes. Em outras palavras, o classificador é uma generalização dos exemplos fornecidos ao sistema. O objetivo principal de um classificador é, dado um novo exemplo cuja classe é desconhecida, predizer a sua classe.

Sistemas de AM, na prática, têm sido desenvolvidos utilizando diferentes paradigmas de aprendizado, tais como estatístico, conexionista, instance-based, genético e sistemas de aprendizado simbólico (Mitchell 1997). Em especial, sistemas de aprendizado simbólico são utilizados em situações em que os conceitos aprendidos precisam ser interpretados por humanos. O conhecimento induzido por algoritmos de AM simbólico é geralmente representado por árvores de decisão ou por um conjunto de regras.

Diversos pesquisadores implementaram algoritmos de AM simbólicos isoladamente e, normalmente, cada um desses algoritmos possui um formato próprio para os dados de entrada bem como para representar o conhecimento induzido.

Além disso, diferentes algoritmos de Aprendizado de Máquina podem ser aplicados à uma mesma amostra de dados e alguns desses algoritmos podem apresentar melhor desempenho que outros. Entretanto, para um dado domínio, não existem garantias que um determinado algoritmo seja necessariamente melhor que outro, ou seja, não existe uma análise matemática capaz de determinar qual algoritmo terá um melhor desempenho em relação aos outros (Kohavi, Sommerfield & Dougherty 1997). Dessa forma, estudos experimentais são necessários.

Para a avaliação, comparação e até mesmo a compreensão do conhecimento extraído a partir dos dados fornecidos, através da verificação das árvores ou regras induzidas, são necessárias informações que permitam derivar facilmente medidas objetivas de precisão, qualidade e interesse dessas regras.

Alguns indutores apresentam essas informações, mas de uma forma própria, como por exemplo em forma de porcentagem em relação à quantidade de exemplos cobertos ou mesmo como um número absoluto. Soma-se a isso o fato de que alguns indutores mostram essas informações separadas por cada classe existente, outros não, enquanto que outros indutores não apresentam nenhuma dessas informações.

Além da variedade de informação extra disponibilizada pelos diversos indutores, é importante notar que, geralmente, ela é decorrente da avaliação do classificador gerado no próprio conjunto de exemplos de treinamento. Dessa forma, os resultados são muito otimistas pois o maior interesse está em obter informações relacionadas com o comportamento futuro do classificador, isto é, utilizando novos exemplos não vistos pelo indutor durante a construção do classificador. Quando existem poucos exemplos para treinamento, essa é a técnica geralmente adotada. Entretanto, quando o número de exemplos cresce, como no caso de *Data Mining*,

no qual encontram-se disponíveis grandes volumes de dados, é possível realizar uma avaliação menos tendenciosa utilizando uma amostra diferente daquela utilizada para treinamento.

Este trabalho propõe o projeto e a implementação de uma biblioteca de programas (ou ferramentas) que, a partir de um conjunto de regras e um conjunto de exemplos (de teste), calcula um conjunto mínimo de informações adicionais para cada regra, de uma forma padronizada. A partir desse conjunto mínimo de informações, medidas de qualidade e interessabilidade de regras podem ser facilmente obtidas.

Essa biblioteca de ferramentas foi desenvolvida utilizando a linguagem de programação PERL (PERL 1999) e está integrada a um sistema computacional de maior porte que está sendo desenvolvido no nosso laboratório de pesquisa — LABIC [1] — para realizar, entre outros, extração automática e análise de conhecimento.

Quanto à organização, este trabalho está dividido da seguinte forma: na Seção 2 é feita uma revisão sobre os principais pontos a serem considerados na avaliação de regras, além de apresentar algumas medidas utilizadas na avaliação de regras. Na Seção 3 é mostrado o formato padrão para as informações adicionais, por nós definido. Na Seção 4 são mostrados os detalhes da implementação da biblioteca para o cálculo das informações e na Seção 6 são apresentadas algumas considerações finais.

2 Avaliação de Regras Induzidas por Indutores Simbólicos

"Works of art make rules, but rules do not make works of art."

Debussy

O crescente uso e a diversidade de aplicações, tanto de algoritmos de Aprendizado de Máquina quanto o uso desses algoritmos na área de *Data Mining*, cria a necessidade de novas formas de avaliação/validação do conhecimento induzido (Weiss & Indurkhya 1998; Fayyad, Piatetsky-Shapiro, Smyth & Uthurusamy 1996). Essas formas de avaliação incluem medidas objetivas (obtidas a partir de diversas métricas propostas na literatura) e medidas subjetivas (obtidas através da inspeção do conhecimento induzido). Essas medidas fornecem informações sobre, entre outros, a precisão, qualidade e interessabilidade do conhecimento induzido (Horst 1999; Lavrač, Flach & Zupan 1999; Freitas 1998a; Freitas 1998b).

Neste trabalho nos focalizamos no projeto e implementação de medidas objetivas de regras obtidas a partir do conhecimento induzido por algoritmos de AM simbólicos.

2.1 Notação e Terminologia

No Aprendizado de Máquina supervisionado é dado ao algoritmo de aprendizado um conjunto de casos (*exemplos, objetos ou dados*) contendo n exemplos classificados (rotulados) segundo uma classe de interesse. Cada exemplo é composto por m atributos. Na Tabela 2.1 é mostrado um conjunto de dados T com n exemplos, cada exemplo com m atributos. Uma linha i se refere ao i-ésimo exemplo ($i = 1, 2, \ldots, n$) e uma entrada x_{ij} se refere ao valor do j-ésimo ($j = 1, 2, \ldots, m$) atributo X_j do exemplo i.

[1] `http:\\labic.icmc.sc.usp.br`

	X_1	X_2	$\ldots$	X_m	Y
T_1	x_{11}	x_{12}	$\ldots$	x_{1m}	y_1
T_2	x_{21}	x_{22}	$\ldots$	x_{2m}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_n	x_{n1}	x_{n2}	$\ldots$	x_{nm}	y_n

Tabela 1: Exemplo de um Conjunto de Dados

Exemplos são pares $T_i = (x_{i1}, x_{i2}, \ldots, x_{im}, y_i) = (\mathbf{x}_i, y_i)$, e o conjunto de exemplos é referenciadas como $(\mathbf{X}, Y)$, onde a última coluna, Y, é um atributo especial, denominado *classe* (ou *rótulo*), que desejamos predizer com base nos outros $\mathbf{X}$ atributos, isto é, $Y = f(\mathbf{X})$. Cada $\mathbf{x}_i$ é um elemento do conjunto $X_1 \times X_2 \times \ldots \times X_m$ onde X_j é o domínio do j-ésimo atributo e y_i pertence a uma das k classes, isto é, $Y \in \{C_1, C_2, \ldots, C_k\}$.

Os conceitos aprendidos por algoritmos de AM simbólicos são geralmente representados por árvores de decisão ou conjuntos de regras. Como sempre é possível escrever uma árvore de decisão como um conjunto de regras disjuntas, deste ponto em diante o termo *regra* refere-se a uma regra extraída de uma árvore de decisão (regras disjuntas) ou uma regra diretamente induzida por um algoritmo de AM supervisionado.

Uma regra é geralmente representada na forma

$$\text{R: } \textit{if} <\text{condição}> \textit{then} <\text{class} = C_i>$$

onde <condição> é uma disjunção de conjunções de testes para os atributos da forma

$$X_i \ \textit{op} \text{ valor}$$

e C_i é um dos possíveis valores para a classe.

Para facilitar a leitura, adotaremos uma representação mais genérica para qualquer regra R, onde

$$\text{R: } \underbrace{\textit{if} \ < \text{condição} >}_{\textit{Body} \text{ ou } B} \textit{then} \ \underbrace{< \text{class} = C_i >}_{\textit{Head} \text{ ou } H}$$

passando a denotar uma regra como

$$Body \rightarrow Head$$

ou resumidamente $B \rightarrow H$.

Dizemos que um exemplo T_i é coberto por uma regra R se e somente se o exemplo satisfaz todas as condições da regra (todas as condições de uma regra são avaliadas como verdadeiras, dada a descrição do exemplo). Ou seja, um exemplo T_i é coberto por uma regra R se e somente se B é verdade. Por outro lado, um exemplo que não satisfaz a condição B da regra não é coberto pela regra.

Dizemos que um exemplo T_i é corretamente coberto por uma regra R se e somente se T_i é coberto pela regra e a classe y_i do exemplo é a mesma prevista pela regra. Ou seja, um

exemplo T_i é corretamente coberto pela regra R se e somente se B é verdade e H é verdade. Entretanto, se o exemplo satisfaz a condição B da regra mas não satisfaz a condição H, o exemplo é incorretamente coberto pela regra. Um resumo das quatro possíveis situações pode ser visto na Tabela 2.

Exemplos satisfazendo ...	são ...
B	cobertos pela regra
$\overline{B}$	não cobertos pela regra
$B \wedge H$	cobertos corretamente pela regra
$B \wedge \overline{H}$	cobertos incorretamente pela regra

Tabela 2: Cobertura de uma Regra $B \rightarrow H$

2.1.1 Matriz de contingência

A matriz de contingência é uma generalização da matriz de confusão, que é a base padrão para calcular medidas de avaliação de hipóteses em problemas de classificação. A matriz de confusão é aplicada ao classificador como um todo, ou seja, o classificador é tratado como uma caixa preta. Já a matriz de contingência é calculada para cada regra, exigindo dessa forma, um classificador simbólico.

Dados uma regra R, e um exemplo $T_i = (\mathbf{x}_i, y_i)$ com a sua respectiva classe y_i, podemos aplicar a regra ao exemplo e comparar o resultado previsto pela cabeça H da regra com a verdadeira classe y_i do exemplo. Essa comparação resulta em quatro possíveis situações:

1. O exemplo é coberto corretamente pela regra, ou seja, B e H são verdade.
2. O exemplo é incorretamente coberto pela regra, ou seja, B é verdade mas H é falso.
3. O exemplo não é coberto pela regra mas a classe prevista pela cabeça H da regra é a mesma classe y_i do exemplo, ou seja, B é falso mas H é verdade.
4. O exemplo não é coberto pela regra e a classe prevista pela cabeça H da regra não é a mesma classe y_i do exemplo, ou seja, B é falso e H também é falso.

Aplicando a regra a um conjunto de teste T que contenha n exemplos, podemos derivar, para cada regra, a sua matriz de contingência (Tabela 3). A matriz de contingência também pode ser representada usando freqüências relativas, como mostrado na Tabela 4, onde os valores presentes na tabela de contingência foram divididos por n, ou seja $f_\epsilon = \frac{\epsilon}{n}$. Neste trabalho, assumiremos a freqüência relativa $\frac{\epsilon}{n}$, associada ao evento ϵ, como uma estimativa de probabilidade para o evento ϵ, e denotaremos como $P(\epsilon)$.

Usando como base a matriz de contingência, é possível definir a maioria das medidas propostas na literatura para avaliação de um conjunto de regras (Lavrač, Flach & Zupan 1999). Na Seção 2.4 na página 12 são apresentadas algumas dessas medidas derivadas da matriz de contingência.

	H	$\overline{H}$	
B	bh	$b\overline{h}$	b
$\overline{B}$	$\overline{b}h$	$\overline{bh}$	$\overline{b}$
	h	$\overline{h}$	n

bh é o número de exemplos para os quais B é verdade e H é verdade.
$b\overline{h}$ é o número de exemplos para os quais B é verdade e H é falso.
$\overline{b}h$ é o número de exemplos para os quais B é falso, mas H é verdade.
$\overline{bh}$ é o número de exemplos para os quais B é falso e H é falso.
b é o número total de exemplos para os quais B é verdade.
$\overline{b}$ é o número total de exemplos para os quais B é falso.
h é o número total de exemplos para os quais H é verdade.
$\overline{h}$ é o número total de exemplos para os quais H é falso.
n é o número total de exemplos.

Tabela 3: Matriz de Contingência para uma Regra R

	H	$\overline{H}$	
B	f_{bh}	$f_{b\overline{h}}$	f_b
$\overline{B}$	$f_{\overline{b}h}$	$f_{\overline{bh}}$	$f_{\overline{b}}$
	f_h	$f_{\overline{h}}$	1

Tabela 4: Matriz de Contingência com Freqüências Relativas para uma Regra R

2.1.2 Formato dos Arquivos

Como já foi dito anteriormente, a biblioteca para o cálculo das informações implementada neste trabalho faz parte de um sistema computacional de maior porte, o sistema Discover, que vem sendo desenvolvido em nosso laboratório de pesquisa (Baranauskas & Batista 2000). O Discover visa, entre outros, integrar alguns dos algoritmos de aprendizado simbólico mais conhecidos em uma única ferramenta, facilitando, entre outros, a realização de estudos comparativos entre esses algoritmos.

Contudo, esses algoritmos de aprendizado possuem diferentes sintaxes para seus arquivos, tanto dos arquivos de exemplos, quanto dos arquivos de saída que contém a descrição do conceito induzido pelo algoritmo (classificador). Para facilitar a integração desses algoritmos, foram propostas sintaxes padrões para esses arquivos, bem como a implementação de ferramentas que fazem a conversão entre os formatos originais desses arquivos para a sintaxe padrão aceita pelo Discover.

Nesta seção são descritos o formato padrão proposto para o arquivo contendo as regras, ou seja, a descrição do conceito induzido por um algoritmo de aprendizado simbólico, e para o arquivo de exemplos. Esses formatos são aceitos como padrão pelo Discover.

Formato Padrão de Regras

O arquivo de regras, induzidas por um algoritmo de aprendizado simbólico, é composto por um conjunto de regras do tipo $B \rightarrow H$, as quais são transformadas para o formato padrão de

regras $\mathcal{PBM}$, definido como padrão para o DISCOVER.

Em (Prati, Baranauskas & Monard 2001) é descrita o formato padrão $\mathcal{PBM}$ com maiores detalhes, além de uma biblioteca de *scripts* que convertem a linguagem de representação de conceitos dos principais algoritmos de AM simbólico — $\mathcal{ID}3$ (Quinlan 1986), $\mathcal{C}4.5$ (Quinlan 1988), $\mathcal{C}4.5rules$ (Quinlan 1988), $\mathcal{C}5.0$ /$\mathcal{S}$ee5 [2], $\mathcal{CN}2$ (Clark & Niblett 1987; Clark & Boswell 1989; Clark & Boswell 1991), $\mathcal{OC}1$ (Murthy, Kasif & Salzberg 1994), $\mathcal{R}ipper$ (Cohen 1995), $\mathcal{T}2$ e $\mathcal{MC}4$ — para o formato $\mathcal{PBM}$. A Figura 1 mostra o arquivo gerado com as regras no formato $\mathcal{PBM}$, cuja sintaxe é definida pela gramática mostrada na Figura 2[3].

```
Standard Rules Conversor<version> Copyright (c) Ronaldo C. Prati
Inducer: <Inducer Name>           Input File: <Input File Name>
Date: <Date>

R0001 IF <condition>
      THEN CLASS = <Ci>

R0002 IF <condition>
      THEN CLASS = <Ci>

...
```

Figura 1: Arquivo de Regras no Formato $\mathcal{PBM}$

Formato Padrão de Exemplos

A sintaxe padrão proposta para os exemplos (Batista 2001) utiliza arquivos do tipo texto para declarar os nomes dos atributos (e seus respectivos domínios) e, quando for o caso, os valores que esses atributos assumem no conjunto de exemplos. Os nomes dos atributos são declarados em um arquivo de nomes, designados com a extensão `.names`. Os valores que esses atributos assumem em um conjunto de exemplos são declarados no arquivo de dados, designados com a extensão `.data`. Um conjunto de exemplos somente está na sintaxe padrão do DISCOVER se esses dois arquivos estiverem presentes. Os dois arquivos devem ter o mesmo nome, se diferenciando apenas pela extensão.

A sintaxe dos arquivos de nomes (`.names`) e dados (`.data`) é uma extensão do formato utilizado pelo $\mathcal{C}4.5$ (Quinlan 1988). Essa extensão tem como principal objetivo adequar o formato dos arquivos com as necessidades do projeto DISCOVER.

A primeira declaração em um arquivo de nomes define qual deve ser o nome do atributo classe. O atributo classe pode ser qualquer atributo presente no conjunto de exemplos. Após a declaração do atributo classe, são declarados os demais atributos. Cada atributo possui um identificador e um domínio a ele associado. São válidos os identificadores que são combinações de números, letras e "_" (sublinhado), em qualquer seqüência. Para identificadores mais complexos que envolvem outros caracteres que não sejam os especificados anteriormente (como espaços,

[2] http://www.rulequest.com

[3] O uso de termos em inglês em figuras, cabeçalhos de arquivos e definições deve-se ao fato que o sistema DISCOVER, que vem sendo implementado em nosso laboratório estará disponível, através da Internet, para uso por instituições no mundo todo. Dessa forma, convencionou-se o uso desses termos no idioma inglês.

```
S ::= <rule>

<rule> ::= IF <complex> THEN CLASS = <class>

<complex> ::= <fator>
        | <fator> OR <condition>

<fator> ::= <term>
        | <term> AND <fator>

<term> ::= <comparison>
        | <linear combination>

<comparison> ::= <atribute><operator><value>

<linear combination> ::= <combination><operator><value>

<combination> ::= <constant> x <numeric attribute>
        | <constant> x <numeric attribute> + <combination>

<operator> ::= <=,>=,in,<,>,!=,=

<class> ::= CLASS = <value>
```

Figura 2: Gramática do Formato Padrão de Regras

letras acentuadas, etc) é necessário colocar o identificador entre aspas. Desta forma, são identificadores válidos: abc, 1, 1a, _1a, " _12a ", "válido".

Na Figuara 3 é mostrado o arquivo de nomes para o arquivo de exemplos `voyage` (descrito na Seção 5 na página 21) e na Figura 4 na página 9 é mostrada a gramática para o arquivo de nomes. Os atributos podem assumir qualquer um dos seguintes tipos de dados:

Nominal: O tipo de dado `nominal` é utilizado para declarar um atributo discreto, ou seja que pode assumir um conjunto finito de valores.

Enumerated: O tipo de dado `enumerated` é muito semelhante ao tipo de dado `nominal`. A principal diferença é que com o tipo `enumerated` é possível identificar uma ordem entre os valores que o atributo pode assumir. Entretanto, não existe uma definição clara de distância entre esses valores. Um exemplo de tipo `enumerated` é um atributo que pode assumir, por exemplo, os valores *pequeno*, *médio* e *grande*.

Integer: O tipo de dado `integer` é utilizado para declarar um atributo que pode assumir valores inteiros.

Real: O tipo de dado `real` é semelhante ao tipo de dado `integer`, com a diferença que um atributo `real` pode assumir valores com ou sem parte fracionária.

String: Um atributo `string` pode assumir como valor um string de tamanho indefinido o qual pode conter quaisquer caracteres incluindo quebra de linha (**n**). Para identificar os limites de um string é necessário inserir o símbolo de aspas (") antes e depois do string.

Date: O tipo de dado `date` permite declarar um atributo que pode conter uma data (dia, mês e ano). A princípio, os valores das datas devem estar no formato "aaaa/mm/dd", utilizado pela maioria dos Gerenciadores de Bases de Dados.

Time: O tipo de dado time permite declarar um atributo que pode conter um horário (hora, minuto e segundo). A princípio, os valores dos horários devem estar no formato "hh:mm:ss".

```
voyage.

outlook:     sunny, overcast, rain.
temperature: continuous.
humidity:    continuous.
windy:       true, false.
voyage:      go, dont_go.
```

Figura 3: O Arquivo `voyage.names`

O arquivo de dados correspondente ao arquivo de nomes contém n exemplos, um exemplo em cada linha. Cada exemplo i consiste nos valores dos atributos desse exemplo, separados por vírgula. Os valores de cada atributo devem pertencer ao domínio especificado para esse atributo no arquivo de nomes. Desta forma, o "separador de registros" (*record separator*) é o caracter de nova linha (representado em muitas linguagem de programação por **\n**) e o separador de campos (*field separator*) é a vírgula (,). Cada valor presente em uma linha está associado a um atributo do arquivo de nomes. Sendo assim, a ordem em que os valores são declarados em uma determinada linha deve ser a mesma na qual os atributos foram declarados no arquivo de nomes. Valores "*desconhecidos*" são representados por um ponto de interrogação (?) e valores "*não-se-aplica*" são representados com um ponto de exclamação (!) (ver Seção 2.3 na página 11). Na Figura 5 é mostrado o arquivo de exemplos `voyage`.

2.2 Interpretação de um Conjunto de Regras

Árvores de Decisão dividem o espaço de descrição do problema em regiões disjuntas, isto é, cada exemplo é classificado por apenas um único ramo da árvore. As regras obtidas pela reescrita da árvore como um conjunto de regras preserva essa propriedade. Dessa forma, ao aplicarmos um exemplo a um conjunto de regras obtidas pela reescrita de uma árvore de decisão, uma única regra cobrirá aquele exemplo. O espaço de descrição também é divido de uma forma completa, isto é, cada ponto no espaço é atribuído a uma região. Na Figura 6 na página 10 é mostrada uma interpretação geométrica para uma árvore de decisão de duas classes (+ e o) e dois atributos (X_1 e X_2).

Um ponto importante a ser considerado na avaliação de um conjunto de regras é a forma em que essas regras foram induzidas. Alguns algoritmos de AM que induzem regras diretamente dos dados criam um conjunto ordenado de regras, outros algoritmos criam um conjunto não ordenado de regras e alguns algoritmos ambos tipos de regras.

Em conjuntos não ordenados, podemos aplicar o exemplo a todas as regras para calcular os valores da matriz de contingência, uma vez que a ordem de aplicação das regras não é importante.

```
S ::= <class-defs>
        | <feature-defs>

<class-defs> ::= <feature-name>.
        | null.

<feature-name> ::= <identifier>

<feature-defs> ::= <feature-name> : <feature-type> .
        | <feature-name> : <feature-type> : <extended-defs> .
        | <feature-name> : <feature-type> := <expression> : <extended-defs> .

<feature-type> ::= real
        | integer
        | boolean
        | nominal
        | nominal (<list>)
        | enumerated (<list>)
        | date
        | time
        | string

<extended-defs> ::= <extended-def>
        | <extended-def> : <extended-defs>

<extended-def> ::= <identifier>
        | <identifier> (<list>)

<list> ::= <identifier>
        | <identifier> , <list>
```

Figura 4: Gramática do Arquivo de Nomes

```
sunny,    25, 72, yes, go
sunny,    28, 91, yes, dont_go
sunny,    22, 70, no,  go
sunny,    23, 95, no,  dont_go
sunny,    30, 85, no,  dont_go
overcast, 23, 90, yes, go
overcast, 29, 78, no,  go
overcast, 19, 65, yes, dont_go
overcast, 26, 75, no,  go
overcast, 20, 87, yes, go
rain,     22, 95, no,  go
rain,     19, 70, yes, dont_go
rain,     23, 80, yes, dont_go
rain,     25, 81, no,  go
rain,     21, 80, no,  go
```

Figura 5: O Arquivo `voyage.data`

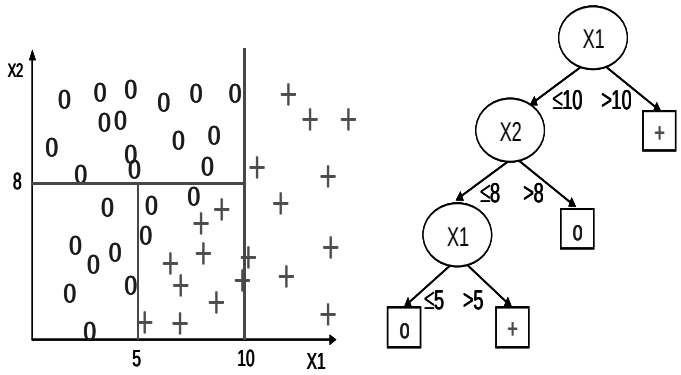

Figura 6: Interpretação Geométrica para uma Árvore de Decisão

Como as regras obtidas pela reescrita de uma árvore de decisão são disjuntas, elas podem ser tratadas como regras não ordenadas, pois somente uma regra cobre o exemplo.

Já para conjuntos ordenados, em que a ordem de aplicação das regras é importante, o exemplo deve ser avalido pelo conjunto de regras, começando pela primeira até encontrarmos uma regra que cubra o exemplo. O exemplo deve ser classificado exclusivamente por essa regra, mesmo que as regras seguintes também cubram isoladamente o exemplo.

Quando um algoritmo de AM induz um conjunto regras não ordenadas a partir dos exemplos, as regras podem definir regiões sobrepostas no espaço de descrição. Na classificação de um exemplo, cada algoritmo deve definir a forma para classificar o exemplo quando mais de uma regra cobri-lo. Pode também acontecer que o espaço de descrição não seja completamente coberto pelo conjunto de regras. Para esses casos, os algoritmos de AM geralmente criam uma regra padrão, atribuindo ao exemplo que não é coberto por nenhuma das regras a classe com o maior número de exemplos no conjunto de dados de treinamento, ou a classe que foi menos coberta

por todas as outras regras. Na Figura 7 é mostrada uma interpretação geométrica para um conjunto de quatro regras não ordenadas.

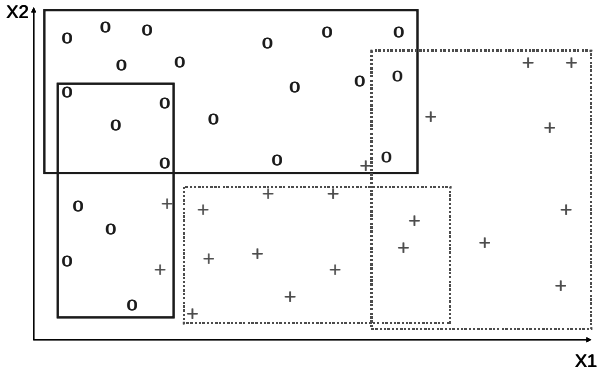

Figura 7: Interpretação Geométrica para um Conjunto de Regras Não Ordenadas

Um outro exemplo interessante são as regras induzidas pelo algoritmo de AM $\mathcal{C}4.5rules$. O $\mathcal{C}4.5rules$ utiliza uma árvore de decisão induzida pelo $\mathcal{C}4.5$ e então deriva um conjunto de regras. As regras são agrupadas por classe e são não ordenadas dentro de uma mesma classe, mas são ordenadas entre as classes. É muito importante notar que o $\mathcal{C}4.5rules$ não reescreve simplesmente a árvore de decisão para um conjunto de regras. Na verdade, ele generaliza as regras desconsiderando condições supérfluas.

2.3 Atributos Com Valores Não Especificados

Para verificar se uma regra cobre um dado exemplo, devemos comparar cada um dos testes de atributos presentes na regra com os seus respectivos valores no exemplo. Como cada teste é baseado em um único atributo, o resultado da classificação não pode ser determinado se o valor para aquele atributo não está especificado no exemplo.

Entretanto, em aplicações do mundo real, é comum encontrar casos em que alguns valores dos atributos, para um determinado exemplo, não estão especificados. Esses valores não especificados podem ser originados de duas situações:

1. Desconhecido (?) – o valor do atributo não é conhecido para aquele exemplo. Deve ser observado que o valor desconhecido é bem diferente, por exemplo, do valor zero para números ou de strings vazias.

2. Não-se-aplica (!) – o valor do atributo não é aplicável para aquele exemplo. Por exemplo, o atributo número de gestações não se aplica quando o paciente é do sexo masculino.

Quando estamos fazendo a avaliação em conjuntos de dados que contenham atributos com valores não especificados, temos duas alternativas a escolher:

1. Descartar do conjunto de dados todos os exemplos em que aparecem valores não especificados. Dessa forma, os exemplos que contenham valores não especificados não seriam utilizados no processo de avaliação de regras.

2. Aplicar, durante a avaliação de cada regra, alguma técnica para tratar os exemplos em que aparecem valores não especificados, ou seja, estabelecer como o sistema deve proceder se o exemplo a ser classificado tem um valor não especificado para o atributo que está sendo testado.

Muitos autores propuseram diferentes técnicas para a segunda alternativa. Algumas dessas técnicas, utilizadas para árvores de decisão, sumarizadas em (Quinlan 1990), são descritas a seguir:

- Se existir um teste especial para atributos desconhecidos, seguir esse ramo da árvore.
- Avaliar o teste como verdadeiro ou falso, sempre que for encontrado um valor não especificado.
- Usar o valor mais provável para o atributo no teste, segundo uma abordagem estatística, e tomar uma decisão de acordo com esse valor.
- Assumir para este atributo o valor mais comum no conjunto de dados para atributos discretos ou, para atributos contínuos, a média ou mediana.
- Explorar todas as possibilidades, combinando os resultados para refletir as diferentes probabilidades de diferentes resultados.

Em geral, a escolha de alguma dessas técnicas é dependente do domínio do problema que está sendo analisado. A escolha da primeira, em especial, depende do algoritmo de AM criar um ramo especial na árvore de decisão para o valor desconhecido. Isto acontece quando o algoritmo de AM não tem a capacidade de trabalhar com valores desconhecidos durante a indução da hipótese.

2.4 Medidas de Avaliação de Regras

Várias medidas já foram pesquisadas com a finalidade de auxiliar o usuário no entendimento e utilização do conhecimento adquirido por sistemas de AM. As medidas apresentadas nesta seção foram compiladas em Lavrač, Flach & Zupan (1999) e utilizam como base a matriz de contingência com freqüências relativas, definida na Seção 2.1.1 na página 4.

Precisão (1): A precisão (*consistência* ou *confidência*) é uma medida do quanto uma regra é específica para o problema. A precisão pode ser definida como a probabilidade condicional de H ser verdade dado que B é verdade. Quanto maior, mais precisamente a regra cobre a classe em questão.

$$Acc(R) = P(H|B) = \frac{P(HB)}{P(B)} = \frac{f_{hb}}{f_b} \tag{1}$$

Erro (2): O erro de uma regra é definido como $1 - Acc(R)$. Quanto maior o erro, menos precisamente a regra cobre a classe em questão.

$$Err(R) = 1 - Acc(r) = P(\overline{H}|B) = \frac{f_{\overline{h}b}}{f_b} \tag{2}$$

Confiança Negativa (3): é o correspondente à precisão, mas para os exemplos que não são cobertos pela regra. É definida como a probabilidade condicional de H ser falso dado que B também é falso.

$$NegRel(R) = P(\overline{H}|\overline{B}) = \frac{P(\overline{HB})}{P(\overline{B})} = \frac{f_{\overline{hb}}}{f_{\overline{b}}} \tag{3}$$

Sensitividade (4): Sensitividade (*completeza* ou *recall*) é uma medida do número (relativo) de exemplos da classe prevista em H cobertoe pela regra. É definida como a probabilidade condicional de B ser verdade dado que H é verdade. Quanto maior a sensitividade, mais exemplos são cobertos pela regra.

$$Sens(R) = P(B|H) = \frac{P(HB)}{P(H)} = \frac{f_{hb}}{f_h} \tag{4}$$

Especificidade (5): é o correspondente à completeza, mas para os exemplos que não são cobertos pela regra R. É definida como a probabilidade condicional de B ser falso dado que H é falso.

$$Spec(R) = P(\overline{B}|\overline{H}) = \frac{P(\overline{HB})}{P(\overline{H})} = \frac{f_{\overline{hb}}}{f_{\overline{h}}} \tag{5}$$

Cobertura (6): Cobertura é uma medida do número (relativo) de exemplos cobertos pela regra R. É definida como a probabilidade de B ser verdade. Quanto maior a cobertura, maior o número de exemplos cobertos pela regra R.

$$Cov(R) = P(B) = f_b \tag{6}$$

Suporte (7): Suporte (*freqüência*) é uma medida do número (relativo) de exemplos cobertos corretamente pela regra R. É definido como a probabilidade de H e B serem verdade. Quanto maior o suporte, maior o número de exemplos da classe em questão que são cobertos corretamente pela regra R.

$$Sup(R) = P(HB) = f_{hb} \tag{7}$$

Novidade (8): Novidade pode ser definida como se a probabilidade de B e H ocorrerem juntos não puder ser inferida pelas probabilidades de B e H isoladamente, isto é, B e H não são estatisticamente independentes. A medida de novidade é obtida comparando o valor esperado $P(HB)$ com os valores de $P(H)$ e $P(B)$. Quanto mais o valor esperado diferir do observado, maior é a probabilidade que exista um correlação verdadeira e inesperada entre B e H. Pode ser demonstrado que $-0,25 \leq Nov(R) \leq 0,25$, e quanto maior um valor positivo (mais próximo de 0,25) mais forte é a associação entre B e H enquanto que, quanto maior um valor negativo (mais próximo de -0,25), mais forte é a associação entre B e $\overline{H}$.

$$Nov(R) = P(HB) - P(H)P(B) = f_{hb} - f_h \cdot f_b \tag{8}$$

Satisfação (9): Satisfação é o aumento relativo na precisão entre a regra $B \rightarrow verdade$ e a regra $B \rightarrow H$. É uma medida mais indicada para tarefas voltadas à descoberta de conhecimento, sendo capaz de promover um equilíbrio entre regras com diferentes condições e conclusões.

$$Sat(R) = \frac{P(\overline{H}) - P(\overline{H}|B)}{P\overline{H}} = \frac{f_{\overline{h}} - \frac{f_{\overline{hb}}}{f_b}}{f_{\overline{h}}} \tag{9}$$

Precisão Relativa (10): A precisão relativa de uma regra mede o ganho de precisão obtido em relação à precisão de uma regra padrão $verdade \rightarrow H$, ou seja, que avalia B como verdade para todos os exemplos. Nesse caso, uma regra só interessa se melhorar a precisão da regra padrão.

$$RAcc(R) = P(H|B) - P(H) = \frac{f_{hb}}{f_b} - f_h \tag{10}$$

Confiança Negativa Relativa (11): É o análogo a precisão relativa para os exemplos que não são cobertos pela regra. Nesse caso, a regra padrão é $falso \rightarrow \overline{H}$.

$$RNegRel(R) = P(\overline{H}|\overline{B}) - P(\overline{H}) = \frac{f_{\overline{hb}}}{f_{\overline{b}}} - f_{\overline{h}} \tag{11}$$

Sensitividade Relativa (12): A sensitividade relativa mede o ganho de sensitividade obtido em relação à sensitividade de uma regra padrão $B \rightarrow verdade$, ou seja, uma regra que avalia H como verdade para todos os exemplos.

$$RAcc(R) = P(B|H) - P(B) = \frac{f_{hb}}{f_h} - f_b \tag{12}$$

Especificidade Relativa (13): É o análogo a sensitividade relativa para os exemplos que não são cobertos pela regra. Nesse caso, a regra padrão é $\overline{B} \rightarrow falso$.

$$RSpec(R) = P(\overline{B}|\overline{H}) - P(\overline{B}) = \frac{f_{\overline{hb}}}{f_{\overline{h}}} - f_{\overline{b}} \tag{13}$$

Um ponto importante referente as medidas relativas (medidas 10, 11, 12 e 13) é que elas dão mais informação sobre a utilidade de uma regra que as informações fornecidas pelas suas respectivas medidas absolutas (medidas 1, 3, 4 e 5). Por exemplo, se a precisão de uma regra é menor que a freqüência relativa da classe que a regra prediz, então a regra tem um desempenho ruim, independentemente de sua precisão absoluta.

Existe, no entanto, um problema com a precisão relativa, pois é fácil obter uma alta precisão relativa para regras muito específicas, ou seja, regras com uma baixa generalidade de $P(B)$.

Para contornar esse problema, é proposto em Lavrač, Flach & Zupan (1999) uma variante das medidas relativas, na qual é atribuído um peso para cada uma dessas medidas. Este peso promove uma balanceamento entre a generalidade e a relatividade dessas medidas.

Dessa forma, a medida $RAcc(R)$ é multiplica pelo seu "coeficiente de peso", $P(B)$, obtendo a nova medida Precisão Relativa com Peso (14), também conhecida na literatura como ganho.

É possível mostrar que essa medida é equivalente a $Nov(R)$, ou seja, regras com alta precisão relativa também tem alta novidade e vice-versa.

$$WRAcc(R) = P(B)(P(H|B) - P(H)) = f_b\left(\frac{f_{hb}}{f_b} - f_h\right) \quad (14)$$

$$WRAcc(R) \equiv Nov(R) \quad (15)$$

Analogamente, as medidas Confiança Negativa (16), Sensitividade (17) e Especificidade (19) Relativas com Peso são obtidas multiplicando as correspondentes medidas relativas, $RNegRel(R)$, $RSens(R)$ e $RSpec(R)$, pelos coeficientes de pesos $P(\overline{B})$, $P(H)$ e $P(\overline{H})$, respectivamente. Lavrač também mostra que essas medidas são equivalentes entre si, ressaltando a importância da precisão relativa com peso como medida fundamental para a avaliação de regras, promovendo um balanceamento entre precisão e as outras medidas.

$$WRNegRel(R) = P((\overline{B}))P(\overline{H}|\overline{B}) - P(\overline{H}) = f_{\overline{b}}\left(\frac{f_{\overline{hb}}}{f_{\overline{b}}} - f_{\overline{h}}\right) \quad (16)$$

$$WRSens(R) = P(H)(P(B|H) - P(B)) = f_h\left(\frac{f_{hb}}{f_h} - f_b\right) \quad (17)$$

$$WRSpec(R) = P((\overline{H}))P(\overline{B}|\overline{H}) - P(\overline{B}) = f_{\overline{h}}\left(\frac{f_{\overline{hb}}}{f_{\overline{h}}} - f_{\overline{b}}\right) \quad (18)$$

$$WRAcc(R) \equiv WRNegRel(R) \equiv WRSens(R) \equiv WRSpec(R) \quad (19)$$

Muitas outras medidas para avaliação de regras podem ser encontradas na literatura. Por exemplo, em Horst (1999) são descritas outras medidas de qualidade e interessabilidade de regras. Bayardo & Agrawal (1999) apresentam em seu artigo medidas usadas com ordenação de regras de associação. Freitas (1998a) e Freitas (1998b) faz o uso de medidas objetivas para avaliar qualidade e surpresa de regras. An & Cercone (1999) fazem uso de medidas derivadas empiricamente na avaliação de qualidade de regras. Em geral, o número de nós presentes em uma árvores ou o número de testes presentes no corpo de uma regra são usados, normalmente, como medida de compreensibilidade de regras.

As medidas apresentadas neste trabalho, bem como outras existentes na literatura, podem ser combinadas ou avaliadas separadamente, para auxiliar o usuário no entendimento e utilização do conhecimento adquirido por sistemas de AM. Também vale lembrar que, mesmo que a precisão global do classificador não seja considerada boa, podem existir algumas regras boas em termos de qualidade, novidade, interessabilidade, etc. Este aspecto é muito importante na área de descoberta de conhecimento (KDD — Knowledge Data Discovery) (Fayyad, Piatetsky-Shapiro, Smyth & Uthurusamy 1996).

3 Extensão do Formato Padrão de Regras

"All we know is infinitely less than all that still remains unknow."

Willian Harvey

Como já mencionado, um dos objetivos do nosso trabalho é fornecer ao usuário ou pesquisador, de uma forma padronizada, um conjunto de informações para cada regra, sobre o qual se possa derivar facilmente as métricas citadas neste relatório, além de outras medidas.

Todas as medidas aqui apresentadas, bem como a maioria das medidas objetivas de regras encontradas na literatura, podem ser facilmente derivadas a partir da matriz de contingência. Assim, o formato padrão de regras, descrito na Seção 2.1.2 na página 5, foi estendido, contendo agora também as freqüências relativas f_{bh}, $f_{b\overline{h}}$, $f_{\overline{bh}}$, e $f_{\overline{b}h}$, além do número de exemplos, n, utilizado na medição. As freqüências marginais, f_b, $f_{\overline{b}}$, f_h e $f_{\overline{h}}$ podem ser facilmente obtidas pela soma das linhas e colunas da matriz de contingência (Tabela 4 na página 5). Os valores absolutos das medidas também podem ser facilmente obtidos multiplicando as freqüências relativas por n. O formato padrão proposto para as informações adicionais contém agora a forma de avaliar as regras e duas listas com os seguintes elementos numéricos:

$$[f_{bh}, f_{b\overline{h}}, f_{\overline{bh}}, f_{\overline{b}h}, n]$$

Esse conjunto de informações é obtido através da medição de um conjunto de regras no formato padrão de regras $\mathcal{PBM}$ utilizando um conjunto de exemplos fornecido pelo usuário, e adicionadas ao final de cada uma das regras.

Como já foi descrito na Seção 2.3 na página 11, a aplicação de regras para exemplos que contenham valores não especificados é dependente do domínio do problema e do algoritmo de AM utilizado para induzir essas regras. Para que esses valores não interfiram nas medições, neste trabalho assumimos como verdadeiro qualquer teste para um atributo com valor *desconhecido*. Essa avaliação foi implementada em uma função separada, podendo ser facilmente modificada caso o pesquisador ou usuário queira implementar uma nova abordagem. O conjunto de informações padrão é medido separadamente para os exemplos que contenham valores *desconhecidos* para os atributos avaliados pela regra, e adicionados ao final de cada regra precedidos de um ponto de interrogação (?). Para atributos com valores *não-se-aplica*, assumimos que aquele atributo tem importância na avaliação da regra, e todo o teste de atributo para exemplos cujo valor é *não-se-aplica* é avaliado como falso. Nas Figura 8 e 9 são mostrados, respectivamente, o arquivo com o conjunto de regras com informações calculadas para cada uma das regras e a gramática estendida que define o formato padrão de regras.

No arquivo de saída também é incluída a expressão "*Rules Evaluated as*" seguido da informação de como o conjunto de regras foi avaliado: `ORDERED` se o conjunto de regras foi avaliado como ordenado, `UNORDERED` como não ordenado ou `INTER-CLASS ORDERED` como ordenado entre as classes e as expressões "*Names File:*", seguida do nome do arquivo contendo o arquivo de nomes, bem como a expressão "*Data File:*", seguida do nome do arquivo contendo os exemplos usados na avaliação.

Standard Rules Conversor<version> Copyright (c) Ronaldo C. Prati
Inducer: <Inducer Name> Input File: <Input File Name>
Date: <Date>

Rules Evaluated as <ORDERED|UNORDERED|INTER-CLASS ORDERED>
Names File: <Names File Name> Data File: <Data File Name>

R0001 IF <condição>
THEN CLASS = C_i $[f_{bh}, f_{b\overline{h}}, f_{\overline{bh}}, f_{\overline{b}h}, n]$?$[f^?_{bh}, f^?_{b\overline{h}}, f^?_{\overline{bh}}, f^?_{\overline{b}h}, n^?]$

R0002 IF <condição>
THEN CLASS = C_i $[f_{bh}, f_{b\overline{h}}, f_{\overline{bh}}, f_{\overline{b}h}, n]$?$[f^?_{bh}, f^?_{b\overline{h}}, f^?_{\overline{bh}}, f^?_{\overline{b}h}, n^?]$

...

Figura 8: Arquivo de Regras no Formato $\mathcal{PBM}$ com Informações Padrão

```
S ::= <rule>

<rule> ::= IF <condition> THEN <class> <extension>

<complex> ::= <fator>
        | <fator> OR <condition>

<fator> ::= <term>
        | <term> AND <fator>

<term> ::= <comparison>
        | <linear combination>

<comparison> ::= <atribute><operator><value>

<linear combination> ::= <combination><operator><value>

<combination> ::= <constant> x <numeric attribute>
        | <constant> x <numeric attribute> + <combination>

<operator> ::= <=,>=,in,<,>,!=,=

<class> ::= CLASS = <value>

<extension> ::= <know values> <unknow values>

<know values> ::= [<value>, <value>, <value>, <value>, <value>]

<unknow values> ::= ?[<value>, <value>, <value>, <value>, <value>]
```

Figura 9: Gramática do Formato Padrão de Regras Extendido

4 A Biblioteca para Cálculo das Informações

> "When you can measure what you are speaking, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science."
>
> *William Thomson(Lord Kelvin)*

A biblioteca que faz o cálculo das informações para cada regra foi implementada utilizando-se a linguagem PERL (PERL 1999). A biblioteca requer o arquivo de regras que serão avaliadas, no formato $\mathcal{PBM}$, o arquivo de dados e o arquivo de nomes, conforme descrito na Seção 2.1.2 na página 5. A saída do script, com as regras e as informações calculadas para cada regra, é armazenada em outro arquivo ou mostrada na tela. Uma visão geral da biblioteca está representada na Figura 10.

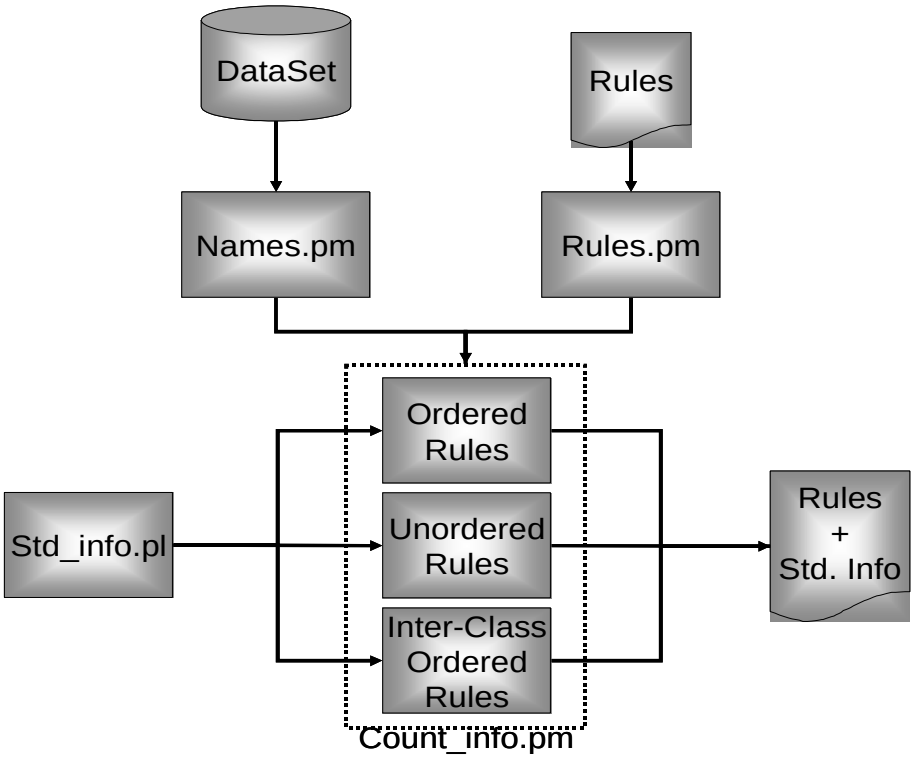

Figura 10: A Biblioteca para Cálculo das Informações Padrão

A biblioteca está dividida em quatro partes: *names.pm* para a leitura do arquivo de nomes, *rules.pm* para a leitura do arquivo de regras, *std_info.pl* que implementa uma interface com o usuário e *count_info.pm* que calcula as informações.

O módulo *names.pm* faz a leitura dos nomes dos atributos e do atributo classe no arquivo de nomes e armazena em uma estrutura de dados. O módulo *rules.pm* faz a leitura do arquivo de regras e também as armazena em uma estrutura de dados. Se existir no corpo da regra algum

atributo que não esteja no arquivo de nomes, uma mensagem de erro é apresentada e não é feita a avaliação. Nessa estrutura de dados serão armazenados os valores calculados durante a avaliação.

Para servir de interface com o usuário, foi implementado o módulo *std_info.pl*. Esse módulo permite ao usuário utilizar a biblioteca através da linha de comando, ocultando do usuário detalhes da programação PERL. Esse módulo chama a biblioteca de conversão com os parâmetros corretos, a partir dos parâmetros passados pela linha de comando. A utilização do módulo se dá pelo comando (os parâmetros entre colchetes –[ ]– são opcionais):

std_info.pl [-o|-u|-i] -d<arquivo de dados> [-n<arquivo de nomes>]
[-f<arquivo de regras>] [-s<arquivo de saída>]

onde:

-o|-u|-i indica como o conjunto de regras será avaliado, sendo -o como ordenado, -u como não ordenado e -i ordenado entre as classes. Se esse parâmeto não for especificado, a opção -u (o conjunto de regras é não ordenado) é assumida como padrão.

-d<arquivo de dados> indica o nome do arquivo contendo os exemplos que serão utilizados para avaliar o conjunto de regras.

-n<arquivo de nomes> indica o nome do arquivo contendo os nomes dos atributos do arquivo de dados. Se esse parâmetro não for especificado, é assumido como padrão o nome do arquivo de dados sem a extensão, adicionado a extensão `.names`.

-f<arquivo de regras> indica o nome do arquivo contendo o conjunto de regras no formato $\mathcal{PBM}$. Esse parâmetro pode ser substituído por um "*pipe*" de entrada, para facilitar a integração com outros sistemas.

-s<arquivo de saída> indica o nome do arquivo no qual serão gravadas as regras e as informações calculadas para cada regra. Se esse parâmetro não for especificado a saída padrão (geralmente o monitor de vídeo) é utilizada. Esse parâmetro também pode ser substituído por um "*pipe*" de saída.

Por exemplo, o comando

```
std_info.pl -dcrx.data < crx.rules_std
```

chama a biblioteca para calcular as informações para o conjunto de regras contido no arquivo `crx.rules_std` com os exemplos contidos no arquivo `crx.data`. As regras são avaliadas como não ordenadas (opção -u padrão) e o arquivo de nomes utilizado é o `crx.names` (arquivo de dados sem o sufixo `.data` com a extensão padrão `.names`). A saída contendo as regras e as informações calculadas na avaliação são mostradas na saída padrão.

O módulo *count_info.pm* (indicado na Figura 10 na página precedente pelo pontilhado) utiliza os módulos *names.pm*, *rules.pm* e os parâmetros passados pelo módulo *std_lib.pl* e faz a avaliação do conjunto de regras com os exemplos contidos no arquivo de dados. A avaliação pode ser feita como se o conjunto de regras fosse não ordenado, ordenado ou ordenado entre as classes.

Para um conjunto de regras não ordenado, deve-se aplicar cada um dos exemplos em todas as regras e contabilizar as informações. No Algoritmo 1 é mostrada a implementação do cálculo das informações para um conjunto de regras não ordenadas, onde a função *count info* calcula as informações de acordo com o resultado da aplicação do exemplo sobre a regra, como descrito na Seção 2.1.1 na página 4.

Algoritmo 1 Cálculo das Informações Padrão para Regras Não Ordenadas

Require: R: Unordered Rule Set
  T: Test Dataset
1: **for all** $t \in T$ **do**
2: **for all** $r \in R$ **do**
3: apply(r,t)
4: count info
5: **end for**
6: **end for**

Já, para um conjunto de regras ordenado, o exemplo deve ser aplicado às regras ordenadamente, a partir da primeira regra, até encontrarmos uma regra que cubra o exemplo. Esse exemplo deve ser classificado apenas por essa regra. Para as outras regras seguintes à regra coberta pelo exemplo, as informações devem ser atualizadas, incrementado-se os valores de $\overline{b}h$ ou $\overline{bh}$ conforme a classe y_i do exemplo, o que é feito pela função *update info*. No Algoritmo 2 é mostrado a implementação do cálculo das informações para um conjunto de regras ordenadas.

Algoritmo 2 Cálculo das Informações Padrão para Regras Ordenadas

Require: R: Ordered Rule Set
  T: Test Dataset
1: **for all** $t \in T$ **do**
2: **repeat**
3: $r \leftarrow$ next rule in R
4: apply(r,t)
5: count info
6: **until** r doesn't cover t
7: **for all** remaining $r \in R$ **do**
8: update info
9: **end for**
10: **end for**

A terceira forma implementada na biblioteca para se avaliar um conjunto de regras é a avaliação do conjunto de regras ordenadas entre as classes. Nessa forma, as regras estão separadas em k blocos para cada uma das k classes $C_1, C_2, \ldots, C_k$. O exemplo deve ser aplicado a todas as regras correspondentes ao primeiro bloco da classe C_1, e se e somente se nenhuma regra do bloco 1 cobrir o exemplo, devemos passar ao segundo bloco C_2, e assim por diante. Quando encontramos uma regra em um bloco i que cobre o exemplo, as regras dos outros $k - i$ blocos seguintes devem ser atualizadas, como descrito anteriormente para regras não ordenadas. No Algoritmo 3 é mostrada a implementação dessa idéia.

Na próxima seção, é mostrado um exemplo de avaliação de conjuntos de regras para cada um dos três tipos de forma de avaliação presente na biblioteca.

Algoritmo 3 Cálculo das Informações Padrão para Regras Ordenadas Entre Classes

Require: R: Inter-Class Ordered Rules
T: Test Dataset

```
for all t ∈ T do
  for all r ∈ R do
    if r.head = same class or r doesn't cover t then
      apply (r,t)
      count info
    else
      if r.head ≠ same class then
        same class ← r.head
        apply (r,t)
        count info
      else
        update info
      end if
    end if
  end for
end for
```

5 Exemplo

> The best way to learn about something is doing it.
>
> *anônimo*

Esta seção apresenta um exemplo, adaptado de Quinlan (1988) por Baranauskas & Monard (2000a), para um melhor entendimento da avaliação de conjuntos de regras ordenados, não ordenados, e ordenados entre as classes. Para induzir o conjunto de regras, foram usados os algoritmos de aprendizado $\mathcal{CN}2$, que pode induzir conjuntos de regras ordenados e não ordenados e $\mathcal{C}4.5rules$, que induz conjuntos de regras ordenados entre as classes.

O conjunto de exemplos utilizados para a indução do conjunto de regras contem medições diárias das condições meteorológicas, e cada exemplo é composto pelos seguintes atributos:

- outlook: assume os valores discretos "*sunny*", "*overcast*" ou "*rain*";
- temperature: assume um valor numérico, indicando a temperatura em graus Celsius;
- humidity: assume um valor numérico, indicando a porcentagem de humidade relativa do ar;
- windy: assume os valores discretos "*yes*" ou "*no*", indicando se há ou não vento.

Além disso, cada dia (exemplo) foi rotulado como "*go*" se as condições meteorológicas daquele dia eram favoráveis para uma viagem à fazenda ou "*dont_go*", caso contrário. Os exemplos podem ser visto na Tabela 5.

Os três conjuntos de regras induzidas serão avaliados com o conjunto de exemplos de teste mostrado na Tabela 6, onde o símbolo de interrogação (?) indica que o valor é desconhecido.

Exemplo	Outlook	Temperature	Humidity	Windy	Voyage?
E_1	sunny	25	72	yes	go
E_2	sunny	28	91	yes	dont_go
E_3	sunny	22	70	no	go
E_4	sunny	23	95	no	dont_go
E_5	sunny	30	85	no	dont_go
E_6	overcast	23	90	yes	go
E_7	overcast	29	78	no	go
E_8	overcast	19	65	yes	dont_go
E_9	overcast	26	75	no	go
E_{10}	overcast	20	87	yes	go
E_{11}	rain	22	95	no	go
E_{12}	rain	19	70	yes	dont_go
E_{13}	rain	23	80	yes	dont_go
E_{14}	rain	25	81	no	go
E_{15}	rain	21	80	no	go

Tabela 5: Conjunto de Exemplos de Treinamento Voyage

Exemplo	Outlook	Temperature	Humidity	Windy	Voyage?
T_1	sunny	25	75	yes	dont_go
T_2	sunny	28	?	yes	dont_go
T_3	sunny	22	70	no	go
T_4	sunny	23	95	no	dont_go
T_5	sunny	?	85	no	dont_go
T_6	overcast	23	90	yes	go
T_7	overcast	29	82	no	dont_go
T_8	overcast	20	65	yes	dont_go
T_9	overcast	26	?	no	go
T_{10}	overcast	20	87	yes	go
T_{11}	rain	22	95	no	go
T_{12}	rain	19	70	?	dont_go
T_{13}	rain	23	80	yes	dont_go
T_{14}	rain	25	81	no	go
T_{15}	rain	21	80	no	go

Tabela 6: Conjunto de Exemplos de Teste Voyage

Nas Figuras 11 na próxima página e 12 na página 24 são mostrados, respectivamente, o conjunto de regras não ordenadas induzidas pelo algortitmo de aprendizado $\mathcal{CN}2$ e o mesmo conjunto de regras, avaliadas como não ordenadas, utilizando o conjunto de exemplos de teste voyage(Tabela 6).

Para um melhor entendimento do processo de avaliação de regras não ordenadas, tomemos o exemplo de teste T_3 e as regras $R0001$ e $R0003$ do conjunto de regras. Como essas duas regras cobrem corretamente o referido exemplo, e as regras são avaliadas independentemente, o exemplo é contabilizado como *bh* em ambas as regras.

Tomemos agora os exemplos de teste T_2, T_9 e T_{12}, e a regra $R0001$. Cada um desses exemplos têm um valor *desconhecido* em um dos atributos que aparecem na regra (Humidity e Windy). Esses três exemplos serão contabilizados na segunda lista de valores — aquela prescedida pelo simbolo de interrogação (?) —, que trata exclusivamente de exemplos que contenham valores *desconhecidos*. Por outro lado o exemplo T_5, que contém um valor *desconhecido* no atribu-

```
Standard Rules Conversor v1.1.4 Copyright (c)Ronaldo C. Prati
Inducer: cn2            Input File: voyage.cn2.unordered_output
Date: Thu Jun 14 21:15:06 2001

R0001   IF humidity < 83.00
            AND windy = no
        THEN CLASS = go

R0002   IF outlook = overcast
            AND temperature > 19.50
        THEN CLASS = go

R0003   IF outlook = sunny
            AND humidity < 76.00
        THEN CLASS = go

R0004   IF outlook = rain
            AND humidity > 87.50
        THEN CLASS = go

R0005   IF outlook = sunny
            AND humidity > 83.00
        THEN CLASS = dont_go

R0006   IF temperature < 24.00
            AND humidity < 80.50
            AND windy = yes
        THEN CLASS = dont_go

R0007   DEFAULT CLASS = go
```

Figura 11: Regras não Ordenadas Geradas pelo Indutor $\mathcal{CN}2$

to Temperature é tratado normalmente na primeira lista de valores, pois o atributo não está presente na regra.

Na Tabela 7 na próxima página são mostradas algumas das medidas apresentadas na Seção 2.4, para esse conjunto de regras não ordenadas (Figura 12).

Semelhantemente, nas Figuras 13 e 14 são apresentadas os conjuntos de regras ordenadas induzidas pelo $\mathcal{CN}2$ e o mesmo conjunto de regras avaliadas como não ordenadas, com o conjunto de exemplos de teste voyage (Tabela 6).

Tomemos os exemplos de teste T_3 e T_{15} e as regras $R0001$ e $R0004$ desse conjunto de regras. Esses exemplos são cobertos por ambas as regras, se estas forem vistas independentemente. Entretanto, como existe uma ordem na avaliação das regras (o exemplo é coberto apenas pela primeira regra disparada), somente na regra $R0001$ são contabilizados os valores para bh.

Analizemos agora o exemplo de teste T_9 e as regras $R0001$ e $R0003$. O exemplo tem um valor *desconhecido* no atributo Humidity. Esse exemplo é tratado em ambas as regras, pois o valor *desconhecido* não pode determinar precisamente qual das regras deve realmente cobrir o exemplo.

```
Standard Rules Conversor v1.1.4 Copyright (c)Ronaldo C. Prati
Inducer: cn2            Input File: voyage.cn2.unordered_output
Date: Thu Jun 14 21:15:06 2001

Rules Evaluated as UNORDERED
Names File: voyage.names   Data File: voyage.test

R0001   IF humidity < 83.00
            AND windy = no
        THEN CLASS = go  [0.250,0.083,0.417,0.250,12] ?[0.333,0.333,0.333,0.000,3]

R0002   IF outlook = overcast
            AND temperature > 19.50
        THEN CLASS = go  [0.214,0.143,0.286,0.357,14] ?[0.000,0.000,1.000,0.000,1]

R0003   IF outlook = sunny
            AND humidity < 76.00
        THEN CLASS = go  [0.077,0.077,0.462,0.385,13] ?[0.000,0.500,0.000,0.500,2]

R0004   IF outlook = rain
            AND humidity > 87.50
        THEN CLASS = go  [0.077,0.000,0.538,0.385,13] ?[0.000,0.000,0.500,0.500,2]

R0005   IF outlook = sunny
            AND humidity > 83.00
        THEN CLASS = dont_go  [0.154,0.000,0.462,0.385,13] ?[0.500,0.000,0.500,0.000,2]

R0006   IF temperature < 24.00
            AND humidity < 80.50
            AND windy = yes
        THEN CLASS = dont_go  [0.167,0.000,0.500,0.333,12] ?[0.333,0.000,0.333,0.333,3]

R0007   DEFAULT CLASS = go
```

Figura 12: Regras Avaliadas como Não Ordenadas

Regra	*Acc*	*Err*	*NegRel*	*Sens*	*Spec*	*Cov*	*Sup*	*Nov*	*Sat*
R0001	0.751	0.249	0.625	0.500	0.834	0.333	0.250	0.084	0.502
R0002	0.599	0.401	0.445	0.375	0.667	0.357	0.214	0.010	0.066
R0003	0.500	0.500	0.545	0.167	0.857	0.154	0.077	0.006	0.072
R0004	1.000	0.000	0.583	0.167	1.000	0.077	0.077	0.041	1.000
R0005	1.000	0.000	0.545	0.286	1.000	0.154	0.154	0.071	1.000
R0006	1.000	0.000	0.600	0.334	1.000	0.167	0.167	0.084	1.000

Tabela 7: Medidas Obtidas a Partir da Avalição das Regras Não Ordenadas

```
Standard Rules Conversor v1.1.4 Copyright (c)Ronaldo C. Prati
Inducer: cn2-ordered     Input File: voyage.cn2.ordered_output
Date: Thu Jun 14 21:15:06 2001

R0001   IF humidity < 83.00
            AND windy = no
        THEN CLASS = go

R0002   IF outlook = sunny
            AND humidity > 78.50
        THEN CLASS = dont_go

R0003   IF humidity > 83.50
        THEN CLASS = go

R0004   IF temperature < 24.00
        THEN CLASS = dont_go

R0005   DEFAULT CLASS = go
```

Figura 13: Regras Ordenadas Geradas pelo Indutor $\mathcal{CN}2$

Regra	*Acc*	*Err*	*NegRel*	*Sens*	*Spec*	*Cov*	*Sup*	*Nov*	*Sat*
R0001	0.751	0.249	0.625	0.500	0.834	0.333	0.250	0.084	0.502
R0002	1.000	0.000	0.545	0.286	1.000	0.154	0.154	0.071	1.000
R0003	1.000	0.000	0.364	0.300	1.000	0.214	0.214	0.061	1.000
R0004	1.000	0.000	0.584	0.375	1.000	0.200	0.200	0.093	1.000

Tabela 8: Medidas Obtidas a Partir da Avalição das Regras Ordenadas

Na Tabela 8 são mostradas algumas das medidas apresentadas na Seção 2.4, para esse conjunto de regras ordenadas(Figura 14).

Finalmente, nas Figuras 15 e 16 são mostradas, respectivamente, os conjuntos de regras ordenadas entre as classes e induzidas pelo algoritmo de aprendizado $\mathcal{C}4.5rules$ e o mesmo conjunto de regras avaliado como regras ordenadas entre as classes, para o conjunto de exemplos de teste voyage.

Seja R_{go} o conjunto de regras correspondente à classe "*go*" e R_{dont_go} o conjunto de regras correspondente à classe "*dont_go*". Então, $R_{go} = \{R0001\}$ e $R_{dont_go} = R0002, R0003$. A avaliação se dá de uma forma ordenada entre esses dois conjuntos (R_1 e R_2). A avaliação das regras dentro do conjunto R_2 se dá de uma forma ordenada.

Na Tabela 9 são mostradas algumas das medidas apresentadas na Seção 2.4, para esse conjunto de regras ordenadas entre as classes(Figura 16).

```
Standard Rules Conversor v1.1.4 Copyright (c)Ronaldo C. Prati
Inducer: cn2-ordered     Input File: voyage.cn2.ordered_output
Date: Thu Jun 14 21:15:06 2001

Rules Evaluated as ORDERED
Names File: voyage.names   Data File: voyage.test

R0001   IF humidity < 83.00
            AND windy = no
        THEN CLASS = go  [0.250,0.083,0.417,0.250,12] ?[0.333,0.333,0.333,0.000,3]

R0002   IF outlook = sunny
            AND humidity > 78.50
        THEN CLASS = dont_go  [0.154,0.000,0.462,0.385,13] ?[0.500,0.000,0.500,0.000,2]

R0003   IF humidity > 83.50
        THEN CLASS = go  [0.231,0.000,0.538,0.231,13] ?[0.500,0.500,0.000,0.000,2]

R0004   IF temperature < 24.00
        THEN CLASS = dont_go  [0.200,0.000,0.467,0.333,15] ?[0.000,0.000,0.000,0.000,0]

R0005   DEFAULT CLASS = go
```

Figura 14: Regras Avaliadas como Ordenadas

```
Standard Rules Conversor v1.1.4 Copyright (c)Ronaldo C. Prati
Inducer: c4.5rules          Input File: voyage.c45rules.output
Date: Thu Jun 14 21:15:06 2001

R0001   IF outlook = rain
            AND windy = no
        THEN CLASS =  go

R0002   IF outlook = sunny
            AND humidity > 78
        THEN CLASS =  dont_go

R0003   IF outlook = rain
            AND windy = yes
        THEN CLASS =  dont_go

R0004   DEFAULT CLASS =  go
```

Figura 15: Regras Ordenadas Entre as Classes Geradas pelo Indutor $\mathcal{C}4.5rules$

```
Standard Rules Conversor v1.1.4 Copyright (c)Ronaldo C. Prati
Inducer: c4.5rules          Input File: voyage.c45rules.output
Date: Thu Jun 14 21:15:06 2001

Rules Evaluated as INTER-CLASS ORDERED
Names File: voyage.names   Data File: voyage.test

R0001   IF outlook = rain
            AND windy = no
        THEN CLASS =  go  [0.214,0.000,0.500,0.286,14] ?[0.000,1.000,0.000,0.000,1]

R0002   IF outlook = sunny
            AND humidity > 78
        THEN CLASS =  dont_go  [0.154,0.000,0.462,0.385,13] ?[0.500,0.000,0.500,0.000,2]

R0003   IF outlook = rain
            AND windy = yes
        THEN CLASS =  dont_go  [0.071,0.000,0.500,0.429,14] ?[1.000,0.000,0.000,0.000,1]

R0004   DEFAULT CLASS =  go
```

Figura 16: Regras Avaliadas como Ordenadas Entre as Classes

Regra	*Acc*	*Err*	*NegRel*	*Sens*	*Spec*	*Cov*	*Sup*	*Nov*	*Sat*
R0001	1.000	0.000	0.636	0.428	1.000	0.214	0.214	0.107	1.000
R0002	1.000	0.000	0.545	0.286	1.000	0.154	0.154	0.071	1.000
R0003	1.000	0.000	0.538	0.142	1.000	0.071	0.071	0.036	1.000

Tabela 9: Medidas Obtidas a Partir da Avalição das Regras Ordenadas Entre as Classes

6 Considerações Finais

> "Researches are adept at saying what they are doing, much less so at saying what they are learning."
>
> *David Etherington*

A análise de um conjunto de regras induzidas por um algoritmo de Aprendizado de Máquina simbólico deve levar em consideração vários aspectos que não somente a precisão do classificador como uma "caixa preta". Por representarem o conhecimento de uma forma explícita e compreensível por serem humanos (regras ou árvores de decisão transformadas em regras, no contexto desse trabalho), diversas questões quanto a qualidade, interessabilidade, novidade, entre outros, que cada uma essas regras possam apresentar podem ser levantadas.

Algumas dessa questões podem ser analisadas considerando medidas objetivas, calculadas a partir de um conjunto de exemplos. Esses exemplos devem ser, preferencialmente, desconhecidos do indutor (não utilizadas como exemplos para a indução do classificador), para uma análise mais realística e menos tendenciosa.

O levantamento dessas medidas pode ser realizado através de um conjunto de informações sobre cada regra. No entanto, essa não é uma tarefa fácil quanto estamos comparando con-

junto de regras provenientes de diversos indutores, uma vez que, geralmente, esses indutores apresentam essas informações, quando presentes, de uma maneira não uniforme.

Este trabalho é uma continuação de Prati, Baranauskas & Monard (2001), no qual os classificadores induzidos por diversos algoritmos de aprendizado simbólicos são transformados em um formato padrão de regras, mas no qual não foi levado em conta a avaliação dessas regras. Neste trabalho, além de estender o formato padrão de regras apresentado em Prati, Baranauskas & Monard (2001), considerando agora as três possíveis formas de avaliar regras induzidas por algoritmos de AM, também implementamos a avaliação da matriz de contingência de cada uma dessas regras segundo esses três critérios, utilizando um conjunto de dados fornecido pelo usuário.

Na realidade, avaliamos duas matrizes de contingência: uma para exemplos que não contêm valores *desconhecidos* e outra para exemplos que contêm valores *desconhecidos*. Deve ser observado que as informações adicionais fornecidas por essas duas matrizes de contingência permitem avaliar mais cuidadosamente as regras induzidas por algoritmos de AM mais sofisticados.

Assim, com base neste novo formato padrão de regras, este trabalho têm como principal objetivo ajudar o pesquisador ou usuário na análise e compreensão das regras induzidas pelos indutores simbólicos mais usados na área de AM, fornecendo, de uma maneira padronizada, além do formato padrão de regras, um conjunto de informações pelas quais é possível derivar facilmente medidas de qualidade de regras, também de uma maneira padronizada.

Referências

An, A. & Cercone, N. (1999). An empirical study on rule quality measures. *Lecture Notes in Artificial Inteligence* (1711), 482–491.

Baranauskas, J. A. & Batista, G. E. A. P. A. (2000). O projeto DISCOVER: Idéias iniciais (comunicação pessoal).

Baranauskas, J. A. & Monard, M. C. (2000b). Reviewing some machine learning concepts and methods. Technical Report 102, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_102.ps.zip`.

Baranauskas, J. A. & Monard, M. C. (2000a). An unified overview of six supervisioned symbolic machine learning inducers. Technical Report 103, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_103.ps.zip`.

Batista, G. E. A. P. A. (2001). Sintaxe padrão do arquivo de exemplos do projeto DISCOVER. `http://www.icmc.sc.usp.br/~gbatista/SintaxePadraoFinal.htm`.

Bayardo, R. J. & Agrawal, R. (1999). Mining the most interesting rules. In *Fifth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pp. 145–154.

Clark, P. & Boswell, R. (1989). The $\mathcal{CN}2$ induction algorothm. *Machine Learning 3*(4), 261–283.

Clark, P. & Boswell, R. (1991). Rule induction with $\mathcal{CN}2$: Some recent improvements. In *Procedings of the Fifth European Conference*, Springer Verlag, pp. 151–163. Y. Kodratoff. `http://www.cs.utexas.edu/users/pclark/papers/newcn.ps`.

Clark, P. & Niblett, T. (1987). Induction in noise domains. In I. Bratko & N. Lavrač (Eds.), *Proceedings of the Second European Working Session on Learning*, Wilmslow, UK, pp. 11–30. Sigma.

Cohen, W. W. (1995). Fast effective rule induction. In *Machine Learning: Procedings of the Twelfth International Conference*, San Francisco, CA, pp. 115–123. Morgan Kaufmam.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R. (1996). *Advances in Knowledge Discovery and Data Mining*. AAAI Press.

Félix, L. C. & et. all (1998). $\mathcal{MLC}$++ biblioteca de aprendizado de máquima em *C*++. Technical Report 72, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_72.ps.zip`.

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

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, pp. 1–9.

Horst, P. S. (1999). Avaliação do conhecimento adquirido por algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC-USP.

Kohavi, R., Sommerfield, D. & Dougherty, J. (1997). Data mining using $\mathcal{MLC}$++: A machine learning library in *C*++. *International Journal on Artificial Intellegence Tools*.

Lavrač, N., Flach, P. & Zupan, B. (1999). Rule evaluatin measures: A unifying view. *Lecture Notes in Artificial Inteligence* (1634), 174–185.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Murthy, S. K., Kasif, S. & Salzberg, S. L. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research 2*(1), 1–32. `http://www.cs.jhu.edu/~salzberg/jair94.ps`.

PERL (1999). *Programming in PERL*. Morgan Kaufmann Publishers, Inc.

Prati, R. C., Baranauskas, J. A. & Monard, M. C. (1999). BibView: Um sistema para auxiliar a manutenção de registros para o BibTeX. Technical Report 95, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_95.ps.zip`.

Prati, R. C., Baranauskas, J. A. & Monard, M. C. (2001). Uma proposta de unificação da linguagem de representação de conceitos de algoritmos de aprendizado de máquina simbólicos. Technical Report 137, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip`.

Quinlan, J. R. (1986). *Induction of Decision Trees*, Volume 1, pp. 81–106. Shavlik and Dietterich.

Quinlan, J. R. (1988). *C4.5 Programs for Machine Learning*. CA: Morgan Kaufmann.

Quinlan, J. R. (1990). Unknow attribute values in induction. In *Proceedings of the Sixth International Workshop on Machine Learning*, pp. 164–168. Morgan Kaufmann Publisher.

Weiss, S. M. & Indurkhya, N. (1998). *Predictive Data Mining: A Practical Guide*. San Francisco, CA: Morgan Kaufmann.