Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Projeto e Implementação do Módulo de Combinação e Explicação de Classificadores Simbólicos do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Flávia Cristina Bernardini
Maria Carolina Monard

Nº 159

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos

Abril/2002

Projeto e Implementação do Módulo de Combinação e Explicação de Classificadores Simbólicos do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$*

Flávia Cristina Bernardini

Maria Carolina Monard

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: {fbernard, mcmonard}@icmc.sc.usp.br

Resumo

Encontra-se em desenvolvimento no LABIC — Laboratório de Inteligência Computacional — um projeto de grande porte denominado DISCOVER, que tem por objetivo fornecer um ambiente integrado para apoiar as etapas do processo de descoberta de conhecimento. No intuito de testar algumas idéias que poderão futuramente ser incorporadas no ambiente DISCOVER, foi proposto um sistema computacional protótipo, implementado na linguagem de programação lógica Prolog, denominado $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. O $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ tem como objetivo (a) avaliar regras de conhecimento induzidas por algoritmos de Aprendizado de Máquina simbólico e (b) avaliar diversas formas de combinar classificadores simbólicos bem como explicar a classificação de novos exemplos realizada por um *ensemble* de classificares simbólicos. O objetivo (a) é implementado pelo Módulo de Análise de Regras e o objetivo (b) pelo Módulo de Combinação e Explicação. Esses módulos constituem os módulos principais do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Este trabalho tem por objetivo apresentar as funcionalidades implementadas no Módulo de Combinação e Explicação. Também é apresentado um exemplo de aplicação do Módulo de Combinação e Explicação utilizando um conjunto de exemplos artificial.

Palavras-Chave: Aprendizado de Máquina, Combinação de Classificadores, Explicação de *Ensembles*, Prolog.

Abril 2002

*Trabalho realizado com auxílio da FAPESP

Sumário

1 Introdução **1**

2 Aprendizado de Máquina Supervisionado **2**

2.1 Classificadores Simbólicos 3

2.2 Avaliação de Regras 3

2.3 *Ensemble* de Classificadores 4

3 Arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ **4**

4 Metodologia para Documentação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ **6**

4.1 Componentes do Diagrama de Fluxo de Execução de Procedimentos 6

4.1.1 Procedimentos 6

4.1.2 Condições 8

4.1.3 Fluxo de Execução 11

4.2 Descrição do Fluxo de Execução do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ 12

5 Módulo de Combinação e Explicação **13**

5.1 Classificação de Exemplos 14

5.2 Determinação da Matriz de Confusão 16

5.3 Explicação de *Ensembles* 17

5.4 Combinação de Classificadores Simbólicos 18

5.4.1 Algoritmo $\mathcal{RCO}$ 19

5.4.2 Algoritmo k-Fold Cross Validation 21

6 Um Exemplo de Aplicação do MCE **23**

6.1 Descrição do Experimento 1 28

6.2 Descrição do Experimento 2 30

6.3 Descrição do Experimento 3 34

6.4 Descrição dos Experimentos 4, 5 e 6 36

6.5 Análise dos Resultados 41

7 Conclusão **42**

A Classificadores Utilizados pelo $\mathcal{RCO}$ **44**

A.1 Classificadores Induzidos pelo $\mathcal{CN}2$ 44

A.1.1 $\mathbf{h}_1$ 44

A.1.2 $\mathbf{h}_2$ 45

A.1.3 $\mathbf{h}_3$ 46

A.1.4 $\mathbf{h}_4$ 48

A.1.5 $\mathbf{h}_5$ 49

A.2 Classificadores Induzidos pelo $\mathcal{S}$ee5 50

A.2.1 $\mathbf{h}_6$ 50

A.2.2 $\mathbf{h}_7$ 51

B Classificadores Construídos pelo $\mathcal{RCO}$ nos Experimentos Realizados **51**

B.1 Experimento 1 52

B.1.1 $\mathbf{h}_{1_\mathtt{accR}}$ 52

B.1.2 $\mathbf{h}_{1_\mathtt{covR}}$ 53

B.1.3 $\mathbf{h}_{1_\mathtt{satR}}$ 54

B.1.4 $\mathbf{h}_{1_\mathtt{sensR}}$ 55

B.1.5 $\mathbf{h}_{1_\mathtt{specR}}$ 56

B.1.6 $\mathbf{h}_{1_\mathtt{supR}}$ 57

B.2 Experimento 2 58

B.2.1 $\mathbf{h}_{2_\mathtt{accR}}$ 58

B.2.2 $\mathbf{h}_{2_\mathtt{covR}}$ 58

B.2.3 $\mathbf{h}_{2_\mathtt{satR}}$ 58

B.2.4 $\mathbf{h}_{2_\mathtt{sensR}}$ 58

B.2.5 $\mathbf{h}_{2_\mathtt{specR}}$ 58

B.2.6 $\mathbf{h}_{2_\mathtt{supR}}$ 58

B.3 Experimento 3 58

B.3.1 $\mathbf{h}_{3_\mathtt{accR}}$ 58

B.3.2 $\mathbf{h}_{3_\mathtt{covR}}$ 58

B.3.3 $\mathbf{h}_{3_\mathtt{satR}}$ 59

B.3.4 $\mathbf{h}_{3_\mathtt{sensR}}$ 59

B.3.5 $\mathbf{h}_{3_\mathtt{specR}}$ 59

B.3.6 $\mathbf{h}_{3_\mathtt{supR}}$ 59

B.4 Experimento 4 60

B.4.1 $\mathbf{h}_{4_\mathtt{accR}}$ 60

B.4.2 $\mathbf{h}_{4_\mathtt{covR}}$ 60

B.4.3 $\mathbf{h}_{4_satR}$ 61

B.4.4 $\mathbf{h}_{4_sensR}$ 62

B.4.5 $\mathbf{h}_{4_specR}$ 62

B.4.6 $\mathbf{h}_{4_supR}$ 63

B.5 Experimento 5 64

B.5.1 $\mathbf{h}_{5_accR}$ 64

B.5.2 $\mathbf{h}_{5_covR}$ 65

B.5.3 $\mathbf{h}_{5_satR}$ 65

B.5.4 $\mathbf{h}_{5_sensR}$ 66

B.5.5 $\mathbf{h}_{5_specR}$ 66

B.5.6 $\mathbf{h}_{5_supR}$ 68

B.6 Experimento 6 68

B.6.1 $\mathbf{h}_{6_accR}$ 68

B.6.2 $\mathbf{h}_{6_covR}$ 69

B.6.3 $\mathbf{h}_{6_satR}$ 69

B.6.4 $\mathbf{h}_{6_sensR}$ 70

B.6.5 $\mathbf{h}_{6_specR}$ 71

B.6.6 $\mathbf{h}_{6_supR}$ 72

Lista de Figuras

1 Arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ 5

2 Diagrama de Fluxo de Execução de Procedimentos do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ 7

3 Exemplo de Representação de um Procedimento 8

4 Exemplo de Representação de um Conjunto de Condições 8

5 Exemplo de Representação de um Subconjunto de Condições 10

6 Tipos de Barramento 11

7 Subconjuntos de Regras — Forma de Avaliação *Interclass* 17

8 Ilustração do Algoritmo $\mathcal{RCO}$ 22

9 Função Verdadeira f 25

10 Hipóteses Induzidas por $\mathcal{CN}2$ e $\mathcal{S}$ee5 Utilizadas nos Experimentos 25

11 Representação Gráfica das Hipóteses Induzidas por $\mathcal{CN}2$. 27

12 Representação Gráfica das Hipóteses Induzidas por $\mathcal{S}$ee5. 28

13 Ilustração do Experimento 1. 29

14 Representação Gráfica das Hipóteses Induzidas por $\mathcal{RCO}$ na Fase 2 — Experimento 1 31

15 Ilustração do Experimento 2. 32

16 Ilustração do Experimento 3. 34

17 Representação Gráfica das Hipóteses Induzidas por $\mathcal{RCO}$ — Experimento 4 38

18 Representação Gráfica das Hipóteses Induzidas por $\mathcal{RCO}$ — Experimento 5 39

19 Representação Gráfica das Hipóteses Induzidas por $\mathcal{RCO}$ — Experimento 6 40

Lista de Tabelas

1 Conjunto de Exemplos no Formato Atributo-Valor 2

2 Medidas Genéricas de Avaliação de Regras 4

3 Procedimentos Principais do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ 5

4 Condições Pertencentes ao Conjunto de Condições `E` 9

5 Condições Pertencentes ao Conjunto de Condições `R` 9

6 Condições `1`, `2`, `3`, `4` e `5` 10

7 Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{S}$`ee5` (Fase 1 — Experimento 1) . . 28

8 Resultados Obtidos — Hipóteses de Entrada do $\mathcal{RCO}$ (Fase 2 — Experimento 1) 30

9 Resultados Obtidos Utilizando $\mathcal{RCO}$ (Fase 2 — Experimento 1) 30

10 Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{S}$`ee5` (Fase 1 — Experimento 2) . . 31

11 Resultados Obtidos — Hipóteses de Entrada do $\mathcal{RCO}$ (Fase 2 — Experimento 2) 33

12 Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{S}$`ee5` (Fase 1 — Experimento 3) . . 35

13 Resultados Obtidos — Hipóteses de Entrada do $\mathcal{RCO}$ (Fase 2 — Experimento 3) 35

14 Resultados Obtidos Utilizando $\mathcal{RCO}$ (Fase 2 — Experimento 3) 36

15 Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 4 37

16 Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 5 37

17 Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 6 37

18 Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{S}$`ee5` na Fase 1 dos Experimentos 1, 2 e 3 41

19 Número de Regras Selecionadas (com a Regra *Default*) nos Experimentos 4, 5 e 6 42

Lista de Algoritmos

1 Avaliação *ordered* 15

2 Avaliação *unordered* 16

3 Matriz de Confusão 18

4 Explicação de *Ensembles* 19

5 $\mathcal{RCO}$ 20

6 k-Fold Cross Validation 23

7 Construção de Partições 24

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

1 Introdução

Encontra-se em desenvolvimento no LABIC[1] — Laboratório de Inteligência Computacional — um projeto de grande porte denominado DISCOVER, inicialmente proposto por Baranauskas e Batista (Baranauskas and Batista, 2000). O projeto DISCOVER tem como objetivo fornecer um ambiente integrado para apoiar as etapas do processo de descoberta de conhecimento, oferecendo funcionalidades voltadas para Aprendizado de Máquina (AM) (Batista, 1997; Caulkins, 2000; Milaré, 2000; Martins, 2001; Pila, 2000), Mineração de Dados[2](MD) (Batista, 2000; Félix, 1998; Horst, 1999; Lee, 2000; Nagai, 2000; Pugliesi, 2001; Baranauskas, 2001) e Mineração de Textos[3](MT) (Imamura, 2001).

As funcionalidades voltadas para AM consideram, entre outros, um formato padrão para as regras induzidas por algoritmos de AM simbólico, denominado $\mathcal{PBM}$ (Prati et al., 2001b; Prati et al., 2001a) bem como um formato padrão para os exemplos utilizados (Batista, 2001).

No intuito de testar algumas idéias que poderão futuramente ser implementadas no ambiente DISCOVER, foi proposto em (Gomes, 2001) e (Bernardini, 2001) um sistema computacional, denominado $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, composto de dois módulos principais, os quais são:

1. Módulo de Análise de Regras (MAR), com o objetivo de analisar diversas medidas de avaliação de regras propostas na literatura para avaliar regras induzidas por algoritmos de AM simbólico; e

2. Módulo de Combinação e Explicação (MCE), com o objetivo de construir, dado um conjunto de classificadores simbólicos, um outro classificador simbólico que contenha regras desse conjunto de classificadores que sejam as "melhores" segundo as diversas medidas de qualidade de regras implementadas no MAR. Outro objetivo deste módulo é fornecer ao usuário, dado um conjunto de classificadores simbólicos e um exemplo do domínio dos classificadores, quais as regras que foram disparadas nesses classificadores e qual a "melhor" regra, segundo, também, alguma das diversas medidas de avaliação de regras implementadas no MAR. Em outras palavras, fornecer ao usuário uma explicação da classificação.

Foi decidido implementar o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ na linguagem de programação lógica Prolog (Bratko, 1990), mais especificamente em LPA-PROLOG (Westwood, 2000b; Westwood, 2000a). A escolha da linguagem Prolog deve-se ao fato de ser uma linguagem apropriada para o desenvolvimento de protótipos nos quais novas idéias possam ser facilmente testadas antes de serem implementadas em um sistema de grande porte. Uma outra característica importante de Prolog é a possibilidade de realizar consultas para obter informações que não ficam restritas àquelas pré-definidas no sistema.

[1] http://labic.icmc.sc.usp.br
[2] Data Mining
[3] Text Mining

Os dados de entrada para o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ consistem de:

1. um ou vários conjuntos de regras induzidas por algoritmos de AM simbólico; e
2. um conjunto de exemplos.

Ambos os dados de entrada estão na Sintaxe Padrão Prolog, que é equivalente à sintaxe padrão $\mathcal{PBM}$ de regras e exemplos adotada no projeto DISCOVER. A conversão do formato padrão $\mathcal{PBM}$ para a Sintaxe Padrão Prolog, descrita em detalhes em (Gomes et al., 2002), foi implementada através de scripts Perl.

O objetivo deste trabalho é descrever em detalhes o projeto e a implementação do Módulo de Combinação e Explicação (MCE) do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Também, para ilustrar melhor os procedimentos do MCE, são descritos diversos experimentos realizados utilizando um conjunto de dados artificiais

O trabalho está organizado da seguinte forma: na Seção 2 é feita uma breve descrição sobre alguns conceitos de aprendizado de máquina, necessários para um melhor entendimento do trabalho realizado. Na Seção 3, é apresentada a arquitetura do sistema $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Na Seção 4, é apresentada uma metodologia para a documentação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Na Seção 5, é apresentada a descrição dos procedimentos que constituem o Módulo de Combinação e Explicação. Na Seção 6, é mostrado um exemplo de utilização desse módulo e, finalmente, na Seção 7, são apresentadas as conclusões do trabalho realizado.

2 Aprendizado de Máquina Supervisionado

No problema padrão de AM supervisionado, ao algoritmo de aprendizado de máquina é dado um conjunto de exemplos S de treinamento, com N exemplos $T_i, i = 1, ..., N$, escolhidos de um domínio X com uma distribuição D fixa, desconhecida e arbitrária, da forma $\{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$ para alguma função desconhecida $y = f(\mathbf{x})$, como mostra a Tabela 1. Os $\mathbf{x}_i$ são tipicamente vetores da forma $<x_{i1},x_{i2},...,x_{iM}>$, com valores discretos ou numéricos, como peso, altura, cor, idade, e assim por diante. Assim, x_{ij} refere-se ao valor do atributo (ou *feature*) j, denominado $\mathbf{X}_j$ do exemplo T_i, como mostra a Tabela 1. Os valores y_i referem-se ao valor do atributo Y, freqüentemente denominado classe.

	X_1	X_2	$\dots$	X_M	Y
T_1	x_{11}	x_{12}	$\dots$	x_{1M}	y_1
T_2	x_{21}	x_{22}	$\dots$	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_N	x_{N1}	x_{N2}	$\dots$	x_{NM}	y_N

Tabela 1: Conjunto de Exemplos no Formato Atributo-Valor

Os valores de y são tipicamente pertencentes a um conjunto discreto de classes $C_v, v = 1, ..., NCl$, da forma $\{C_1, ..., C_{NCl}\}$, quando se trata de *classificação* ou ao conjunto de números reais em caso de *regressão*.

Dado um conjunto S de exemplos de treinamento a um algoritmo de AM, um *classificador* **h** será induzido. O classificador consiste da *hipótese* feita sobre a verdadeira (mas desconhecida) função f. Dados novos exemplos **x**, o classificador, ou hipótese, **h** prediz o valor correspondente y.

2.1 Classificadores Simbólicos

Algoritmos de AM simbólicos induzem o classificador **h** de tal forma que o conceito descrito por **h** é facilmente interpretável por seres humanos e, geralmente, pode ser transformado em conjuntos de regras *if-then*, ou seja, regras do tipo *Corpo* $\rightarrow$ *Cabeça* ou $Body \rightarrow Head$. Uma regra R_u pode então ser resumidamente denotada como $B \rightarrow H$.

Pode-se medir um conjunto de regras **h** dado um conjunto de exemplos S. Neste caso, **h** consiste de um conjunto de regras com NR regras $R_u, u = 1, ..., NR$, ou seja, **h** $=\{R_1, ..., R_{NR}\}$, o qual denominamos classificador simbólico. Neste caso, além de medir a precisão de **h** como um todo, isto é, como um classificador tipo caixa preta, é possível avaliar separadamente cada uma das regras que constituem **h**.

2.2 Avaliação de Regras

Várias medidas de avaliação de regras têm sido propostas na literatura (Lavrac et al., 1999; Freitas, 1998b; Freitas, 1998a; Hilderman and Hamilton, 1999). Neste trabalho, serão utilizadas medidas de avaliação de regras propostas por Lavrac, em (Lavrac et al., 1999), as quais podem ser divididas em três categorias distintas:

1. Medidas genéricas de avaliação de regras;
2. Medidas relativas de avaliação de regras; e
3. Medidas relativas de avaliação de regras com peso.

O Módulo de Combinação e Explicação descrito neste trabalho usa somente as medidas genéricas de avaliação de regras, as quais estão descritas sucintamente na Tabela 2, bem como o nome pelo qual cada uma dessas medidas é referenciada nos procedimentos do MCE.

Em (Gomes, 2001) há uma descrição completa das medidas pertencentes a cada uma dessas categorias.

`accR`:	Precisão da regra
`errR`:	Erro da regra
`negrelR`:	Confiança negativa da regra
`sensR`:	Sensitividade da regra
`specR`:	Especificidade da regra
`covR`:	Cobertura da regra
`supR`:	Suporte da regra
`novR`:	Medida que mostra o quanto uma regra é nova, interessante ou fora do comum
`satR`:	Satisfação da regra

Tabela 2: Medidas Genéricas de Avaliação de Regras

2.3 *Ensemble* de Classificadores

Dado um conjunto $\mathcal{H}$ de classificadores, denotados por $\mathbf{h}_1$,...,$\mathbf{h}_L$, pode-se combinar suas decisões individuais — tipicamente através da votação com ou sem peso — para classificar novos exemplos, formando, assim, um *ensemble* de classificadores. Entretanto, um *ensemble* de classificadores é um classificador caixa preta, isto é, sem condições de fornecer ao usuário uma explicação das suas decisões, independentemente de cada classificador $\mathbf{h}_1$,...,$\mathbf{h}_L$ que constitui o *ensemble* ser ou não ser simbólico.

O Módulo de Combinação e Explicação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ tem por objetivo testar a possibilidade de, dado um conjunto $\mathcal{H} = \{\mathbf{h}_1, ..., \mathbf{h}_L\}$ de classificadores simbólicos, construir outro classificador simbólico $\mathbf{h}^*$ que consiste das "melhores" regras do conjunto $\mathcal{H}$ de classificadores simbólicos. Para determinar essas melhores regras, são utilizadas as medidas genéricas de avaliação de regras, mostradas na Tabela 2, implementadas no MAR.

A arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ é descrita a seguir[4].

3 Arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

A Figura 1 mostra a arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Os dois módulos principais (MAR e MCE) consistem de um conjunto de procedimentos Prolog específicos de cada módulo. O Módulo Auxiliar (MA) contém procedimentos para ler arquivos especificados pelo usuário — arquivos que contém conjuntos de regras e o conjunto de exemplos na Sintaxe Padrão Prolog —, bem como uma biblioteca de procedimentos Prolog auxiliares, os quais são compartilhados pelos procedimentos pertencentes ao MAR e ao MCE. Deve ser observado que o usuário não tem acesso aos procedimentos auxiliares do MA.

[4]A descrição do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$foi realizada conjuntamente com Alan Keller Gomes. Assim, a mesma descrição encontra-se também em (Gomes and Monard, 2002).

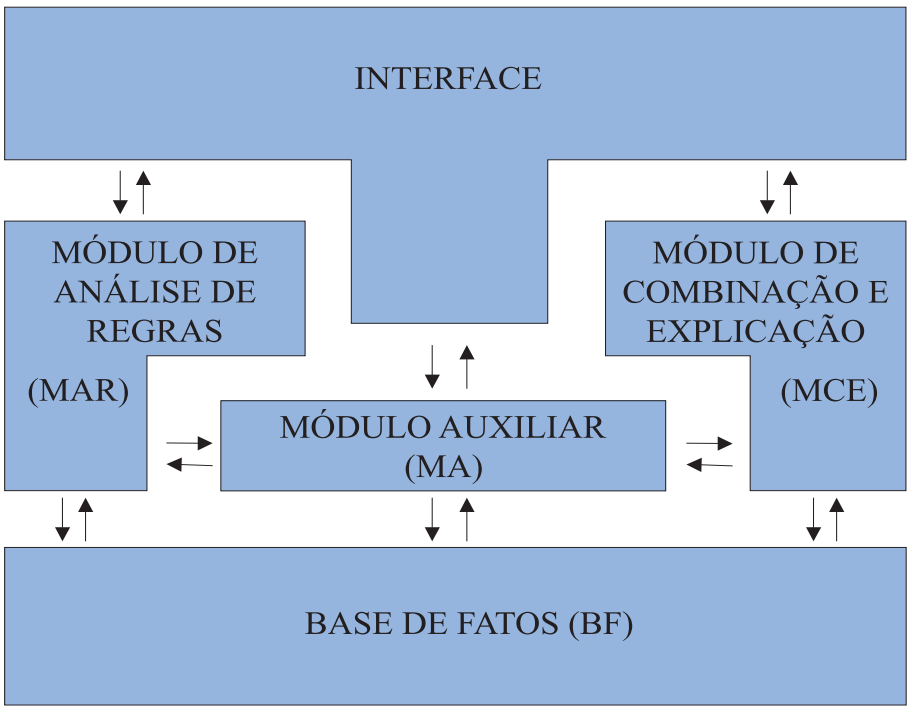

Figura 1: Arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

A Base de Fatos (BF) é utilizada para armazenar os dados de entrada (um ou mais conjuntos de regras e um único conjunto de exemplos) bem como para armazenar informações adicionais, resultantes da execução de algum procedimento ativado pelo usuário.

É importante observar que o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ é um sistema interativo, guiado pelas necessidades do usuário, ou seja, o usuário pode ativar os procedimentos principais de cada um dos três módulos.

A Tabela 3 mostra os procedimentos principais (`<nome do procedimento>/<aridade>`) acessíveis ao usuário, implementados em cada módulo do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$.

MA	MAR	MCE
`loadRules/0`	`evaluateAllSetOfRulesFrequency/1`	`rco/4`
`loadExamples/0`	`indAttSurp/1`	`classifyExample/2`
`verifyBase/0`	`calculateMeasuresOverAllSetOfRules/1`	`confusionMatrix/2`
	`smallDisjSurpr/3`	`classifyExampleByEnsemble/2`
		`kFoldCrossValidation/4`

Tabela 3: Procedimentos Principais do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Para a maioria desses procedimentos, há um conjunto de pré-condições que devem ser satisfeitas para sua correta execução. Também, vários desses procedimentos apresentam

um conjunto de pós-condições. Essas condições, que são tanto pré-condições para disparar um procedimento quanto pós-condições geradas após a execução de um procedimento, são armazenadas na Base de Fatos.

Com o objetivo de documentar o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, foi por nós desenvolvida uma metodologia específica, descrita a seguir.

4 Metodologia para Documentação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Quando foi iniciada a implementação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ foram pesquisadas na literatura algumas metodologias que pudessem servir para documentar o sistema. Entretanto, nenhuma das metodologias pesquisadas foi suficientemente adequada para atender as especificidades de representação de pré e pós condições de procedimentos em fluxo de execução. Assim, foi decidido propor uma metodologia de representação na forma de um Diagrama de Fluxo de Execução de Procedimentos (PEFD — *Procedures Execution Flow Diagram*), para documentar o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. O PEFD do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ é apresentado na Figura 2.

4.1 Componentes do Diagrama de Fluxo de Execução de Procedimentos

Os componentes do PEFD são 3:

1. Procedimentos;
2. Condições; e
3. Fluxos de Execução.

Cada um desses componentes é descrito a seguir.

4.1.1 Procedimentos

Um procedimento é representado por um retângulo identificado com o nome do procedimento. O módulo ao qual pertence esse procedimento encontra-se no canto superior direito, dentro de um retângulo menor. No caso do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, são 3 os módulos — MA, MAR e MCE. A Figura 3 mostra a representação do procedimento `evaluateAllSetOfRulesFrequency/1` do Módulo de Análise de Regras. Quando não for permitido ao usuário disparar um procedimento, o retângulo que se refere a esse procedimento encontra-se preenchido com um tom de cinza claro. Por exemplo, pode-se visualizar na Figura 2 que os procedimentos `listOfExamples/1` e `dictionary/2` não são acessíveis ao usuário.

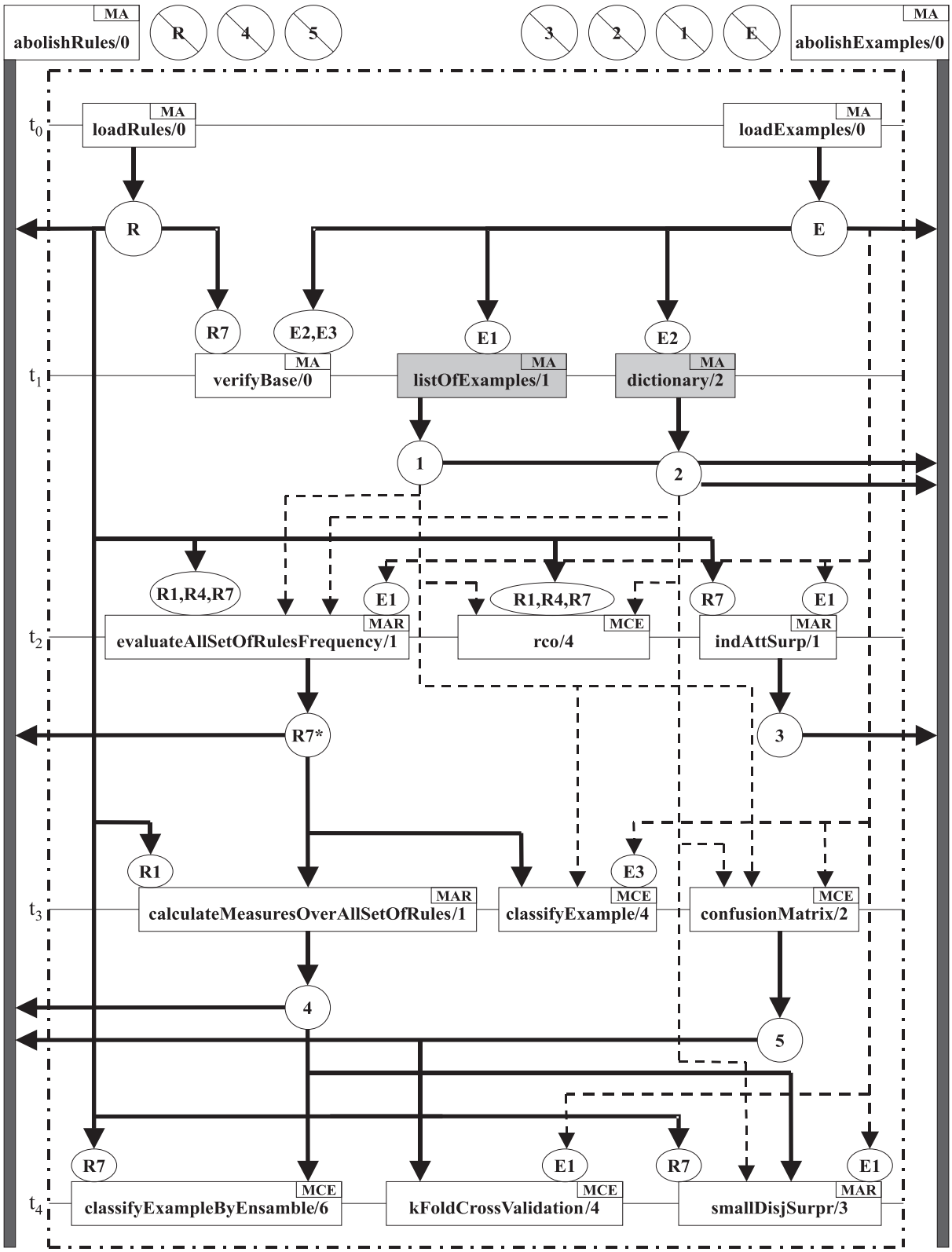

Figura 2: Diagrama de Fluxo de Execução de Procedimentos do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Figura 3: Exemplo de Representação de um Procedimento

4.1.2 Condições

As pré-condições necessárias para a execução de um procedimento, armazenadas previamente na Base de Fatos, bem como as pós-condições de um procedimento, armazenadas na BF após a sua execução, são representadas por círculos identificados por letras, letras seguidas de números, ou números, onde:

- `<letra>` nomeia um conjunto de condições;
- `<letra><números>` nomeia a condição `<números>` que pertence ao conjunto de condições nomeado `<letra>`; e
- `<número>` nomeia uma única condição que não pertence a nenhum conjunto de condições.

No $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, essas condições são:

`<letra>:=E|R` onde `E={E1,E2,E3}` e `R={R1,R2,R3,R4,R5,R6,R7}`

`<número>:=1|2|3|4|5`

O conjunto de condições `E` e `R` referem-se, respectivamente, a condições (informações) relacionadas ao conjunto de exemplos e ao conjunto de regras, todas elas na Sintaxe Padrão Prolog (Gomes et al., 2002). A Figura 4 ilustra a representação do conjunto de condições `E`.

Figura 4: Exemplo de Representação de um Conjunto de Condições

Uma descrição dos fatos relacionados a cada condição que pertence ao conjunto de condições `E` é encontrada na Tabela 4.

Analogamente, uma descrição dos fatos relacionados a cada condição que pertence ao conjunto de condições `R` é encontrada na Tabela 5.

`E1: ex/2`	Representa cada um dos exemplos do conjunto de dados.
`E2: feature/3`	Representa informações de cada um dos atributos do conjunto de dados.
`E3: classFeature/1`	Identifica qual o atributo classe do conjunto de dados.

Tabela 4: Condições Pertencentes ao Conjunto de Condições `E`

`R1: inducer/2`	Identifica o algoritmo de AM utilizado para induzir as regras.
`R2: inputFile/2`	Identifica o arquivo onde encontram-se armazenadas as regras originais induzidas pelo algoritmo de AM identificado por `R1`.
`R3: dAte/2`	Identifica data e hora em que as regras foram induzidas.
`R4: evaluatedAs/2`	Indica a forma de avaliação de regras utilizada pelo algoritmo de AM identificado por `R1`.
`R5: nameFile/2`	Identifica o arquivo `.names` relacionado ao conjunto de exemplos utilizado para induzir as regras.
`R6: dataFile/2`	Identifica o arquivo `.data` relacionado ao conjunto de exemplos utilizado para induzir as regras.
`R7: rule/6`	Representa cada uma das regras induzidas pelo algoritmo de AM utilizado.

Tabela 5: Condições Pertencentes ao Conjunto de Condições `R`

Quando é necessário representar um subconjunto de um conjunto de condições, esse subconjunto é identificado pelas várias condições que o compõe, separadas por vírgula. Um exemplo pode ser visualizado na Figura 5, onde é ilustrado o subconjunto {`R1`, `R4` e `R7`} de `R`.

Figura 5: Exemplo de Representação de um Subconjunto de Condições

Finalmente, na Tabela 6, segue uma descrição das condições `1`, `2`, `3`, `4` e `5`.

`1: listOfExamples/1`	Representa uma lista de identificadores (número) de cada exemplo pertencente ao conjunto de exemplos.
`2: auxDic/2`	Representa um dicionário binário com informações sobre os atributos do conjunto de exemplos. Este dicionário é utilizado para minimizar a busca pelos atributos, melhorando assim o tempo de execução dos procedimentos.
`3: infoC/1`	Representa o ganho de informação do atributo classe do conjunto de exemplos.
`4: ruleInfo/3`	Representa os valores obtidos no cálculo das medidas de avaliação de regras utilizadas.
`5: cm/2`	Representa a matriz de confusão, para um determinado conjunto de regras, dado o conjunto de exemplos presente na base.

Tabela 6: Condições `1`, `2`, `3`, `4` e `5`

Foram também definidos os seguintes dois símbolos de representação do estado de uma condição na BF em um dado instante:

`*`: para indicar uma pré-condição atualizada pelo procedimento. Isto é, a pré-condição aparece também como pós-condição do procedimento, mas com diferentes informações. Por exemplo, na Figura 2, a pré-condição `R7` será atualizada para a pós-condição `R7*` após a execução do procedimento `evaluateAllSetOfRulesFrequency/1`.

`\`: indica a eliminação ou inexistência da condição na Base de Fatos. Um exemplo é a inexistência ou eliminação do conjunto de condições `R` que aparece ao lado do procedimento `abolishRules/0` na Figura 2. Nesse caso, o conjunto de condições `R` não existe no estado inicial da base ou foi excluído após a execução de `abolishRules/0`. O mesmo vale para as condições `4` e `5`.

Além disso, pós-condições de alguns procedimentos podem ser pré-condições para outros procedimentos. Assim, a execução desses procedimentos deve obedecer um fluxo temporal, detalhado a seguir.

4.1.3 Fluxo de Execução

O fluxo temporal é representado através de linhas de tempo $t_0, t_1, t_2, ..., t_q$ as quais são somente aplicáveis a procedimentos representados dentro de um retângulo tracejado[5]. Procedimentos fora desse retângulo tracejado são *atemporais*, ou seja, podem ser executados a qualquer momento.

No caso do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, foram identificadas quatro linhas de tempo, ou seja, t_0, t_1, t_2, t_3 — Figura 2. Nessa figura, pode ser observado que os procedimentos `abolishRules/0` e `abolishExamples/0` encontram-se fora do retângulo tracejado, o que indica que eles podem ser executados a qualquer momento. Em outras palavras, são procedimentos atemporais.

Considerando os procedimentos representados dentro do retângulo tracejado, pode-se observar que os procedimentos `loadRules/0` e `loadExamples/0` na linha t_0 devem ser executados antes que os procedimentos `verifyBase/0`, `listOfExamples/1` e `dictionary/2`, na linha t_1. Analogamente, esses procedimentos da linha t_1 devem ser executados antes que os procedimentos da linha t_2, e assim sucessivamente.

Outro aspecto a ser observado é que podem ocorrer relacionamentos entre um conjunto de condições e um procedimento, ou entre um conjunto de condições e um subconjunto desse conjunto de condições. Esses relacionamentos são representados por meio de uma linha, contínua ou tracejada[6], denominada *barramento*. Existem três tipos de barramento, ilustrados na Figura 6, e explicados a seguir:

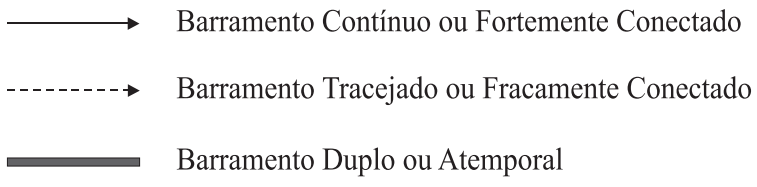

Figura 6: Tipos de Barramento

contínuo ou fortemente conectado indica que, se uma pré-condição for apagada da Base de Fatos, todas as pós-condições oriundas de procedimentos relacionados com essa pré-condição devem também ser apagadas da BF.

[5]tracejado do tipo $\{- \cdot - \cdot -\}$.
[6]tracejado do tipo $\{- - -\}$.

tracejado ou fracamente conectado indica que, se uma pré-condição for apagada da BF, as pós-condições oriundas de procedimentos relacionados com essa pré-condição não são apagadas da BF.

duplo ou atemporal indica que, ao executar um procedimento relacionado com esse tipo de barramento, todas as condições relacionadas serão apagadas da BF indistintamente.

Quando se relaciona um conjunto de condições a um procedimento, pode-se ter a necessidade de especificar qual é o subconjunto de condições que realmente faz parte do conjunto de pré-condições do procedimento. Para se detalhar esse tipo de informação, substitui-se o relacionamento entre o conjunto de condições e o procedimento por um relacionamento entre o conjunto de condições e o subconjunto de pré-condições. Posiciona-se o círculo que representa esse subconjunto de pré-condições sobre o retângulo que representa o procedimento. Por exemplo, na Figura 2, o procedimento `verifyBase/0` somente necessita da pré-condição `R7` do conjunto de condições `R` e do subconjunto {`E2, E3`} de condições do conjunto de condições `E`.

4.2 Descrição do Fluxo de Execução do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Para um melhor entendimento do PEFD do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$(Figura 2), aqui é apresentada uma descrição do fluxo de execução dos procedimentos implementados nesse sistema.

Na parte superior da Figura 2 está representado o estado inicial da Base de Fatos. O estado inicial é caracterizado pela ausência de todas as condições que podem estar presentes na BF, ou também, pela exclusão de todas essas condições. Dessa forma, os procedimentos `abolishRules/0` e `abolishExamples/0` são responsáveis por excluir, respectivamente, as condições `R, 4` e `5` e `E, 1, 2 e 3`. A ausência ou a exclusão dessas condições é representada por meio do traço que corta o círculo contendo a identificação de cada condição.

Considerando que nenhuma condição está presente na Base de Fatos, o início do fluxo de execução do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ dá-se no instante t_0, quando podem ser executados seguidamente, em qualquer ordem, os procedimentos `loadRules/0` e `loadExamples/0`. Por exemplo, na execução de `loadRules/0` o conjunto de condições `R` é gravado na BF. Esse conjunto de condições é fortemente conectado ao barramento atemporal e pode ser removido da BF somente pela execução do procedimento `abolishRules/0`. Ao ser executado o procedimento `loadExamples/0`, o conjunto de condições `E` é gravado na BF. Esse conjunto de condições também é fortemente conectado ao barramento atemporal e apenas a execução do procedimento `abolishExamples/0` pode apagar esse conjunto de condições da BF.

Seguindo para o instante t_1, os procedimentos `verifyBase/0`, `listOfExamples/1` e `dictionary/2` podem ser executados seguidamente. O procedimento `verifyBase/0` tem como pré-condições subconjuntos de `R` e `E` aos quais encontram-se fortemente conectados. Especificamente, a pré-condição `R7` do conjunto de condições `R` e o subconjunto de condições

E2 e E3 do conjunto de condições E. A conexão forte indica que essas pré-condições específicas serão apagadas da BF se o conjunto R e E de condições for apagado da BF.

Ainda no instante t_1, os procedimentos listOfExamples/1 e dictionary/2 podem ser executados seguidamente em qualquer ordem. Como pode ser observado, os retângulos que identificam esses procedimentos estão preenchidos com um tom de cinza claro, o que significa que não podem ser acessados pelo usuário. Após serem executados, esses procedimentos gravam na BF, respectivamente, as condições 1 ou 2. Essas duas condições estão fortemente conectadas ao barramento atemporal, portanto elas podem ser apagadas somente através da execução do procedimento abolishExamples/0.

Partindo para o instante t_2, os procedimentos rco/4, evaluateAllSetOfRulesFrequency/1 e indAttSurp/1 podem ser executados em qualquer ordem. Esses procedimentos estão fracamente conectados às pré-condições 1 e 2 e fortemente conectados com os conjuntos de condições R e E.

Por exemplo, o procedimento evaluateAllSetOfRulesFrequency/1 está fracamente conectado às pré-condições 1 e 2, fortemente conectado à condição R através do subconjunto de pré-condições R1, R4, R7 e fracamente conectado com a pré-condição E através da pré-condição E1. Após sua execução, esse procedimento modifica a pré-condição R7 gravando a pós-condição R7* na BF. Isso implica que, se 1, 2 ou E forem apagadas da BF, a pós-condição R7* continua válida. R7* só pode ser apagada pela execução do procedimento abolishRules/0.

O fluxo de execução do PEFD do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ nos instantes t_3 e t_4 é semelhante ao descrito no instante t_2.

Em qualquer dos instantes $t_0, ..., t_4$ os procedimentos atemporais abolishRules/0 e abolishExamples/0 podem ser executados. Por exemplo, se abolishExamples/0 for executado no instante t_2, todas as condições presentes na BF até aquele instante e conectadas ao barramento atemporal serão excluídas — no caso, serão excluídas as condições E, 1 e 2. Deve ser observado que os procedimentos atemporais abolishRules/0 e abolishExamples/0 retornam a BF ao seu estado inicial, ou seja, as condições ligadas a esses procedimentos (respectivamente R, 4 e 5 e E, 1, 2 e 3) são apagadas da BF por meio do barramento fortemente conectado.

Na seção seguinte é apresentado o Módulo de Combinação e Explicação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ — Figura 1 na página 5. O Módulo de Análise de Regras é detalhado em (Gomes and Monard, 2002).

5 Módulo de Combinação e Explicação

Neste módulo estão implementados cinco procedimentos principais relacionados com:

1. *Classificação de Exemplos* — `classifyExample/4`;
2. *Determinação da Matriz de Confusão* — `confusionMatrix/2`;
3. *Explicação de Ensembles* — `classifyExampleByEnsemble/6`;
4. *Combinação de Classificadores Simbólicos* — `rco/4` e `kFoldCrossValidation/4`.

descritos nas próximas seções. A instanciação[7] de cada um dos i argumentos de um procedimento de aridade i é descrita utilizando a notação padrão para documentar programas Prolog, ou seja,

+ <**arg-i**> o argumento deve estar instanciado;

- <**arg-i**> o argumento deve estar desinstanciado, ou seja, é uma variável livre;

? <**arg-i**> o argumento pode estar instanciado ou desinstanciado.

5.1 Classificação de Exemplos

Dada uma hipótese **h** e um exemplo **x**, o procedimento `classifyExample/4` é responsável pela classificação desse exemplo segundo a hipótese **h**. Os quatro argumentos são:

+ <**arg-1**> identificador da hipótese;

+ <**arg-2**> exemplo **x**;

- <**arg-3**> classificação do exemplo **x**;

- <**arg-4**> regras de **h** que cobrem o exemplo **x**.

Sabendo-se que $\mathbf{h} = \{R_1, ..., R_{NR}\}$, existem 3 formas diferentes de avaliação das regras que constituem **h** (Baranauskas and Monard, 2000): *ordered*, *unordered* ou *inter-class*. A forma de avaliação é determinada pelo algoritmo de AM que induziu **h**. Assim, após determinada a forma de avaliação, através do `<arg-1>`, `classifyExample/4` ativa o procedimento responsável pela avaliação correspondente.

[7] Na realidade, a instanciação dos argumentos dos procedimentos aqui descrita refere-se à instanciação na execução padrão realizada por um usuário do sistema. Entretanto, para o implementador do sistema, a instanciação do tipo `+` pode, *às vezes*, ser substituída por uma instanciação do tipo `?`, e a do tipo `-` pode *sempre* ser substituída por uma instanciação do tipo `?`. Esta característica de Prolog permite verificar resultados específicos além de auxiliar ao implementador a realizar rapidamente a depuração dos procedimentos.

No caso de avaliação *ordered*, existe um **else** implícito entre as regras induzidas pelo algoritmo de AM. Assim, a avaliação *ordered* é muito simples: o conjunto de regras da hipótese **h** deve ser varrido até encontrar a primeira regra R cujo *Corpo* cobre o exemplo $\mathbf{x}$[8]. O Algoritmo 1 descreve a forma de avaliação *ordered*.

Algoritmo 1 Avaliação *ordered*

Pré-condições: **h** $=\{R_1, ...R_{NR}\}$;
x: exemplo a ser classificado.

1: **procedure** classifyExampleOrdered(**h**,**x**)
2: $RuleFired$:=false;
3: u:=1;
4: **while** not($RuleFired$) and $u \leq NR$ **do**
5: **if** cobre($Corpo(R_u)$,**x**) **then**
6: $RuleFired$:=true;
7: **else**
8: u:=u + 1;
9: **end if**
10: **end while**
11: $Classification$:=$Cabeça(R_u)$;
12: $Rules$:=$\{R_u\}$;
13: **return** $Classification$,$Rules$

O Algoritmo 2 descreve a forma de avaliação *unordered* por nós realizada. Ele varre todas as regras da hipótese **h** e inclui no conjunto $Rules$ as regras R cujo *Corpo* não vazio cobre o exemplo **x**, ou seja, não é considerada a regra *default*[9]. Para encontrar a classificação do exemplo, na implementação por nós realizada, consideramos que a regra do conjunto $Rules$ com maior cobertura define a classe do exemplo **x**. No caso do conjunto $Rules$ ser o conjunto vazio, então o exemplo é classificado pela regra *default*.

Deve ser observado que esta não é a forma de avaliação de regras *unordered* utilizada pelo algoritmo $\mathcal{CN}2$, por exemplo. Entretanto, os resultados esperados são similares para a maioria dos conjuntos de dados naturais e do mundo real.

Finalmente, será tratada a forma de avaliação *interclass*. Quando a avaliação de regras de uma hipótese é *interclass*, a avaliação é do tipo *ordered* entre as classes, mas a avaliação do conjunto de regras que prediz a mesma classe é *unordered*. Por exemplo, a Figura 7 mostra que, para cada classe, existe um conjunto de subconjuntos de regras[10] $\mathbf{h}' = \{\mathbf{h}'_1, ..., \mathbf{h}'_{NCl}\}$.

[8] Observar que qualquer que seja a forma de avaliação, sempre existirá uma classificação para o exemplo **x** devido à existência da regra *default*

[9] A regra *default* de uma hipótese somente é disparada quando nenhuma outra regra da hipótese cobre o exemplo a ser classificado. É uma regra especial cujo *Corpo* é vazio e sua *Cabeça* é definida geralmente pela classe com maior freqüência no conjunto de exemplos utilizado para a construção dessa hipótese

[10] **h** e **h**$'$ diferem unicamente na regra *default*, ou seja, $\mathbf{h} = \mathbf{h}' - \{default\}$. Também é suposto que R_u de **h** é a regra *default*.

Algoritmo 2 Avaliação *unordered*

Pré-condições: $\mathbf{h} = \{R_1, ...R_{NR}\}$;
$\mathbf{x}$: exemplo a ser classificado.

1: **procedure** classifyExampleUnordered($\mathbf{h}$,$\mathbf{x}$)
2: $Rules = \emptyset$;
3: **for** u:=1 to NR **do**
4: **if** cobre($Corpo(R_u)$,$\mathbf{x}$) and not default_rule(R_u) **then**
5: $Rules$:=$Rules \cup R_u$;
6: **end if**
7: **end for**
{disparar a regra *default* - nenhuma outra regra foi disparada}
8: **if** $Rules = \emptyset$ **then**
9: $Rules$:={hypothesis_default_rule($\mathbf{h}$)};
10: **end if**
11: $Rule$:= regra com maior medida de cobertura em $Rules$;
12: $Classification$:=$Cabeça(Rule)$;
13: **return** $Classification$,$Rules$

É importante observar que existe um **else** implícito entre cada um desses subconjuntos de regras e que esses subconjuntos de regras são formados conforme a ordem em que o algoritmo de AM induz as classes que compõem a hipótese $\mathbf{h}$. Como cada elemento de $\mathbf{h}'$, ou seja, $\mathbf{h}'_1, ..., \mathbf{h}'_{NCl}$, é uma hipótese (conjunto de regras) com regras que predizem a mesma classe e avaliadas de forma *unordered*, cada um desses elementos deve ser fornecido ao algoritmo de avaliação *unordered*. O processo pára assim que o algoritmo encontrar um conjunto de regras $Rules$ não vazio para algum elemento $\mathbf{h}'_v \in \mathbf{h}', v = 1, ..., NCl$. Caso contrário, se todo o conjunto $\mathbf{h}'$ for varrido e nenhum elemento de $\mathbf{h}'$ satisfizer a condição $Rules \neq \emptyset$, para algum $v = 1, ..., NCl$, o exemplo $\mathbf{x}$ é classificado segundo a regra *default* de $\mathbf{h}$.

É importante observar que todas as formas de avaliação retornam a classificação do exemplo $\mathbf{x}$ e o conjunto de regras $Rules$ com todas as regras de $\mathbf{h}$ que cobrem o exemplo $\mathbf{x}$.

5.2 Determinação da Matriz de Confusão

Dada uma hipótese $\mathbf{h}$, um conjunto de exemplos $S = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$ e o conjunto de classes $C = \{C_1, ..., C_{NCl}\}$ preditas por $\mathbf{h}$, o procedimento `confusionMatrix/2` é responsável por calcular a matriz de confusão CM da hipótese $\mathbf{h}$. Os dois argumentos do procedimento `confusionMatrix/2` são:

+ <**arg-1**> identificador da hipótese;

+ <**arg-2**> conjunto de exemplos S.

$$
\begin{array}{ll}
\mathbf{h}'_1 & \begin{cases} R_1 : Corpo \text{ then CLASS } = \alpha \\ R_2 : Corpo \text{ then CLASS } = \alpha \end{cases} \\
 & \textbf{else} \\
\mathbf{h}'_2 & \begin{cases} R_3 : Corpo \text{ then CLASS } = \beta \\ R_4 : Corpo \text{ then CLASS } = \beta \\ R_5 : Corpo \text{ then CLASS } = \beta \end{cases} \\
 & \textbf{else} \\
\vdots & \vdots \\
\mathbf{h}'_{NCl} & \begin{cases} R_{u-2} : Corpo \text{ then CLASS } = \gamma \\ R_{u-1} : Corpo \text{ then CLASS } = \gamma \end{cases}
\end{array}
$$

com $\alpha, \beta, \gamma \in C = C_1, ..., C_{NCl}$.

Figura 7: Subconjuntos de Regras — Forma de Avaliação *Interclass*

O Algoritmo 3 descreve como esse cálculo é realizado. O procedimento primeiramente inicializa todos os elementos da matriz CM com valor zero. A dimensão da matriz CM é NClxNCl. Depois, para cada exemplo rotulado $(\mathbf{x}_i, y_i)$, classifica-se $\mathbf{x}_i$ com o procedimento `classifyExample/4`, o qual retorna uma classificação $Classification$ e, assim, para $y_i = C_m$ e $Classification = C_n$, o elemento CM_{mn} da matriz é incrementado de 1.

5.3 Explicação de *Ensembles*

Dado um conjunto de hipóteses $\mathcal{H} = \{\mathbf{h}_1, ..., \mathbf{h}_L\}$, um exemplo a ser classificado $\mathbf{x}$ e uma opção $Measure$, o procedimento `classifyExampleByEnsemble/6` é responsável por decidir a classificação de $\mathbf{x}$ dadas as classificações feitas por $\mathbf{h}_1, ..., \mathbf{h}_L$. Se $Measure$ não estiver instanciado, o procedimento classifica $\mathbf{x}$ com votação por maioria sem fornecer uma explicação para essa classificação. Se $Measure$ for instanciado com uma das medidas de avaliação de regras descritas na Tabela 2 na página 4 então, dentre as regras do conjunto de hipóteses $\mathcal{H}$ que cobrem $\mathbf{x}$, a regra que tiver a melhor medida é utilizada para classificar $\mathcal{H}$. Além disso, o procedimento retorna todas as regras em $\mathcal{H}$ que cobrem $\mathbf{x}$ tanto se $Measure$ estiver instanciado ou se estiver desinstanciado. Os seis argumentos do procedimento `classifyExampleByEnsemble/6` são:

+ <**arg-1**> lista de identificadores de hipóteses;

+ <**arg-2**> exemplo $\mathbf{x}$;

? <**arg-3**> medida de avaliação de regra a ser usada para classificação (*majority* no caso de estar desinstanciado);

Algoritmo 3 Matriz de Confusão

Pré-condições: **h**: hipótese;
$S = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$: conjunto de exemplos;
$C = \{C_1, ..., C_{NCl}\}$: conjunto de classes.

1: **procedure** confusionMatrix(**h**,S,C)
2: **for** $m := 1$ to NCl **do**
3: **for** $n := 1$ to NCl **do**
4: CM_{mn}:=0;
5: **end for**
6: **end for**
7: **for** $i := 1$ to N **do**
8: $Classification$:=classifyExample(**h**,$\mathbf{x}_i$);
9: m:=index(C,y_i);
10: n:=index(C,$Classification$);
11: CM_{mn}:=CM_{mn} + 1;
12: **end for**
13: **return** CM

- <**arg-4**> classificação do exemplo;

- <**arg-5**> a regra que classifica o exemplo segundo a instanciação do `<arg-3>`. 0 (zero) se `<arg-3>` estiver desinstanciado.

- <**arg-6**> conjunto de regras que cobrem **x**.

O Algoritmo 4 descreve como a decisão de classificação do exemplo **x** é tomada. Primeiro o algoritmo executa o procedimento `classifyExample/4` para **x** e cada $\mathbf{h}_l$. Depois, se $Measure$ estiver desinstanciado, toma a classe com maior quantidade de votos como sendo a classificação de **x**, ou seja, faz votação por maioria para decidir a classificação de **x**. Caso contrário, toma, dentre as regras disparadas nas hipóteses $\mathbf{h}_l, l = 1, ..., L$, a regra *Rule* com maior medida $Measure$. A classificação de **x** é dada pela *Cabeça* de *Rule*. Em ambos os casos, o algoritmo retorna a classificação de **x** e todas as regras que cobrem **x**.

5.4 Combinação de Classificadores Simbólicos

Os dois procedimentos relacionados com combinação de classificadores são:

1. `rco/4`; e
2. `kFoldCrossValidation/4`

ambos descritos a seguir.

Algoritmo 4 Explicação de *Ensembles*

Pré-condições: $\mathcal{H} = \mathbf{h}_l, l = 1, ..., L$: conjunto de hipóteses;
$(\mathbf{x}_i, y_i)$: exemplo a ser classificado;
$Measure$: medida a ser considerada na classificação ou uma decisão com votação por maioria.

1: **procedure** classifyExampleByEnsemble($\mathcal{H}$,$(\mathbf{x}_i, y_i)$,$Measure$)
2: $Classes := \emptyset$;
3: $Rules := \emptyset$;
4: **for** $l := 1$ to L **do**
5: classifyExample($\mathbf{h}_l$,$\mathbf{x}$,$Class_l$,$HipRules_l$);
6: $Classes := Classes \bigcup Class_l$;
7: $Rules := Rules \bigcup HipRules_l$;
8: **end for**
9: **if** var($Measure$) {**true** se $Measure$ não estiver instanciado} **then**
10: $Classification$:=majority($Classes$); {Decidir a maioria dos votos}
11: **else**
12: $Rule$:=max($Rules$,$Measure$); {Escolher a regra com a melhor $Measure$}
13: $Classification$:=$Cabeça$($Rule$); {Classificar o exemplo com a regra $Rule$}
14: **end if**
15: **return** $Classification$,$Rules$

5.4.1 Algoritmo $\mathcal{RCO}$

Dado um conjunto de hipóteses $\mathcal{H} = \{\mathbf{h}_1, ..., \mathbf{h}_L\}$, um conjunto de exemplos de treinamento $S = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$ e uma medida de avaliação de regra $Measure$, o procedimento `rco/4` constrói uma nova hipótese como uma combinação das regras $R \in \mathbf{h}_l, l = 1, ..., L$, segundo a medida $Measure$. Os quatro argumentos do procedimento `rco/4` são:

+ <**arg-1**> lista de identificadores de hipóteses;

+ <**arg-2**> uma medida de avaliação de regra;

- <**arg-3**> identificador da hipótese criada;

- <**arg-4**> lista de exemplos não cobertos pela nova hipótese.

O Algoritmo 5 mostra como essa nova hipótese é construída. O algoritmo faz inicialmente uma cópia das regras de todas as hipóteses e cria o cabeçalho da hipótese a ser construída. Nesse cabeçalho, constam os fatos relacionados às condições `R1` até `R6`, descritas na Tabela 5 na página 9. Depois do cabeçalho criado, o algoritmo segue os seguintes passos:

1. Verifica se o conjunto de exemplos S está vazio. Se sim, a iteração pára;

Algoritmo 5 $\mathcal{RCO}$

Pré-condições: $\mathcal{H} = \mathbf{h}_l, l = 1, ..., L$: conjunto de hipóteses;
$S = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$: conjunto de exemplos;
$Measure$: medida de regra.

1: **procedure** rco($\mathcal{H}$,S,$Measure$)
2: Fazer cópia das regras das L hipóteses em $Rules$;
3: S':=S; {Cópia dos exemplos originais}
4: Criar o cabeçalho da hipótese $\mathbf{h}_{new}$;
5: **loop**
6: **if** vazio(S) or vazio($Rules$) **then**
7: **exit loop**; {Não há mais exemplos de treinamento ou não há mais regras em $Rules$ para processar}
8: **end if**
9: calculateMeasuresOverAllSetOfRules($Measure$);
10: $Rule$:=max($Rules, Measure$); {Encontrar a regra $Rule$ em $Rules$ com maior medida $Measure$}
11: S_{cov}:=cov(S,$Rule$); {Criar o conjunto de exemplos cobertos pela regra $Rule$}
12: **if** vazio(S_{cov}) **then**
13: **exit loop** {Nenhum exemplo foi coberto pela regra $Rule$}
14: **end if**
15: S:=S - S_{cov}; {Retirar de S os exemplos contidos em S_{cov}}
16: $\mathbf{h}_{new} := \mathbf{h}_{new} \bigcup \{Rule\}$; {Adicionar $Rule$ à hipótese $\mathbf{h}_{new}$}
17: $Rules := Rules - \{Rule\}$; {Retirar $Rule$ de $Rules$}
18: **end loop**
19: $defaultRule$:=calculate_default_rule(S'); {Calcular a regra *default*}
20: $\mathbf{h}_{new} := \mathbf{h}_{new} \bigcup \{defaultRule\}$; {adicioná-la a $\mathbf{h}_{new}$}
21: **return** $\mathbf{h}_{new}$

2. Calcula $Measure$ para todas as regras $R \in \mathbf{h}_l, l = 1, ..., L$;

3. Retira a regra $Rule$, dentre as L hipóteses, com melhor valor $Measure$;

4. Retira do conjunto de exemplos S todos os exemplos que são cobertos pela regra $Rule$. Se nenhum exemplo for coberto, a iteração pára[11];

5. Adiciona a regra $Rule$ à nova hipótese;

6. Volta ao passo inicial.

Ao sair da iteração, segundo qualquer dos dois critérios de parada, o algoritmo constrói a regra *default*. A regra *default* possui *Corpo* vazio e sua *Cabeça* é definida pela classe com maior distribuição dos exemplos do conjunto de exemplos inicial.

Deve ser observado que, além da hipótese construída, o usuário pode visualizar as regras do conjunto de hipóteses que não foram utilizadas pelo algoritmo para construir a nova hipótese. Após, o conjunto de hipóteses original é recuperado.

O algoritmo $\mathcal{RCO}$ (*Rule Combination*) pode ser visualizado na Figura 8, onde se pode observar que os dados de entrada para o algoritmo $\mathcal{RCO}$ consistem de L classificadores e um conjunto de exemplos de treinamento. Com esses dados de entrada, uma nova hipótese é construída e, com um conjunto de teste, pode-se medir a precisão e/ou erro dessa hipótese.

Avaliar hipóteses induzidas ou construídas por algoritmos de AM é fundamental. Para medir a precisão/erro de hipóteses construídas pelo $\mathcal{RCO}$, é utilizada a técnica de estimativa k-fold cross validation, a qual é implementada no procedimento `kFoldCrossValidation/4`. Esse procedimento é descrito a seguir.

5.4.2 Algoritmo k-Fold Cross Validation

Dado um conjunto de hipóteses $\mathcal{H} = \{\mathbf{h}_1, ..., \mathbf{h}_L\}$, um conjunto de treinamento $S = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$, um número de partições K e uma medida de avaliação de regras $Measure$, o procedimento `kFoldCrossValidation/4` é responsável por estimar o erro e e o desvio padrão dp da hipótese construída pelo algoritmo `rco/4`, utilizando a estimativa k-fold cross validation. Os quatro argumentos do procedimento `kFoldCrossValidation/4` são:

+ <**arg-1**> lista de identificadores de hipóteses;

+ <**arg-2**> uma medida de avaliação de regras;

[11] Se a regra *Rule* não cobre nenhum exemplo, isso significa que $Measure$ atingiu seu mínimo (ou máximo) absoluto para todas as regras $R \in \mathbf{h}_l, l = 1, ..., L$ nesse passo da iteração. Por exemplo, no caso da medida ser cobertura ($Measure$ =`covR`), o mínimo valor absoluto é 0(zero), e no caso da medida ser erro ($Measure$ =`errR`), o máximo valor absoluto é 1.

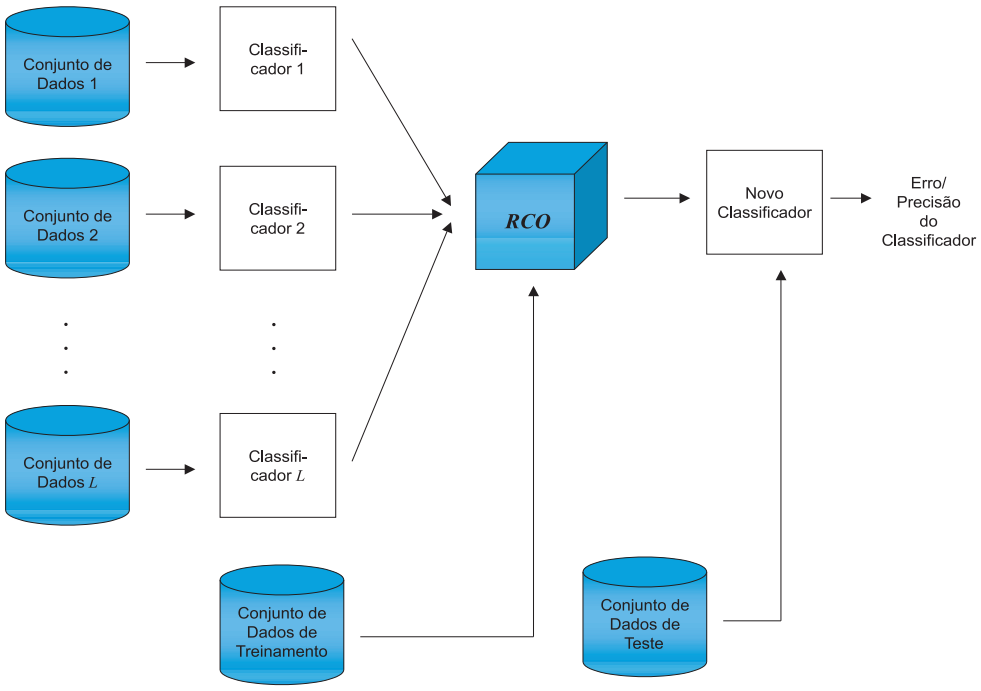

Figura 8: Ilustração do Algoritmo $\mathcal{RCO}$

+ <**arg-3**> número de partições K;

+ <**arg-4**> especifica o dispositivo de saída do algoritmo — console ou arquivo — para mostrar ou gravar os resultados do procedimento.

O Algoritmo 6 descreve como são estimadas as medidas de erro e desvio padrão. O algoritmo cria, inicialmente, K partições do conjunto de treinamento S com aproximadamente $\frac{N}{K}$ exemplos em cada partição. Para cada iteração $k = 1, ..., K$, o algoritmo segue os seguintes passos:

1. Toma-se a k-ésima partição como sendo S_{test} — conjunto de teste — e as $(K-1)$ partições restantes como sendo $S_{training}$ — conjunto de treinamento;

2. Cria-se uma hipótese $\mathbf{h}_k$ utilizando o procedimento que implementa o algoritmo $\mathcal{RCO}$ e o conjunto $S_{training}$;

3. Calcula-se a matriz de confusão para $\mathbf{h}_k$ utilizando o conjunto S_{test};

4. Calcula-se o erro cometido por $\mathbf{h}_k$ utilizando a matriz de confusão já calculada.

Por fim, calcula-se a média $Mean$ dos K erros calculados e seu respectivo desvio padrão SD.

Algoritmo 6 k-Fold Cross Validation

Pré-condições: $\mathcal{H} = \{\mathbf{h}_1, ..., \mathbf{h}_L\}$;
$S = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$: conjunto de exemplos;
K: número de partições;
$Measure$: medida de avaliação de regra.

1: **procedure** kFoldCrossValidation(L,S,K,$Measure$)
2: S_{partic} = cria_particoes(K,S);
3: $Erro := \emptyset$;
4: **for** $k := 1$ to K **do**
5: $S_{test} := S_k | S_k \in S_{partic}$;
6: $S_{training} := S$ - S_k;
7: $\mathbf{h}_k$:= rco(L,$S_{training}$,$Measure$);
8: $ConfusionMatrix_k$:= confusionMatrix($\mathbf{h}_k$,S_{test},C_v);
9: $Erro_k$:= calculo_erro($ConfusionMatrix_k$);
10: $Erro := Erro \bigcup \{Erro_k\}$
11: **end for**
12: $Mean$ = mean($Erro, K$);
13: SD = sd($Erro, Mean, K$);
14: **return** $Mean, SD$

Dado o conjunto de exemplos $S = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$, as K partições são criadas utilizando-se um procedimento que implementa o Algoritmo 7 (Weiss and Kulikowski, 1991). Para cada iteração $k = K, ..., 2$, $\frac{N}{k}$ elementos escolhidos aleatoriamente de S são movidos para S_k.

6 Um Exemplo de Aplicação do MCE

Para melhor ilustrar os procedimentos do Módulo de Combinação e Explicação, nesta seção serão descritos alguns experimentos realizados sobre um conjunto de exemplos artificial cuja verdadeira função f, também denominada de hipótese verdadeira, é dada por

$$f(att_1, att_2) = \begin{cases} o & \text{se } 10 \leq att_1 \leq 20, 40 \leq att_2 \leq 50, \\ o & \text{se } 20 \leq att_1 \leq 30, 20 \leq att_2 \leq 30, \\ o & \text{se } 40 \leq att_1 \leq 50, 30 \leq att_2 \leq 40, \\ x & \text{se } 10 \leq att_1 \leq 20, 20 \leq att_2 \leq 30, \\ x & \text{se } 30 \leq att_1 \leq 40, 10 \leq att_2 \leq 20, \\ x & \text{se } 50 \leq att_1 \leq 60, 30 \leq att_2 \leq 40. \end{cases} \quad (1)$$

para $att_1, att_2 \in \mathbb{R}$, ilustrada na Figura 9. Os retângulos contornados por uma linha

Algoritmo 7 Construção de Partições

Pré-condições: K: número de partições;
$S = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$: conjunto de exemplos.

1: **procedure** cria_particoes(K,S)
2: $N_{aux} = N$;
3: **for** $k := K$ to 2 **do**
4: $S_k := \emptyset$;
5: $N_k = \frac{N_{aux}}{k}$;
6: **for** $i := 1$ to N_k **do**
7: $\mathbf{x}$:= retira_aleatorio(S);
8: $S := S - \{\mathbf{x}\}$;
9: $N_{aux} = N_{aux}$ - 1;
10: $S_k := S_k \bigcup \mathbf{x}$;
11: **end for**
12: **end for**
13: $S_1 = S$; {Colocar os elementos restantes de S em S_1}
14: **return** $\{S_1, ..., S_K\}$

tracejada correspondem à classe o, e os retângulos contornados por uma linha contínua correspondem à classe x.

Dada a verdadeira função f, infinitos conjuntos de exemplos podem ser gerados. Assim, foram gerados 8 conjuntos de exemplos diferentes, denominados $S_1, ..., S_8$. Cada um desses conjuntos possui 120 exemplos, onde o número de exemplos em cada classe é balanceado, ou seja, 50% dos exemplos pertencem à cada classe o e 50% dos exemplos pertencem à cada classe x. Os sete primeiros conjuntos $S_1, ..., S_7$ foram utilizados para induzir 7 (sete) hipóteses, denominadas $\mathbf{h}_1, ..., \mathbf{h}_7$, com dois algoritmos de aprendizado de máquina: o $\mathcal{CN}2$ e o $\mathcal{S}$ee5. O $\mathcal{CN}2$ (Clark and Niblett, 1989) induz hipóteses com regras não-ordenadas e ordenadas (*unordered* e *ordered*), e o $\mathcal{S}$ee5 (Quinlan, 1997) induz árvores de decisão e regras não-ordenadas. As sete hipóteses $\mathbf{h}_1, ..., \mathbf{h}_7$ foram induzidas como regras não-ordenadas, sendo que $\mathbf{h}_1, ..., \mathbf{h}_5$ foram induzidas pelo algoritmo $\mathcal{CN}2$ e $\mathbf{h}_6$ e $\mathbf{h}_7$ foram induzidas pelo algoritmo $\mathcal{S}$ee5, como mostra a Figura 10.

Para ilustrar a forma como as hipóteses induzidas se comportam em relação à verdadeira função f, foram gerados gráficos da seguinte forma:

1. Cada regra é representada por uma região retangular preenchida com um tom de cinza;

2. As regras que classificam um exemplo como sendo da classe o são plotadas como retângulos preenchidos com cinza claro; as regras que classificam um exemplo como sendo da classe x são plotadas como retângulos preenchidos com cinza escuro.

3. Uma hipótese $\mathbf{h}$ composta por NR regras tem suas regras plotadas em ordem inversa,

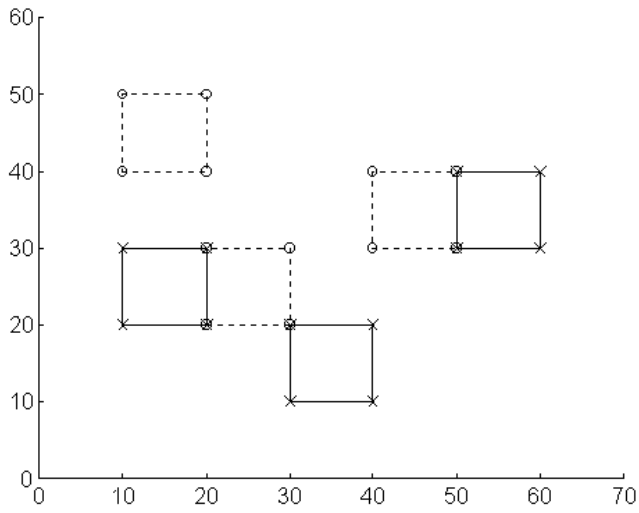

Figura 9: Função Verdadeira f

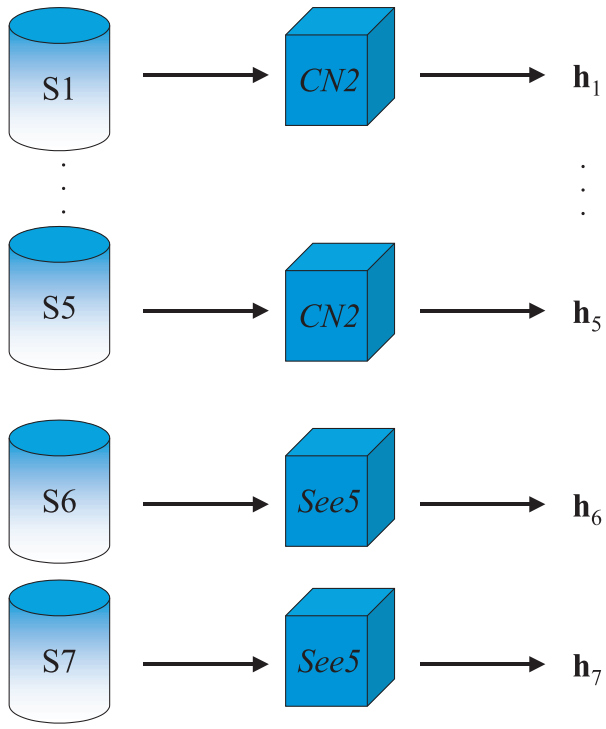

Figura 10: Hipóteses Induzidas por $\mathcal{CN}2$ e $\mathcal{S}$ee5 Utilizadas nos Experimentos

ou seja, a primeira regra a ser plotada é a regra R_{NR} (regra *default*), e a última regra a ser plotada é a regra R_1, havendo, assim, uma sobreposição de retângulos e, portanto, das regras.

Deve ser observado que essa forma de representação facilita a visualização de como as hipóteses induzidas por $\mathcal{CN}2$ e $\mathcal{S}$ee5 e como as hipóteses construídas utilizando o MCE cobrem os exemplos do domínio.

Assim, a forma como as hipóteses induzidas pelo $\mathcal{CN}2$ — $\mathbf{h}_1, ..., \mathbf{h}_5$ — se comportam em relação à função verdadeira f pode ser vista na Figura 11; a forma como as hipóteses induzidas pelo $\mathcal{S}$ee5 — $\mathbf{h}_6$ e $\mathbf{h}_7$ — se comportam em relação à função verdadeira f pode ser vista na Figura 12.

Utilizando as hipóteses induzidas $\mathbf{h}_1, ..., \mathbf{h}_7$ e o conjunto de exemplos S_8, foram realizados 3(três) experimentos diferentes para combinação de regras utilizando o procedimento que implementa o algoritmo $\mathcal{RCO}$. Em cada experimento, foram construídos 6(seis) classificadores diferentes, cada um utilizando um critério (medida) de avaliação de regra, constantes na Tabela 2 na página 4. Essas hipóteses são denominadas $\mathbf{h}_{Exp_\mathtt{accR}}$, $\mathbf{h}_{Exp_\mathtt{covR}}$, $\mathbf{h}_{Exp_\mathtt{satR}}$, $\mathbf{h}_{Exp_\mathtt{sensR}}$, $\mathbf{h}_{Exp_\mathtt{specR}}$ e $\mathbf{h}_{Exp_\mathtt{supR}}$, onde Exp referencia a experiência que está sendo considerada, ou seja, $Exp \in \{1, 2, 3\}$. Para ilustrar o comportamento dessas hipóteses em relação à hipótese verdadeira f, também foram construídos gráficos conforme descrito anteriormente.

São vários os objetivos desses experimentos, dentre eles:

- Verificar qual o erro aparente das hipóteses construídas pelo $\mathcal{RCO}$ e o número de exemplos não cobertos por essas hipóteses, ou seja, exemplos que são somente cobertos pela regra *default*;
- Verificar o erro dessas hipóteses em um conjunto de dados de teste independente S_{teste};
- Verificar as diferenças entre os resultados obtidos pelas hipóteses construídas pelo $\mathcal{RCO}$ com as hipóteses induzidas por $\mathcal{CN}2$ e $\mathcal{S}$ee5 quando todos os conjuntos de exemplos de treinamento são utilizados por esses dois indutores;
- Analisar cada uma dessas hipóteses internamente, ou seja, as regras que constituem essas hipóteses.

O conjunto S_{teste} mencionado foi gerado com 1000(mil) exemplos, também balanceado, independente dos conjuntos $S_1, ..., S_8$ gerados anteriormente. Para realizar as comparações, os experimentos podem ser divididos em duas fases:

Fase 1: Verificar as hipóteses induzidas por $\mathcal{CN}2$ e $\mathcal{S}$ee5 utilizando todos os exemplos em cada experimento.

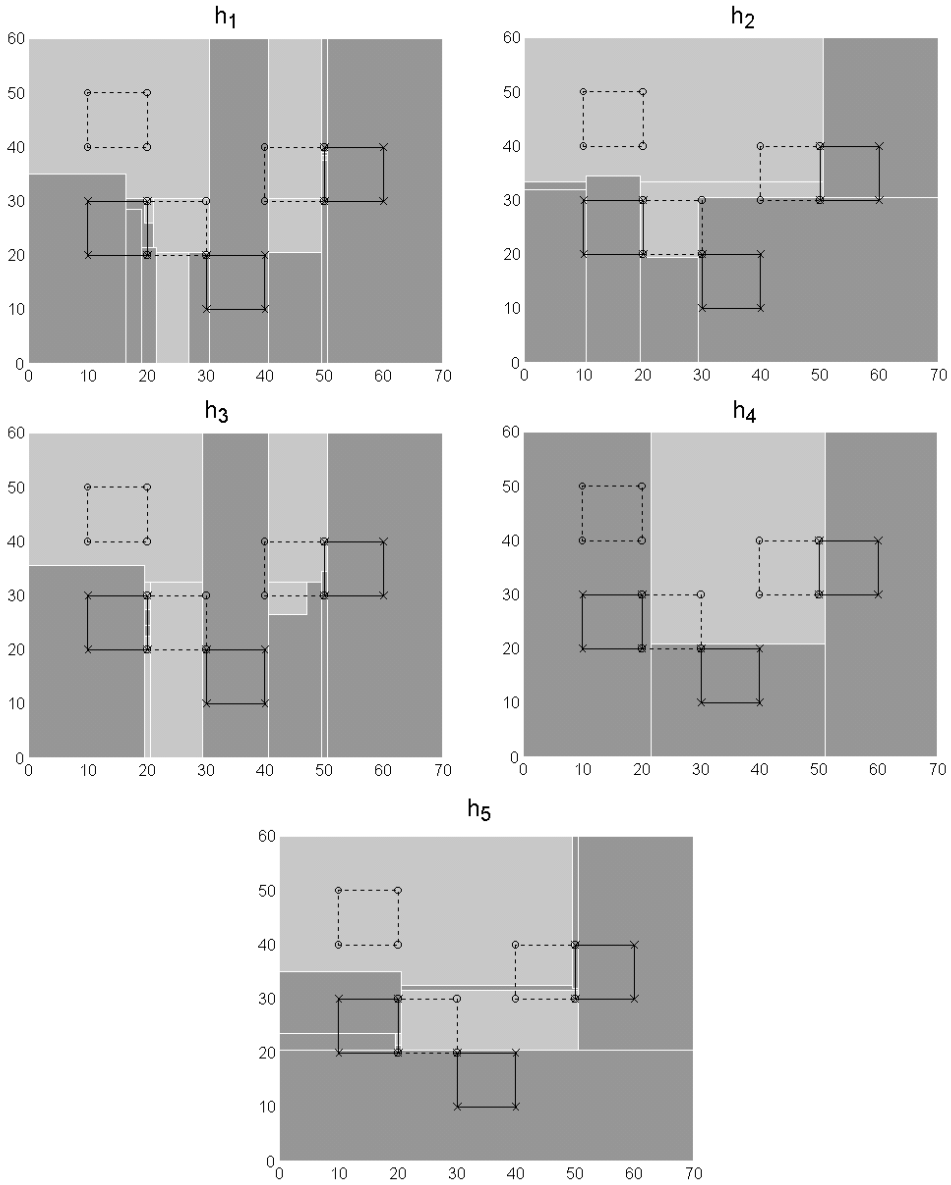

Figura 11: Representação Gráfica das Hipóteses Induzidas por $\mathcal{CN}2$.

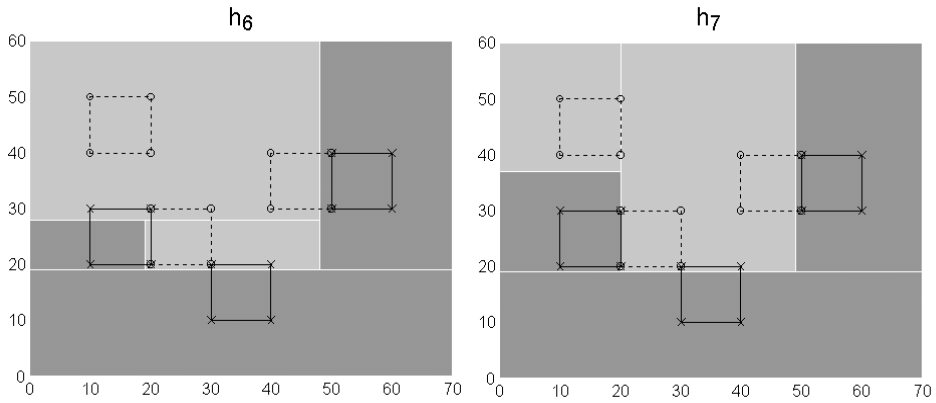

Figura 12: Representação Gráfica das Hipóteses Induzidas por $\mathcal{S}$ee5.

Fase 2: Verificar as hipóteses construídas por $\mathcal{RCO}$.

A seguir, é feita uma descrição de cada um dos três experimentos.

6.1 Descrição do Experimento 1

As duas fases do experimento 1 podem ser visualizadas na Figura 13. A seguir, são mostrados os resultados obtidos nas fases 1 e 2 desse experimento.

Fase 1 Na fase 1, foram induzidas 2(duas) hipóteses, uma com o $\mathcal{CN}2$ e outra com o $\mathcal{S}$ee5, utilizando como conjunto de exemplos a união dos conjuntos S_1, S_2, S_3 e S_8. Para cada uma das hipóteses, foi medido o erro aparente e utilizou-se a técnica de 10-fold cross validation para estimar a taxa de erro e desvio padrão de cada hipótese, bem como o número de regras induzidas utilizando todos os exemplos. Deve ser observado que nesta primeira fase, ambos os indutores têm acesso a todos os exemplos utilizados no experimento. Assim, poderia se supor que cada indutor induziu a hipótese mais precisa nesta primeira fase do experimento. Os resultados obtidos e o número de regras induzidas para cada hipótese são mostrados na Tabela 7.

Algoritmos	Subconjunto/ # Ex. Treinamento	Erro Aparente	$e \pm dp$	# de Regras Induzidas
$\mathcal{CN}2$	$S_1 \bigcup S_2 \bigcup S_3 \bigcup S_8/480$	1.40%	4.46% ± 0.85%	15
$\mathcal{S}$ee5	$S_1 \bigcup S_2 \bigcup S_3 \bigcup S_8/480$	2.20%	3.30% ± 0.90%	7

Tabela 7: Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{S}$ee5 (Fase 1 — Experimento 1)

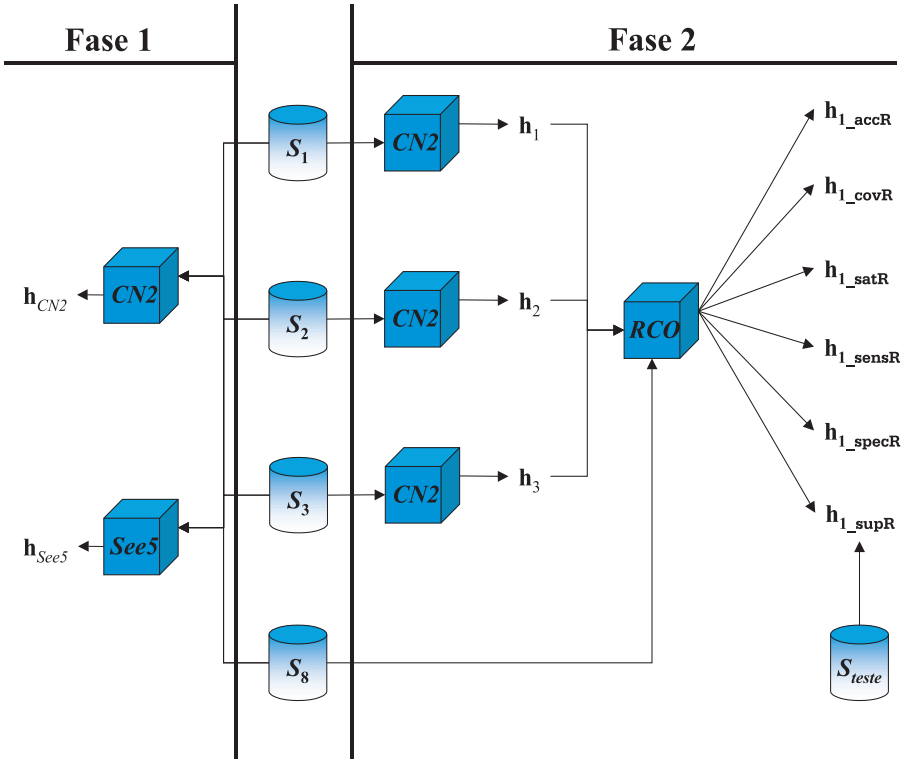

Figura 13: Ilustração do Experimento 1.

Fase 2 Na fase 2, foram construídas as hipóteses $\mathbf{h}_{1_\texttt{accR}}$, $\mathbf{h}_{1_\texttt{covR}}$, $\mathbf{h}_{1_\texttt{satR}}$, $\mathbf{h}_{1_\texttt{sensR}}$, $\mathbf{h}_{1_\texttt{specR}}$ e $\mathbf{h}_{1_\texttt{supR}}$. Para isso, foram utilizadas as hipóteses $\mathbf{h}_1$, $\mathbf{h}_2$ e $\mathbf{h}_3$ e o conjunto de exemplos S_8 — Figura 13. As hipóteses $\mathbf{h}_1$, $\mathbf{h}_2$ e $\mathbf{h}_3$ foram induzidas utilizando, respectivamente, os conjuntos de exemplos S_1, S_2 e S_3. O erro aparente de cada hipótese e o número de regras induzidas bem como a estimativa da taxa de erro e desvio padrão utilizando a técnica de 10-fold cross-validation são mostrados na Tabela 8. Para as hipóteses $\mathbf{h}_{1_\texttt{accR}}$, $\mathbf{h}_{1_\texttt{covR}}$, $\mathbf{h}_{1_\texttt{satR}}$, $\mathbf{h}_{1_\texttt{sensR}}$, $\mathbf{h}_{1_\texttt{specR}}$ e $\mathbf{h}_{1_\texttt{supR}}$ construídas, a Tabela 9 mostra o número de regras selecionadas (inclusive a regra *default*), o número de exemplos que restaram na construção de cada uma dessas hipóteses (que, portanto, foram classificados pela regra *default*), o erro aparente cometido no conjunto S_8 e o erro cometido no conjunto de teste independente S_{teste}. A forma como essas hipóteses se comportam em relação à hipótese verdadeira pode ser vista na Figura 14.

Classificadores	Subconjunto/ # Ex. Treinamento	Erro Aparente	$e \pm dp$	# de Regras Induzidas
$\mathbf{h_1}$	$S_1/120$	0.80%	7.50% ± 2.31%	13
$\mathbf{h_2}$	$S_2/120$	0.00%	3.33% ± 2.55%	7
$\mathbf{h_3}$	$S_3/120$	0.80%	4.15% ± 1.38%	12
Total	–	–	–	32

Tabela 8: Resultados Obtidos — Hipóteses de Entrada do $\mathcal{RCO}$ (Fase 2 — Experimento 1)

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
$\mathbf{h}_{1_\texttt{accR}}$	0.00%	6.00%	12	03 – 02.50%
$\mathbf{h}_{1_\texttt{covR}}$	0.00%	6.00%	3	60 – 50.00%
$\mathbf{h}_{1_\texttt{satR}}$	0.00%	6.00%	12	03 – 02.50%
$\mathbf{h}_{1_\texttt{sensR}}$	0.00%	6.00%	3	60 – 50.00%
$\mathbf{h}_{1_\texttt{specR}}$	10.00%	7.50%	6	57 – 47.50%
$\mathbf{h}_{1_\texttt{supR}}$	0.00%	6.00%	3	60 – 50.00%

Tabela 9: Resultados Obtidos Utilizando $\mathcal{RCO}$ (Fase 2 — Experimento 1)

6.2 Descrição do Experimento 2

As duas fases do experimento 2 podem ser visualizadas na Figura 15. A seguir, são mostrados os resultados obtidos nas fases 1 e 2 desse experimento.

Fase 1 Neste caso foi utilizado por ambos os indutores o conjunto de exemplos constituído pela união dos subconjuntos de exemplos $S_1, ..., S_5$ e S_8. Os resultados obtidos e o número de regras induzidas para cada hipótese são mostrados na Tabela 10.

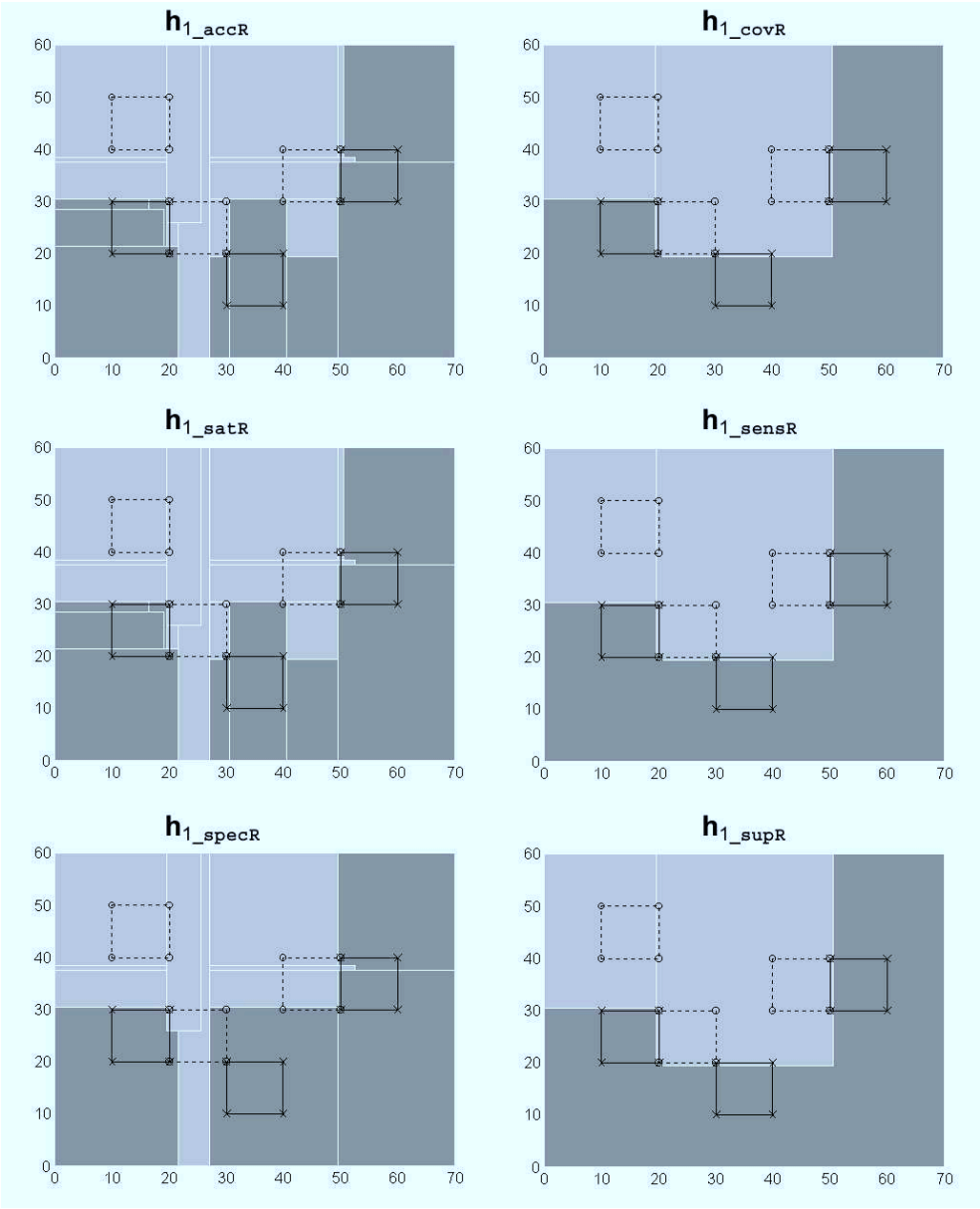

Figura 14: Representação Gráfica das Hipóteses Induzidas por $\mathcal{RCO}$ na Fase 2 — Experimento 1

Algoritmos	Subconjunto/ # Ex. Treinamento	Erro Aparente	$e \pm dp$	# de Regras Induzidas
$\mathcal{CN}2$	$S_1 \bigcup ... \bigcup S_5 \bigcup S_8/720$	2.00%	2.99% ± 0.60%	15
$\mathcal{S}$ee5	$S_1 \bigcup ... \bigcup S_5 \bigcup S_8/720$	2.00%	3.00% ± 1.20 %	7

Tabela 10: Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{S}$ee5 (Fase 1 — Experimento 2)

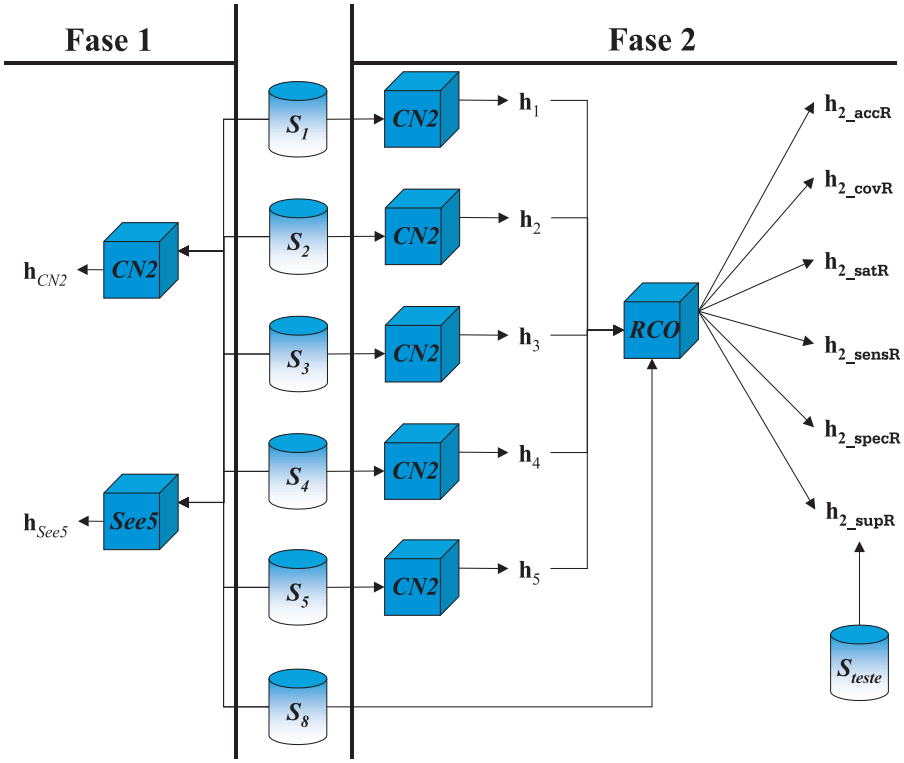

Figura 15: Ilustração do Experimento 2.

Fase 2 Na fase 2, foram construídas as hipóteses $\mathbf{h}_{2_\texttt{accR}}$, $\mathbf{h}_{2_\texttt{covR}}$, $\mathbf{h}_{2_\texttt{satR}}$, $\mathbf{h}_{2_\texttt{sensR}}$, $\mathbf{h}_{2_\texttt{specR}}$ e $\mathbf{h}_{2_\texttt{supR}}$. Para isso, foram utilizadas as hipóteses $\mathbf{h}_1, ..., \mathbf{h}_5$ e o conjunto de exemplos S_8 — Figura 15. As hipóteses $\mathbf{h}_1, ..., \mathbf{h}_5$ foram induzidas utilizando, respectivamente, os conjuntos de exemplos $S_1, ..., S_5$. O erro aparente de cada hipótese e o número de regras induzidas bem como a estimativa da taxa de erro e desvio padrão utilizando a técnica de 10-fold cross-validation são mostrados na Tabela 11.

Classificadores	Subconjunto/ # Ex. Treinamento	Erro Aparente	$e \pm dp$	# de Regras Induzidas
$\mathbf{h_1}$	$S_1/120$	0.80%	7.50% ± 2.31%	13
$\mathbf{h_2}$	$S_2/120$	0.00%	3.33% ± 2.55%	7
$\mathbf{h_3}$	$S_3/120$	0.80%	4.15% ± 1.38%	12
$\mathbf{h_4}$	$S_4/120$	0.00%	0.00% ± 0.00%	7
$\mathbf{h_5}$	$S_5/120$	0.00%	5.82% ± 3.29%	9
Total	–	–	–	48

Tabela 11: Resultados Obtidos — Hipóteses de Entrada do $\mathcal{RCO}$ (Fase 2 — Experimento 2)

Os resultados obtidos para as hipóteses $\mathbf{h}_{2_\texttt{accR}}$, $\mathbf{h}_{2_\texttt{covR}}$, $\mathbf{h}_{2_\texttt{satR}}$, $\mathbf{h}_{2_\texttt{sensR}}$, $\mathbf{h}_{2_\texttt{specR}}$ e $\mathbf{h}_{2_\texttt{supR}}$ são idênticos aos obtidos na fase 2 do experimento 1 — Tabela 9. Ou seja, foram construídas as mesmas hipóteses que no experimento anterior. A explicação para esse resultado é encontrada analisando dois fatores:

1. As regras que constituem cada uma das hipóteses utilizadas por $\mathcal{RCO}$;
2. A forma que o $\mathcal{RCO}$ escolhe a melhor regra, no caso de mais de uma regra possuir o mesmo melhor valor para a medida sendo considerada por $\mathcal{RCO}$ na construção da nova hipótese (desempate). Dada uma hipótese $\mathbf{h}$ e uma medida de avaliação de regra $Measure$, a "melhor" regra de $\mathbf{h}$ é definida pela regra com melhor medida $Measure$ em $\mathbf{h}$.

No caso do experimento 2, as regras que constituem as hipóteses $\mathbf{h}_4$ e $\mathbf{h}_5$ nunca foram consideradas melhores por $\mathcal{RCO}$ pois, em caso de empate, as regras dos classificadores anteriores são escolhidas. Em outras palavras, se $\mathcal{RCO}$ considera como entrada hipóteses $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_L$, nessa ordem, e em uma iteração existir duas ou mais regras, por exemplo, a regra R_u da hipótese $\mathbf{h}_i$ e a regra R_q da hipótese $\mathbf{h}_j$, $i < j$, ambas com a mesma melhor medida, segundo o critério sendo considerado por $\mathcal{RCO}$, então $\mathcal{RCO}$ escolhe a regra R_u, pois $i < j$, ou seja, a hipótese $\mathbf{h}_i$ precede a hipótese $\mathbf{h}_j$.

Dessa forma, $\mathcal{RCO}$ dá preferência, no caso de empate, às regras que fazem parte das primeiras hipóteses. Uma forma de diminuir esse *bias* é escolher aleatoriamente a melhor regra a ser utilizada na hipótese construída pelo $\mathcal{RCO}$, como visto a seguir.

6.3 Descrição do Experimento 3

As duas fases do experimento 3 podem ser visualizadas na Figura 16. Como descrito na Seção 6.2, as regras que constituem as hipóteses $\mathbf{h}_4$ e $\mathbf{h}_5$ induzidas por $\mathcal{CN}2$ não têm influência nas hipóteses construídas por $\mathcal{RCO}$. Entretanto, essas duas hipóteses foram mantidas pois os exemplos utilizados para induzi-las são também utilizados por $\mathcal{CN}2$ e $\mathcal{S}$ee5 na fase 1 do experimento. Em outras palavras, ainda que elas não influem na hipótese construída pelo $\mathcal{RCO}$, os exemplos utilizados influem no cálculo do erro e desvio padrão dos classificadores induzidos na fase 1. A seguir, são mostrados os resultados obtidos nas fases 1 e 2 desse experimento.

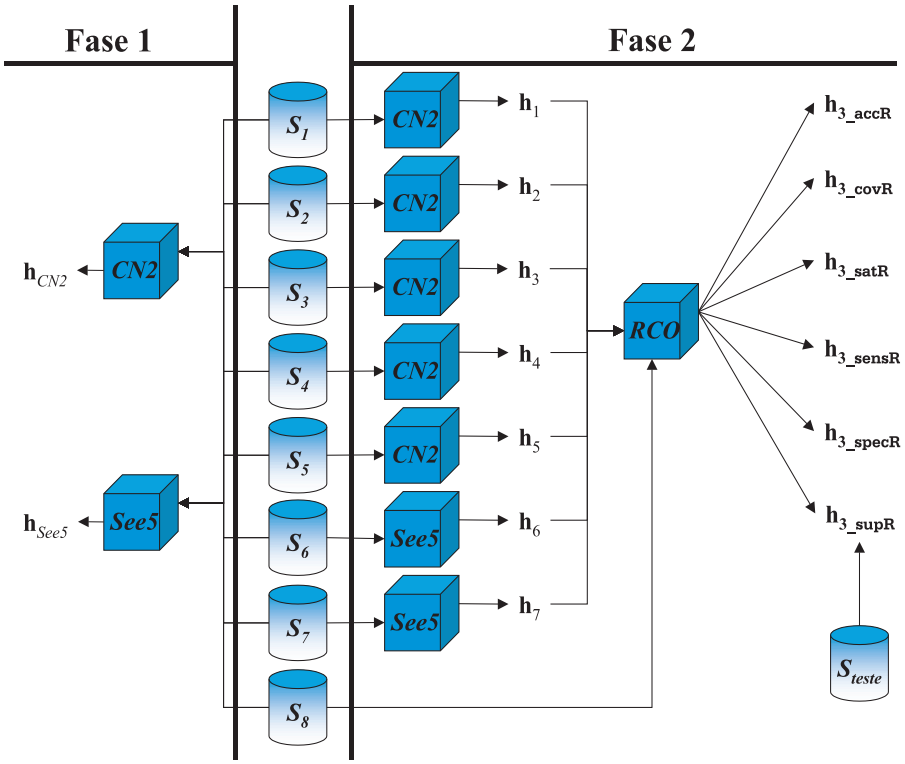

Figura 16: Ilustração do Experimento 3.

Fase 1 Neste caso, o conjunto de exemplos construído pela união dos subconjuntos de exemplos $S_1, ..., S_8$ foi utilizada por ambos os indutores $\mathcal{CN}2$ e $\mathcal{S}$ee5. Os resultados obtidos e o número de regras induzidas para cada hipótese são mostrados na Tabela 12.

Fase 2 Na fase 2, foram construídas as hipóteses $\mathbf{h}_{3_\mathtt{accR}}$, $\mathbf{h}_{3_\mathtt{covR}}$, $\mathbf{h}_{3_\mathtt{satR}}$, $\mathbf{h}_{3_\mathtt{sensR}}$, $\mathbf{h}_{3_\mathtt{specR}}$ e $\mathbf{h}_{3_\mathtt{supR}}$. Para isso, foram utilizadas as hipóteses $\mathbf{h}_1, ..., \mathbf{h}_7$ e o conjunto de exemplos S_8 — Figura 16. As hipóteses $\mathbf{h}_1, ..., \mathbf{h}_7$ foram induzidas utilizando, respectivamente,

Algoritmos	Subconjunto/ # Ex. Treinamento	Erro Aparente	$e \pm dp$	# de Regras Induzidas
$\mathcal{CN}2$	$S_1 \bigcup ... \bigcup S_8/960$	1.40%	2.76% ± 0.44%	15
$\mathcal{S}$ee5	$S_1 \bigcup ... \bigcup S_8/960$	2.10%	3.00% ± 0.40%	8

Tabela 12: Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{S}$ee5 (Fase 1 — Experimento 3)

os conjuntos de exemplos $S_1, ..., S_7$. O erro aparente de cada hipótese e o número de regras induzidas bem como a estimativa da taxa de erro e desvio padrão utilizando a técnica de 10-fold cross-validation são mostrados na Tabela 13.

Para as hipóteses $\mathbf{h}_{3_\mathtt{accR}}$, $\mathbf{h}_{3_\mathtt{covR}}$, $\mathbf{h}_{3_\mathtt{satR}}$, $\mathbf{h}_{3_\mathtt{sensR}}$, $\mathbf{h}_{3_\mathtt{specR}}$ e $\mathbf{h}_{3_\mathtt{supR}}$ construídas, a Tabela 14 mostra, de forma semelhante aos experimentos 1 e 2, o número de regras selecionadas (inclusive a regra *default*), o número de exemplos que restaram na construção de cada uma dessas hipóteses (que, portanto, foram classificados pela regra *default*), o erro aparente cometido no conjunto S_8 e o erro cometido no conjunto de teste independente S_{teste}. Observando os conjuntos de regras construídos pelo $\mathcal{RCO}$ nesse experimento, apenas a hipótese $\mathbf{h}_{3_\mathtt{covR}}$ apresentou uma pequena diferença em relação a $\mathbf{h}_{1_\mathtt{covR}}$ e $\mathbf{h}_{2_\mathtt{covR}}$. Todas as outras hipóteses são idênticas às hipóteses construídas nos experimentos anteriores.

Classificadores	Subconjunto/ # Ex. Treinamento	Erro Aparente	$e \pm dp$	# de Regras Induzidas
$\mathbf{h_1}$	$S_1/120$	0.80%	7.50% ± 2.31%	13
$\mathbf{h_2}$	$S_2/120$	0.00%	3.33% ± 2.55%	7
$\mathbf{h_3}$	$S_3/120$	0.80%	4.15% ± 1.38%	12
$\mathbf{h_4}$	$S_4/120$	0.00%	0.00% ± 0.00%	7
$\mathbf{h_5}$	$S_5/120$	0.00%	5.82% ± 3.29%	9
$\mathbf{h_6}$	$S_6/120$	0.80%	4.20% ± 2.60%	6
$\mathbf{h_7}$	$S_7/120$	2.50%	5.80% ± 2.50%	6
Total	–	–	–	60

Tabela 13: Resultados Obtidos — Hipóteses de Entrada do $\mathcal{RCO}$ (Fase 2 — Experimento 3)

Esses resultados não são os esperados, isto é, obter as mesmas hipóteses nos diversos experimentos. Assim, foi decidido investigar cuidadosamente o motivo das hipóteses geradas por $\mathcal{RCO}$ não serem distintas quando o conjunto de hipóteses $\mathcal{H}$ é incrementado com novas hipóteses. A explicação é simples e está intimamente ligada ao critério de desempate, por nós implementado, para escolher a melhor regra. O critério de desempate utilizado é descrito pelos passos 1 e 2 a seguir:

1. Sejam as hipóteses $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_L$, nessa ordem, os dados de entrada do $\mathcal{RCO}$. Então, se em uma iteração do $\mathcal{RCO}$ existirem duas regras R_u e R_q, com $R_u \in \mathbf{h}_i$ e $R_q \in \mathbf{h}_j$,

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
$\mathbf{h}_{3_\texttt{accR}}$	0.00%	6.00%	12	03 – 02.50%
$\mathbf{h}_{3_\texttt{covR}}$	0.00%	9.00%	3	60 – 50.00%
$\mathbf{h}_{3_\texttt{satR}}$	0.00%	6.00%	12	03 – 02.50%
$\mathbf{h}_{3_\texttt{sensR}}$	0.00%	6.00%	3	60 – 50.00%
$\mathbf{h}_{3_\texttt{specR}}$	10.00%	7.50%	6	57 – 47.50%
$\mathbf{h}_{3_\texttt{supR}}$	0.00%	6.00%	3	60 – 50.00%

Tabela 14: Resultados Obtidos Utilizando $\mathcal{RCO}$ (Fase 2 — Experimento 3)

$i < j$, ambas com a mesma melhor medida segundo o critério implementado, o $\mathcal{RCO}$ escolhe a regra R_u, pois $i < j$, isto é a hipótese $\mathbf{h}_i$ precede a hipótese $\mathbf{h}_j$;

2. Supondo que existam, em uma iteração do $\mathcal{RCO}$, três regras R_u, R_q e R_z nessa ordem, todas com a mesma melhor medida segundo o critério implementado, então, sempre será escolhida a regra R_u e, no caso desta regra não cobrir nenhum exemplo, a iteração do $\mathcal{RCO}$ pára.

Como pode ser observado, esse critério de desempate favorece a escolha das regras das primeiras hipóteses. Em outras palavras, $\mathcal{RCO}$ não "enxerga" as regras das hipóteses posteriores. Isso explica o motivo de $\mathcal{RCO}$ gerar a mesma hipótese quando é utilizado o conjunto de hipóteses $\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\}$ ou o conjunto $\{\mathbf{h}_1, ..., \mathbf{h}_5\}$.

Assim, foi decidido revisar o critério utilizado para escolher a melhor regra em caso de empate. O novo critério é tal que, em cada iteração, seleciona-se, de forma aleatória, uma regra do subconjunto de regras que contém as regras com a mesma melhor medida. Se a regra selecionada não cobre nenhum exemplo, seleciona-se aleatoriamente outra regra desse subconjunto. O processo continua até que alguma regra selecionada cubra ao menos um exemplo do conjunto de exemplos disponível na iteração, ou até que esse subconjunto de regras esteja vazio.

Com o $\mathcal{RCO}$ revisado, o experimento 3 foi repetido 3(três) vezes, com as mesmas hipóteses de entrada $\mathbf{h}_1, ..., \mathbf{h}_7$ e o mesmo conjunto de exemplos S_8, com o intuito de verificar se as hipóteses construídas são distintas e melhores que as anteriores. Os resultados são descritos a seguir.

6.4 Descrição dos Experimentos 4, 5 e 6

Analogamente à fase 2 do experimento 3, para as hipóteses $\mathbf{h}_{4_\texttt{accR}}$, $\mathbf{h}_{4_\texttt{covR}}$, $\mathbf{h}_{4_\texttt{satR}}$, $\mathbf{h}_{4_\texttt{sensR}}$, $\mathbf{h}_{4_\texttt{specR}}$ e $\mathbf{h}_{4_\texttt{supR}}$ construídas, a Tabela 15 mostra o número de regras selecionadas (inclusive a regra *default*), o número de exemplos que restaram na construção de cada uma dessas hipóteses (que, portanto, foram classificados pela regra *default*), o erro aparente cometido

no conjunto S_8 e o erro cometido no conjunto de teste independente S_{teste}. Já esses dados medidos para as hipóteses $\mathbf{h}_{5_\mathtt{accR}}$, $\mathbf{h}_{5_\mathtt{covR}}$, $\mathbf{h}_{5_\mathtt{satR}}$, $\mathbf{h}_{5_\mathtt{sensR}}$, $\mathbf{h}_{5_\mathtt{specR}}$ e $\mathbf{h}_{5_\mathtt{supR}}$ são mostrados na Tabela 16, e para as hipóteses $\mathbf{h}_{6_\mathtt{accR}}$, $\mathbf{h}_{6_\mathtt{covR}}$, $\mathbf{h}_{6_\mathtt{satR}}$, $\mathbf{h}_{6_\mathtt{sensR}}$, $\mathbf{h}_{6_\mathtt{specR}}$ e $\mathbf{h}_{6_\mathtt{supR}}$, são mostrados na Tabela 17.

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
$\mathbf{h}_{4_\mathtt{accR}}$	0.00%	6.00%	6	20 – 16.67%
$\mathbf{h}_{4_\mathtt{covR}}$	3.33%	9.00%	3	60 – 50.00%
$\mathbf{h}_{4_\mathtt{satR}}$	0.83%	5.50%	7	05 – 4.17%
$\mathbf{h}_{4_\mathtt{sensR}}$	3.33%	9.00%	3	60 – 50.00%
$\mathbf{h}_{4_\mathtt{specR}}$	0.83%	6.00%	7	23 – 19.17%
$\mathbf{h}_{4_\mathtt{supR}}$	0.00%	6.00%	3	60 – 50.00%

Tabela 15: Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 4

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
$\mathbf{h}_{5_\mathtt{accR}}$	0.00%	4.50%	8	20 – 16.67%
$\mathbf{h}_{5_\mathtt{covR}}$	3.33%	9.00%	3	60 – 50.00%
$\mathbf{h}_{5_\mathtt{satR}}$	0.00%	6.00%	11	04 – 33.33%
$\mathbf{h}_{5_\mathtt{sensR}}$	3.33%	9.00%	3	60 – 50.00%
$\mathbf{h}_{5_\mathtt{specR}}$	0.83%	3.00%	9	25 – 20.83%
$\mathbf{h}_{5_\mathtt{supR}}$	3.33%	9.00%	3	60 – 50.00%

Tabela 16: Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 5

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
$\mathbf{h}_{6_\mathtt{accR}}$	0.00%	6.00%	7	24 – 20.00%
$\mathbf{h}_{6_\mathtt{covR}}$	3.33%	9.00%	3	60 – 50.00%
$\mathbf{h}_{6_\mathtt{satR}}$	0.00%	4.50%	8	20 – 16.67%
$\mathbf{h}_{6_\mathtt{sensR}}$	3.33%	9.00%	3	60 – 50.00%
$\mathbf{h}_{6_\mathtt{specR}}$	0.00%	6.00%	7	20 – 16.67%
$\mathbf{h}_{6_\mathtt{supR}}$	3.33%	9.00%	3	60 – 50.00%

Tabela 17: Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 6

A forma como as hipóteses dos experimentos 4, 5 e 6 se comportam em relação à hipótese verdadeira pode ser vista, respectivamente, nas Figuras 17, 18 e 19.

Deve ser observado, nesses 3 experimentos (experimentos 4, 5 e 6), que:

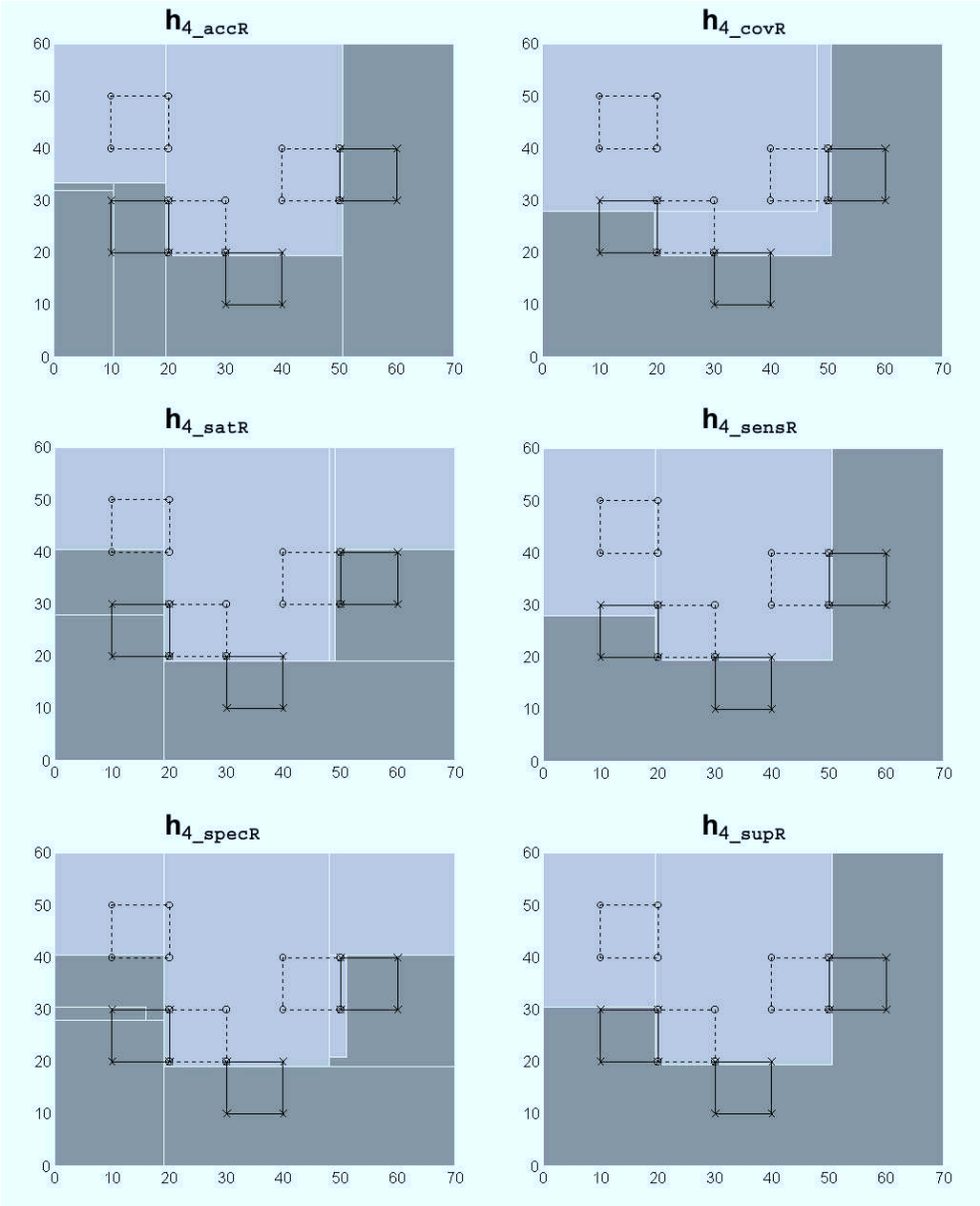

Figura 17: Representação Gráfica das Hipóteses Induzidas por $\mathcal{RCO}$ — Experimento 4

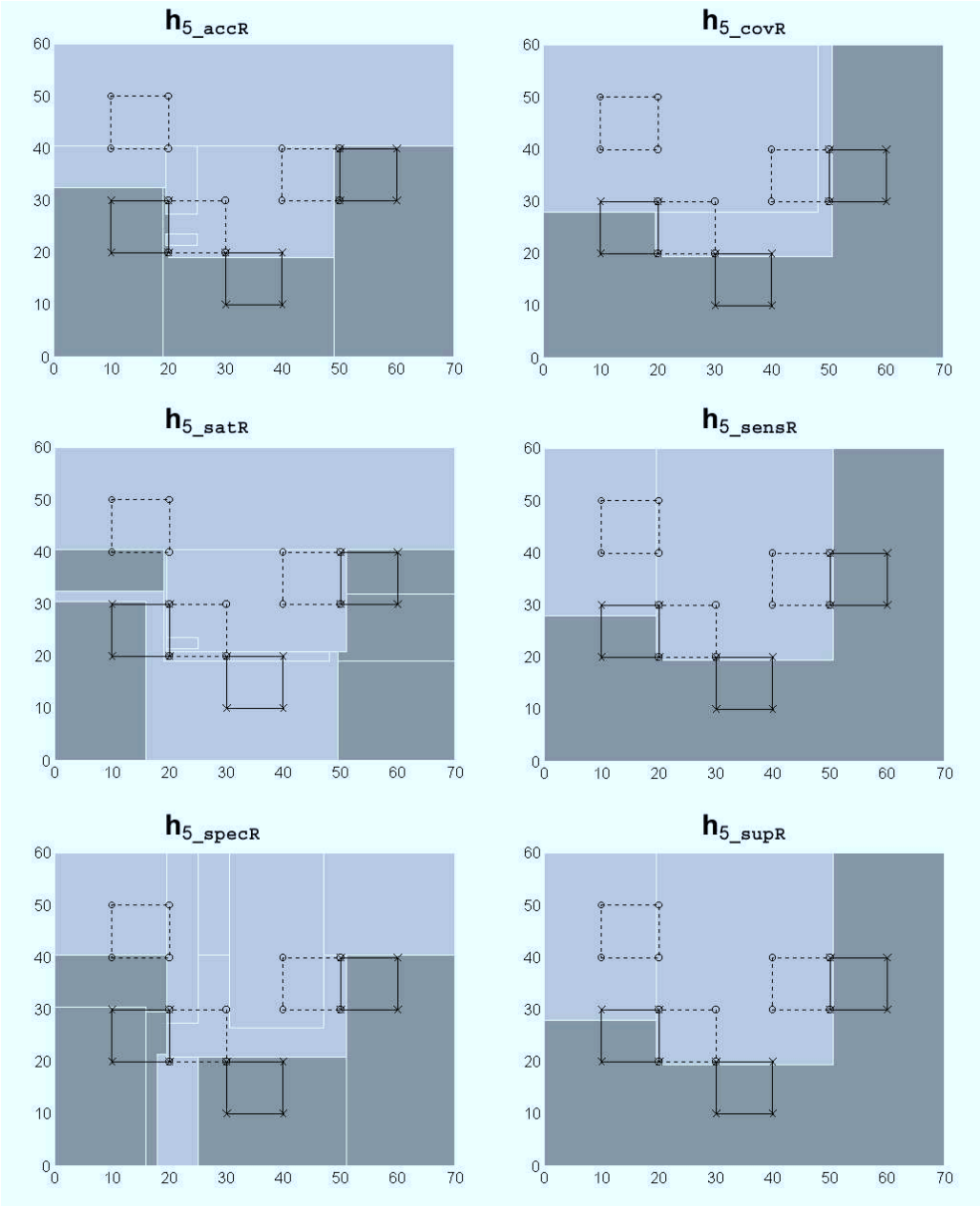

Figura 18: Representação Gráfica das Hipóteses Induzidas por $\mathcal{RCO}$ — Experimento 5

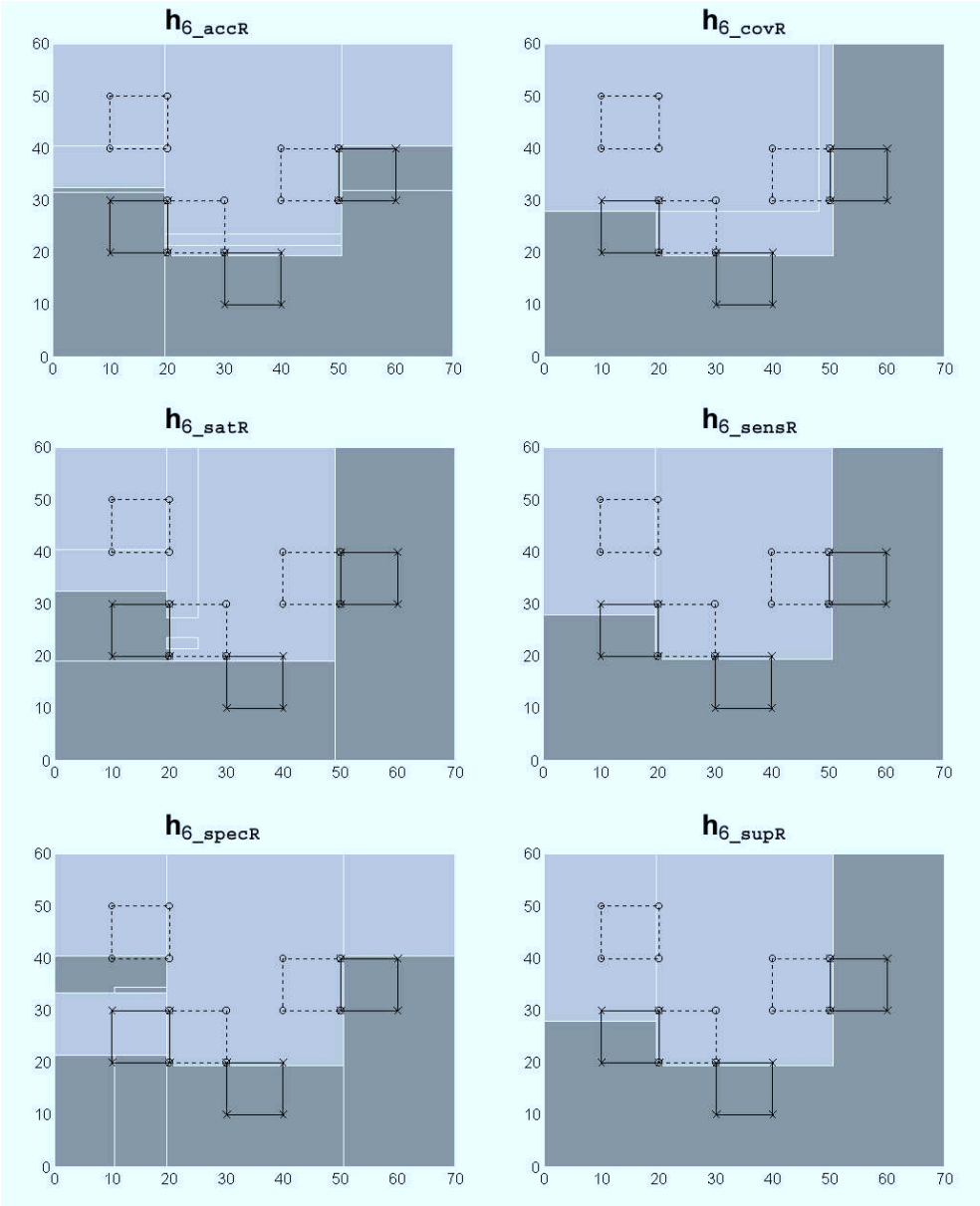

Figura 19: Representação Gráfica das Hipóteses Induzidas por $\mathcal{RCO}$ — Experimento 6

- Com os critérios de avaliação de regra `covR` e `sensR`, foram induzidas hipóteses idênticas nos 3 experimentos. A diferença está na ordem em que as regras foram selecionadas, conforme pode ser visto no Apêndice B.
- Para o critério `supR`, foram selecionadas regras diferentes no experimento 4, somente.
- Para os outros critérios — `accR`, `satR` e `specR` —, as regras selecionadas foram diferentes nos 3 experimentos, como se pode observar nas Tabelas 15, 16 e 17 — alguns dados apresentados podem ser iguais, mas as hipóteses construídas por $\mathcal{RCO}$ não são idênticas.

Esses resultados mostram que, com o novo critério de desempate implementado, o $\mathcal{RCO}$ consegue "enxergar" e utilizar as melhores regras de todas as hipóteses, independentemente da ordem dessas hipóteses.

6.5 Análise dos Resultados

Para facilitar a visualização dos resultados, a Tabela 18 mostra os resultados obtidos na fase 1 dos experimentos 1, 2 e 3. Como esperado, o erro e o desvio padrão tendem a se estabilizar quanto maior é o número de exemplos.

Experimento	Algoritmos	Erro Aparente	$e \pm dp$	# de Regras Induzidas
1	$\mathcal{CN}2$	1.40%	4.46% ± 0.85%	15
	$\mathcal{S}$ee5	2.20%	3.30% ± 0.90%	7
2	$\mathcal{CN}2$	2.00%	2.99% ± 0.60%	15
	$\mathcal{S}$ee5	2.00%	3.00% ± 1.20 %	7
3	$\mathcal{CN}2$	1.40%	2.76% ± 0.44%	15
	$\mathcal{S}$ee5	2.10%	3.00% ± 0.40%	8

Tabela 18: Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{S}$ee5 na Fase 1 dos Experimentos 1, 2 e 3

Com a revisão do $\mathcal{RCO}$, verificou-se que, para as medidas `covR`, `sensR` e `supR`, não existem regras no conjunto de hipóteses de entrada com alto valor nessas medidas que cobrem exemplos da classe x. Isso se observa quando se olha para os exemplos que restaram na construção das hipóteses, cobertos pela regra *default*: todos os exemplos são da classe x. Deve ser lembrado que a regra *default* classifica os exemplos como pertencentes à classe x, o que explica o erro relativamente baixo em S_{teste} — Tabelas 15, 16 e 17 —, ainda no caso de muitos exemplos não terem sido cobertos durante a construção da nova hipótese. No futuro, esta característica do $\mathcal{RCO}$ será melhor investigada utilizando outros conjuntos de dados artificiais com mais de duas classes.

Se forem comparadas as taxas de erro em S_{teste} obtidas nos experimentos 4, 5 e 6 com as taxas de erro e desvio padrão obtidas por $\mathcal{CN}2$ e $\mathcal{S}$ee5 na fase 1 do experimento 3, é possível observar que, como esperado, as hipóteses induzidas por $\mathcal{CN}2$ e $\mathcal{S}$ee5 são melhores que as geradas por $\mathcal{RCO}$. Entretanto, considerando que o número de regras selecionadas (incluindo a regra *default*) por $\mathcal{RCO}$ é menor que o número total de regras disponíveis em cada experimento (60 regras) — Tabela 19 —, e observando que as hipóteses construídas por $\mathcal{RCO}$ apresentam um erro menor que o erro da classe majoritária, pode ser considerado que os resultados são positivos. É importante notar que as hipóteses de entrada $\mathbf{h}_1, ..., \mathbf{h}_7$ são as mesmas nos experimentos 4, 5 e 6. Outro fato importante é que as hipóteses induzidas pelo $\mathcal{CN}2$ são distintas umas das outras, como se pode observar na Figura 11 e no Apêndice A, com algumas semelhanças entre as hipóteses $\mathbf{h}_1$ e $\mathbf{h}_3$ e as hipóteses $\mathbf{h}_2$ e $\mathbf{h}_5$. Quanto às hipóteses $\mathbf{h}_6$ e $\mathbf{h}_7$ induzidas pelo $\mathcal{S}$ee5, nota-se uma grande semelhança entre elas, como se pode observar na Figura 12 e no Apêndice A. Esse fato indica que as duas hipóteses juntas têm praticamente o mesmo efeito de ter somente uma delas como hipótese de entrada do $\mathcal{RCO}$. Ainda assim, o número de regras distintas continua sendo grande em relação ao número de regras selecionadas.

Critério	Experimento 4	Experimento 5	Experimento 6
`accR`	6	8	7
`covR`	3	3	3
`satR`	7	11	8
`sensR`	3	3	3
`specR`	7	9	7
`supR`	3	3	3

Tabela 19: Número de Regras Selecionadas (com a Regra *Default*) nos Experimentos 4, 5 e 6

Quanto à representação gráfica das hipóteses, observa-se que as hipóteses construídas se aproximam bastante da hipótese verdadeira. As regiões da função verdadeira cobertas erroneamente pelas hipóteses explicam as taxas de erro obtidas com S_{teste}. É importante notar que, para hipóteses avaliadas como *unordered* com muitas regras, a forma de visualização proposta muitas vezes não é válida por não considerar empates de classificação, pois as regiões sobrepostas não são visualizáveis.

7 Conclusão

A prototipagem rápida oferecida pela linguagem Prolog agilizou o processo de implementação do sistema. O maior problema encontrado foi para documentar o sistema, já que não foi encontrada uma metodologia que atendesse as necessidades de documentação das especificidades do sistema. Para isso, foi criada uma metodologia de representação na forma de um Diagrama de Fluxo de Execução de Procedimentos (PEFD), a qual atendeu as

necessidades de documentação do sistema implementado.

Quanto ao Módulo de Combinação e Explicação, detalhado neste trabalho, o exemplo utilizado para mostrar o funcionamento do algoritmo $\mathcal{RCO}$ mostrou que as idéias propostas para combinação de classificadores funcionam. Deve ser lembrado que o objetivo principal de todo o trabalho realizado é propor uma forma de combinação de classificadores para se trabalhar com grandes bases de dados. Dada uma grande base de dados, divide-se esta base aleatoriamente em pequenas bases de tal forma que é viável fornecer essas bases de tamanho menor a um ou vários algoritmos de AM. Assim, com o trabalho realizado, verificou-se que se todas essas bases são mostradas ao(s) algoritmo(s) e para cada uma delas uma hipótese é induzida, uma seleção das melhores regras segundo algum critério de avaliação de regra pode ser realizada e, a partir da nova hipótese construída, pode-se utilizá-la para extrair conhecimento ou para classificar novos exemplos. Através dos experimentos 4, 5 e 6 realizados com o conjunto de dados artificial, verificou-se a viabilidade da técnica proposta, e a importância do critério de desempate utilizado pelo $\mathcal{RCO}$. Futuramente, outros critérios serão investigados e implementados no $\mathcal{RCO}$, permitindo ao usuário realizar a escolha do critério mais apropriado para o domínio em questão.

Um outro aspecto a ser considerado é que no início da implementação do MCE, pensava-se que o algoritmo de 10-fold cross validation utilizado para estimar o erro e o desvio padrão das hipóteses construídas pelo $\mathcal{RCO}$ (Seção 5) fosse válido. Depois, percebeu-se que o algoritmo deveria também induzir novamente as hipóteses de entrada do $\mathcal{RCO}$ a cada iteração do algoritmo de 10-fold cross-validation. Futuramente, essa implementação completa será incluída no MCE.

Finalizando, pode-se concluir que a idéia utilizada pelo $\mathcal{RCO}$ para construir a hipótese como um conjunto das "melhores" regras que constituem outras hipóteses é válida. Entretanto, as diversas medidas consideradas apresentam, do ponto de vista de exemplos não cobertos, comportamentos bem diferentes, o que pode influenciar no erro em S_{teste}, no caso do número de classes ser maior que dois. Este tema será melhor investigado no futuro.

Com relação à implementação da explicação de *ensembles*, o que o MCE faz para explicar a classificação feita por um conjunto de classificadores a um exemplo $\mathbf{x}$ é selecionar, do conjunto de hipóteses $\mathcal{H}$, um conjunto de regras $\mathbf{R}$ com as regras que cobrem o exemplo $\mathbf{x}$. Depois, selecionar de $\mathbf{R}$ a regra R com a melhor medida segundo algum critério implementado e escolhido pelo usuário, e retornar essa regra como sendo a explicação da classificação. Futuramente, deve-se analisar como as outras regras do conjunto $\mathbf{R}$ cobrem o exemplo $\mathbf{x}$ e deve-se tentar criar uma única regra R' a partir do conjunto de regras $\mathbf{R}$. Se R' for gerada, essa será a regra fornecida ao usuário para explicar a classificação de $\mathbf{x}$.

A Classificadores Utilizados pelo $\mathcal{RCO}$

Os 7(sete) classificadores simbólicos $\mathbf{h}_1, ..., \mathbf{h}_7$ utilizados no experimento, escritos em linguagem Prolog, encontram-se a seguir. Os classificadores $\mathbf{h}_1, ..., \mathbf{h}_5$ foram induzidos pelo algoritmo $\mathcal{CN}2$ e os classificadores $\mathbf{h}_6$ e $\mathbf{h}_7$ foram induzidos pelo algoritmo $\mathcal{S}$ee5. Deve ser observado que a matriz de contingência (freqüências) das regras que constituem cada um desses classificadores foi calculado utilizando todo o conjunto de dados disponível para os experimentos com exceção do conjunto S_{teste}, ou seja, 960 exemplos.

A.1 Classificadores Induzidos pelo $\mathcal{CN}2$

A.1.1 $\mathbf{h}_1$

```
inducer(id_198,cn2).
inputFile(id_198,-).
dAte(id_198,day('Tue','Jan',8,2002),time(16 : 37 : 33)).
evaluatedAs(id_198,'UNORDERED').
dataFile(id_198,'ds_art1.data').
nameFile(id_198,'ds_art.names').

rule(id_198,1,
  [greater(att_1,30.5),less(att_1,40.5)],
  class(x),
  [0.153,0.006,0.494,0.347,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,2,
  [greater(att_1,50.5)],
  class(x),
  [0.159,0.000,0.500,0.341,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,3,
  [less(att_1,16.5),less(att_2,35)],
  class(x),
  [0.096,0.000,0.500,0.404,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,4,
  [less(att_1,19),less(att_2,28.5)],
  class(x),
  [0.121,0.000,0.500,0.379,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,5,
  [less(att_1,21.5),less(att_2,21.5)],
  class(x),
  [0.031,0.003,0.497,0.469,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,6,
```

```
  [greater(att_1,49.5),greater(att_2,39)],
  class(x),
  [0.008,0.000,0.500,0.492,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,7,
  [greater(att_1,49.5),less(att_2,37.5)],
  class(x),
  [0.127,0.007,0.493,0.373,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,8,
  [less(att_1,49.5),greater(att_2,30.5)],
  class(o),
  [0.314,0.000,0.500,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,9,
  [greater(att_1,21),greater(att_2,20.5),less(att_2,30.5)],
  class(o),
  [0.144,0.006,0.494,0.356,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,10,
  [less(att_1,52.5),greater(att_2,37.5),less(att_2,38.5)],
  class(o),
  [0.017,0.001,0.499,0.483,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,11,
  [greater(att_1,21.5),less(att_1,27)],
  class(o),
  [0.086,0.000,0.500,0.414,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,12,
  [greater(att_1,19.5),less(att_1,25.5),greater(att_2,26)],
  class(o),
  [0.038,0.005,0.495,0.463,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_198,13,
  [],
  class(x),
  [],[]).
```

A.1.2 h_2

```
inducer(id_168,cn2).
inputFile(id_168,-).
dAte(id_168,day('Tue','Jan',8,2002),time(16 : 38 : 2)).
evaluatedAs(id_168,'UNORDERED').
dataFile(id_168,'ds_art2.data').
nameFile(id_168,'ds_art.names').
```

```
rule(id_168,1,
  [greater(att_1,29.5),less(att_2,30.5)],
  class(x),
  [0.173,0.019,0.481,0.327,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_168,2,
  [greater(att_1,50.5)],
  class(x),
  [0.159,0.000,0.500,0.341,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_168,3,
  [greater(att_1,10.5),less(att_1,19.5),less(att_2,34.5)],
  class(x),
  [0.144,0.000,0.500,0.356,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_168,4,
  [less(att_1,10.5),less(att_2,32)],
  class(x),
  [0.011,0.000,0.500,0.489,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_168,5,
  [less(att_1,50.5),greater(att_2,33.5)],
  class(o),
  [0.264,0.003,0.497,0.236,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_168,6,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_168,7,
  [],
  class(x),
  [],[]).
```

A.1.3 $\mathbf{h}_3$

```
inducer(id_178,cn2).
inputFile(id_178,-).
dAte(id_178,day('Tue','Jan',8,2002),time(16 : 38 : 12)).
evaluatedAs(id_178,'UNORDERED').
dataFile(id_178,'ds_art3.data').
nameFile(id_178,'ds_art.names').

rule(id_178,1,
  [greater(att_1,29.5),less(att_1,40.5)],
  class(x),
```

```
  [0.167,0.014,0.486,0.333,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,2,
  [greater(att_1,50.5)],
  class(x),
  [0.159,0.000,0.500,0.341,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,3,
  [less(att_1,19.5),less(att_2,35.5)],
  class(x),
  [0.155,0.000,0.500,0.345,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,4,
  [greater(att_1,49.5),less(att_2,34.5)],
  class(x),
  [0.075,0.004,0.496,0.425,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,5,
  [less(att_1,20.5),greater(att_2,22.5),less(att_2,24.5)],
  class(x),
  [0.040,0.003,0.497,0.460,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,6,
  [less(att_1,20.5),greater(att_2,24),less(att_2,27.5)],
  class(x),
  [0.042,0.004,0.496,0.458,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,7,
  [less(att_1,50.5),greater(att_2,32.5)],
  class(o),
  [0.288,0.004,0.496,0.213,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,8,
  [greater(att_1,20.5),less(att_1,29.5)],
  class(o),
  [0.146,0.000,0.500,0.354,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,9,
  [greater(att_1,18),less(att_1,25),less(att_2,21.5)],
  class(o),
  [0.013,0.004,0.496,0.488,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,10,
  [greater(att_1,30.5),less(att_1,47),greater(att_2,26.5)],
  class(o),
  [0.109,0.000,0.500,0.391,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,11,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,27.5)],
  class(o),
  [0.028,0.002,0.498,0.472,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_178,12,
  [],
```

```
  class(x),
  [],[]).
```

A.1.4 $\mathbf{h}_4$

```
inducer(id_184,cn2).
inputFile(id_184,-).
dAte(id_184,day('Tue','Jan',8,2002),time(16 : 39 : 17)).
evaluatedAs(id_184,'UNORDERED').
dataFile(id_184,'ds_art4.data').
nameFile(id_184,'ds_art.names').

rule(id_184,1,
  [less(att_2,21.5)],
  class(x),
  [0.198,0.028,0.472,0.302,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_184,2,
  [greater(att_1,51)],
  class(x),
  [0.140,0.000,0.500,0.360,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_184,3,
  [less(att_1,19.5),less(att_2,29.5)],
  class(x),
  [0.144,0.000,0.500,0.356,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_184,4,
  [less(att_1,16),less(att_2,30.5)],
  class(x),
  [0.079,0.000,0.500,0.421,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_184,5,
  [greater(att_1,19.5),less(att_1,51),greater(att_2,21)],
  class(o),
  [0.315,0.017,0.483,0.185,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_184,6,
  [greater(att_2,40.5)],
  class(o),
  [0.157,0.000,0.500,0.343,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_184,7,
  [],
  class(x),
  [],[]).
```

A.1.5 h_5

```
inducer(id_195,cn2).
inputFile(id_195,-).
dAte(id_195,day('Tue','Jan',8,2002),time(16 : 39 : 28)).
evaluatedAs(id_195,'UNORDERED').
dataFile(id_195,'ds_art5.data').
nameFile(id_195,'ds_art.names').

rule(id_195,1,
  [less(att_2,20.5)],
  class(x),
  [0.180,0.008,0.492,0.320,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_195,2,
  [greater(att_1,50.5)],
  class(x),
  [0.159,0.000,0.500,0.341,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_195,3,
  [less(att_1,20.5),greater(att_2,23.5),less(att_2,35)],
  class(x),
  [0.103,0.008,0.492,0.397,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_195,4,
  [less(att_1,19.5),less(att_2,31.5)],
  class(x),
  [0.155,0.000,0.500,0.345,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_195,5,
  [greater(att_1,49.5),greater(att_2,32)],
  class(x),
  [0.125,0.006,0.494,0.375,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_195,6,
  [less(att_1,49.5),greater(att_2,32.5)],
  class(o),
  [0.281,0.000,0.500,0.219,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_195,7,
  [greater(att_1,20.5),less(att_1,55),greater(att_2,20.5),less(att_2,31.5)],
  class(o),
  [0.178,0.014,0.486,0.322,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_195,8,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,21.5),less(att_2,23.5)],
  class(o),
  [0.018,0.001,0.499,0.482,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_195,9,
```

```
  [],
  class(x),
  [],[]).
```

A.2 Classificadores Induzidos pelo $\mathcal{S}$ee5

A.2.1 $\mathbf{h}_6$

```
inducer(id_148,c4.5rules).
inputFile(id_148,-).
dAte(id_148,day(-,'Jan',15,2002),time(13 : 2 : 14)).
evaluatedAs(id_148,'UNORDERED').
dataFile(id_148,'ds_art6.data').
nameFile(id_148,'ds_art.names').

rule(id_148,1,
  [lessORequal(att_2,19)],
  class(x),
  [0.160,0.000,0.500,0.340,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_148,2,
  [greater(att_1,48)],
  class(x),
  [0.167,0.029,0.471,0.333,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_148,3,
  [lessORequal(att_1,19),lessORequal(att_2,28)],
  class(x),
  [0.135,0.000,0.500,0.365,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_148,4,
  [lessORequal(att_1,48),greater(att_2,28)],
  class(o),
  [0.328,0.022,0.478,0.172,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_148,5,
  [greater(att_1,19),lessORequal(att_1,48),greater(att_2,19)],
  class(o),
  [0.314,0.018,0.482,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_148,6,
  [],
  class(x),
  [],[]).
```

A.2.2 h_7

```
inducer(id_135,c4.5rules).
inputFile(id_135,-).
dAte(id_135,day(-,'Jan',15,2002),time(13 : 3 : 0)).
evaluatedAs(id_135,'UNORDERED').
dataFile(id_135,'ds_art7.data').
nameFile(id_135,'ds_art.names').

rule(id_135,1,
  [lessORequal(att_2,19)],
  class(x),
  [0.160,0.000,0.500,0.340,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_135,2,
  [lessORequal(att_1,20),lessORequal(att_2,37)],
  class(x),
  [0.167,0.014,0.486,0.333,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_135,3,
  [greater(att_1,49)],
  class(x),
  [0.167,0.008,0.492,0.333,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_135,4,
  [greater(att_1,20),lessORequal(att_1,49),greater(att_2,19)],
  class(o),
  [0.311,0.006,0.494,0.189,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_135,5,
  [greater(att_2,37)],
  class(o),
  [0.200,0.040,0.460,0.300,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_135,6,
  [],
  class(x),
  [],[]).
```

B Classificadores Construídos pelo $\mathcal{RCO}$ nos Experimentos Realizados

Os 6(seis) classificadores simbólicos construídos pelo $\mathcal{RCO}$ resultantes de cada experimento estão descritos na Sintaxe Padrão Prolog do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Deve ser observado que a matriz de contingência (freqüências) das regras que constituem cada um desses classificadores foi

calculado utilizando todo o conjunto de dados disponível para os experimentos com exceção do conjunto S_{teste}, ou seja, 960 exemplos.

B.1 Experimento 1

B.1.1 $\mathbf{h}_{1_accR}$

```
%Exemplos não cobertos: [24,37,79]

inducer(id_accR211,rco).
inputFile(id_accR211,-).
dAte(id_accR211,day(-,'Jan',10,2002),time(11 : 1 : 29)).
evaluatedAs(id_accR211,'UNORDERED').
dataFile(id_accR211,-).

rule(id_accR211,1,
  [greater(att_1,19.5),less(att_1,25.5),greater(att_2,26)],
  class(o),
  [0.038,0.005,0.495,0.463,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,2,
  [greater(att_1,21.5),less(att_1,27)],
  class(o),
  [0.086,0.000,0.500,0.414,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,3,
  [less(att_1,52.5),greater(att_2,37.5),less(att_2,38.5)],
  class(o),
  [0.017,0.001,0.499,0.483,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,4,
  [less(att_1,49.5),greater(att_2,30.5)],
  class(o),
  [0.314,0.000,0.500,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,5,
  [greater(att_1,49.5),less(att_2,37.5)],
  class(x),
  [0.127,0.007,0.493,0.373,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,6,
  [less(att_1,21.5),less(att_2,21.5)],
  class(x),
  [0.031,0.003,0.497,0.469,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,7,
```

```
  [less(att_1,19),less(att_2,28.5)],
  class(x),
  [0.121,0.000,0.500,0.379,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,8,
  [less(att_1,16.5),less(att_2,35)],
  class(x),
  [0.096,0.000,0.500,0.404,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,9,
  [greater(att_1,50.5)],
  class(x),
  [0.159,0.000,0.500,0.341,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,10,
  [greater(att_1,30.5),less(att_1,40.5)],
  class(x),
  [0.153,0.006,0.494,0.347,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,11,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,12,
  [],
  class(x),
  [],[]).
```

B.1.2 $\mathbf{h}_{1_covR}$

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,30,
%31,32,33,34,35,36,37,38,39,60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79,100,101,102,103,104,105,106,
%107,108,109,110,111,112,113,114,115,116,117,118,119]

inducer(id_covR211,rco).
inputFile(id_covR211,-).
dAte(id_covR211,day(-,'Jan',10,2002),time(11 : 7 : 1)).
evaluatedAs(id_covR211,'UNORDERED').
dataFile(id_covR211,-).

rule(id_covR211,1,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
```

```
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_covR211,2,
  [less(att_1,49.5),greater(att_2,30.5)],
  class(o),
  [0.314,0.000,0.500,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_covR211,3,
  [],
  class(x),
  [],[]).
```

B.1.3 h_{1_satR}

```
%Exemplos não cobertos: [24,37,79]

inducer(id_satR211,rco).
inputFile(id_satR211,-).
dAte(id_satR211,day(-,'Jan',10,2002),time(11 : 9 : 29)).
evaluatedAs(id_satR211,'UNORDERED').
dataFile(id_satR211,-).

rule(id_satR211,1,
  [greater(att_1,19.5),less(att_1,25.5),greater(att_2,26)],
  class(o),
  [0.038,0.005,0.495,0.463,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,2,
  [greater(att_1,21.5),less(att_1,27)],
  class(o),
  [0.086,0.000,0.500,0.414,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,3,
  [less(att_1,52.5),greater(att_2,37.5),less(att_2,38.5)],
  class(o),
  [0.017,0.001,0.499,0.483,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,4,
  [less(att_1,49.5),greater(att_2,30.5)],
  class(o),
  [0.314,0.000,0.500,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,5,
  [greater(att_1,49.5),less(att_2,37.5)],
  class(x),
  [0.127,0.007,0.493,0.373,960.000],[0.000,0.000,0.000,0.000,0.000]).
```

```
rule(id_satR211,6,
  [less(att_1,21.5),less(att_2,21.5)],
  class(x),
  [0.031,0.003,0.497,0.469,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,7,
  [less(att_1,19),less(att_2,28.5)],
  class(x),
  [0.121,0.000,0.500,0.379,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,8,
  [less(att_1,16.5),less(att_2,35)],
  class(x),
  [0.096,0.000,0.500,0.404,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,9,
  [greater(att_1,50.5)],
  class(x),
  [0.159,0.000,0.500,0.341,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,10,
  [greater(att_1,30.5),less(att_1,40.5)],
  class(x),
  [0.153,0.006,0.494,0.347,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,11,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,12,
  [],
  class(x),
  [],[]).
```

B.1.4 $\mathbf{h}_{\mathtt{1_sensR}}$

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,30,
%31,32,33,34,35,36,37,38,39,60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79,100,101,102,103,104,105,106,
%107,108,109,110,111,112,113,114,115,116,117,118,119]

inducer(id_sensR211,rco).
inputFile(id_sensR211,-).
dAte(id_sensR211,day(-,'Jan',10,2002),time(11 : 4 : 36)).
evaluatedAs(id_sensR211,'UNORDERED').
```

```
dataFile(id_sensR211,-).

rule(id_sensR211,1,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_sensR211,2,
  [less(att_1,49.5),greater(att_2,30.5)],
  class(o),
  [0.314,0.000,0.500,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_sensR211,3,
  [],
  class(x),
  [],[]).
```

B.1.5 $\mathbf{h}_{1_specR}$

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,30,
%31,32,33,34,35,36,37,38,39,41,42,43,46,48,49,50,51,53,54,
%55,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,
%78,79,99,102,107,108,114,115]

inducer(id_specR211,rco).
inputFile(id_specR211,-).
dAte(id_specR211,day(-,'Jan',10,2002),time(11 : 5 : 25)).
evaluatedAs(id_specR211,'UNORDERED').
dataFile(id_specR211,-).

rule(id_specR211,1,
  [greater(att_1,19.5),less(att_1,25.5),greater(att_2,26)],
  class(o),
  [0.038,0.005,0.495,0.463,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,2,
  [greater(att_1,21.5),less(att_1,27)],
  class(o),
  [0.086,0.000,0.500,0.414,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,3,
  [less(att_1,52.5),greater(att_2,37.5),less(att_2,38.5)],
  class(o),
  [0.017,0.001,0.499,0.483,960.000],[0.000,0.000,0.000,0.000,0.000]).
```

```
rule(id_specR211,4,
  [less(att_1,49.5),greater(att_2,30.5)],
  class(o),
  [0.314,0.000,0.500,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,5,
  [greater(att_1,49.5),less(att_2,37.5)],
  class(x),
  [0.127,0.007,0.493,0.373,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,6,
  [],
  class(x),
  [],[]).
```

B.1.6 $\mathbf{h}_{1_supR}$

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,30,
%31,32,33,34,35,36,37,38,39,60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79,100,101,102,103,104,105,106,
%107,108,109,110,111,112,113,114,115,116,117,118,119]

inducer(id_supR211,rco).
inputFile(id_supR211,-).
dAte(id_supR211,day(-,'Jan',10,2002),time(11 : 7 : 50)).
evaluatedAs(id_supR211,'UNORDERED').
dataFile(id_supR211,-).

rule(id_supR211,1,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_supR211,2,
  [less(att_1,49.5),greater(att_2,30.5)],
  class(o),
  [0.314,0.000,0.500,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_supR211,3,
  [],
  class(x),
  [],[]).
```

B.2 Experimento 2

B.2.1 $\mathbf{h}_{2_\mathtt{accR}}$

Mesma hipótese que $\mathbf{h}_{1_\mathtt{accR}}$ — Seção B.1.1.

B.2.2 $\mathbf{h}_{2_\mathtt{covR}}$

Mesma hipótese que $\mathbf{h}_{1_\mathtt{covR}}$ — Seção B.1.2.

B.2.3 $\mathbf{h}_{2_\mathtt{satR}}$

Mesma hipótese que $\mathbf{h}_{1_\mathtt{satR}}$ — Seção B.1.3.

B.2.4 $\mathbf{h}_{2_\mathtt{sensR}}$

Mesma hipótese que $\mathbf{h}_{1_\mathtt{sensR}}$ — Seção B.1.4.

B.2.5 $\mathbf{h}_{2_\mathtt{specR}}$

Mesma hipótese que $\mathbf{h}_{1_\mathtt{specR}}$ — Seção B.1.5.

B.2.6 $\mathbf{h}_{2_\mathtt{supR}}$

Mesma hipótese que $\mathbf{h}_{1_\mathtt{supR}}$ — Seção B.1.6.

B.3 Experimento 3

B.3.1 $\mathbf{h}_{3_\mathtt{accR}}$

Mesma hipótese que $\mathbf{h}_{1_\mathtt{accR}}$ — Seção B.1.1.

B.3.2 $\mathbf{h}_{3_\mathtt{covR}}$

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,30,
%31,32,33,34,35,36,37,38,39,60,61,62,63,64,65,66,67,68,69,
```

```
%70,71,72,73,74,75,76,77,78,79,100,101,102,103,104,105,106,
%107,108,109,110,111,112,113,114,115,116,117,118,119]

inducer(id_covR211,rco). inputFile(id_covR211,-).
dAte(id_covR211,day(-,'Jan',15,2002),time(20 : 56 : 50)).
evaluatedAs(id_covR211,'UNORDERED'). dataFile(id_covR211,-).

rule(id_covR211,1,
  [lessORequal(att_1,48),greater(att_2,28)],
  class(o),
  [0.328,0.022,0.478,0.172,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_covR211,2,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_covR211,3,
  [],
  class(x),
  [],[]).
```

B.3.3 $\mathbf{h}_{3_satR}$

Mesma hipótese que $\mathbf{h}_{1_satR}$ — Seção B.1.3.

B.3.4 $\mathbf{h}_{3_sensR}$

Mesma hipótese que $\mathbf{h}_{1_sensR}$ — Seção B.1.4.

B.3.5 $\mathbf{h}_{3_specR}$

Mesma hipótese que $\mathbf{h}_{1_specR}$ — Seção B.1.5.

B.3.6 $\mathbf{h}_{3_supR}$

Mesma hipótese que $\mathbf{h}_{1_supR}$ — Seção B.1.6.

B.4 Experimento 4

B.4.1 $\mathbf{h}_{4_accR}$

```
%Exemplos não cobertos: [60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79]

inducer(id_accR211,rco). inputFile(id_accR211,-).
dAte(id_accR211,day(-,'Mar',14,2002),time(10 : 0 : 50)).
evaluatedAs(id_accR211,'UNORDERED'). dataFile(id_accR211,-).

rule(id_accR211,1,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,2,
  [less(att_1,50.5),greater(att_2,33.5)],
  class(o),
  [0.157,0.000,0.500,0.343,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,3,
  [less(att_1,10.5),less(att_2,32)],
  class(x),
  [0.011,0.000,0.500,0.489,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,4,
  [greater(att_1,10.5),less(att_1,19.5),less(att_2,34.5)],
  class(x),
  [0.144,0.000,0.500,0.356,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,5,
  [greater(att_1,50.5)],
  class(x),
  [0.159,0.000,0.500,0.341,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,6,
  [],
  class(x),
  [],[]).
```

B.4.2 $\mathbf{h}_{4_covR}$

Mesma hipótese que $\mathbf{h}_{3_covR}$ — Seção B.3.2.

B.4.3 h_{4_satR}

```
%Exemplos não cobertos: [11,20,22,26,28]

inducer(id_satR211,rco). inputFile(id_satR211,-).
dAte(id_satR211,day(-,'Mar',14,2002),time(10 : 1 : 35)).
evaluatedAs(id_satR211,'UNORDERED'). dataFile(id_satR211,-).

rule(id_satR211,1,
  [greater(att_1,19),lessORequal(att_1,48),greater(att_2,19)],
  class(o),
  [0.314,0.018,0.482,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,2,
  [greater(att_1,20),lessORequal(att_1,49),greater(att_2,19)],
  class(o),
  [0.021,0.000,0.500,0.479,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,3,
  [lessORequal(att_1,19),lessORequal(att_2,28)],
  class(x),
  [0.135,0.000,0.500,0.365,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,4,
  [lessORequal(att_2,19)],
  class(x),
  [0.160,0.000,0.500,0.340,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,5,
  [greater(att_2,40.5)],
  class(o),
  [0.149,0.000,0.500,0.351,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,6,
  [greater(att_1,49)],
  class(x),
  [0.167,0.008,0.492,0.333,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,7,
  [],
  class(x),
  [],[]).
```

B.4.4 $\mathbf{h}_{4_sensR}$

Deve ser observado que as regras dessa hipótese são as mesmas regras de $\mathbf{h}_{3_covR}$, mas apresentadas em ordem diferente. Pela forma de avaliação da hipótese (*unordered*), pode-se afirmar que as hipóteses são idênticas.

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,30,
%31,32,33,34,35,36,37,38,39,60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79,100,101,102,103,104,105,106,
%107,108,109,110,111,112,113,114,115,116,117,118,119]

inducer(id_sensR211,rco). inputFile(id_sensR211,-).
dAte(id_sensR211,day(-,'Mar',14,2002),time(10 : 1 : 0)).
evaluatedAs(id_sensR211,'UNORDERED'). dataFile(id_sensR211,-).

rule(id_sensR211,1,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_sensR211,2,
  [lessORequal(att_1,48),greater(att_2,28)],
  class(o),
  [0.157,0.020,0.480,0.343,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_sensR211,3,
  [],
  class(x),
  [],[]).
```

B.4.5 $\mathbf{h}_{4_specR}$

```
%Exemplos não cobertos: [11,26,28,100,101,102,103,104,105,
%106,107,108,109,110,111,112,113,114,115,116,117,118,119]

inducer(id_specR211,rco). inputFile(id_specR211,-).
dAte(id_specR211,day(-,'Mar',14,2002),time(10 : 1 : 6)).
evaluatedAs(id_specR211,'UNORDERED'). dataFile(id_specR211,-).

rule(id_specR211,1,
```

```
  [greater(att_1,19),lessORequal(att_1,48),greater(att_2,19)],
  class(o),
  [0.314,0.018,0.482,0.186,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,2,
  [lessORequal(att_1,19),lessORequal(att_2,28)],
  class(x),
  [0.135,0.000,0.500,0.365,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,3,
  [greater(att_2,40.5)],
  class(o),
  [0.149,0.000,0.500,0.351,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,4,
  [greater(att_1,19.5),less(att_1,51),greater(att_2,21)],
  class(o),
  [0.029,0.007,0.493,0.471,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,5,
  [less(att_1,16),less(att_2,30.5)],
  class(x),
  [0.006,0.000,0.500,0.494,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,6,
  [lessORequal(att_2,19)],
  class(x),
  [0.160,0.000,0.500,0.340,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,7,
  [],
  class(x),
  [],[]).
```

B.4.6 $\mathbf{h}_{4_supR}$

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,30,
%31,32,33,34,35,36,37,38,39,60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79,100,101,102,103,104,105,106,
%107,108,109,110,111,112,113,114,115,116,117,118,119]

inducer(id_supR211,rco). inputFile(id_supR211,-).
dAte(id_supR211,day(-,'Mar',14,2002),time(10 : 1 : 29)).
evaluatedAs(id_supR211,'UNORDERED'). dataFile(id_supR211,-).

rule(id_supR211,1,
```

```
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_supR211,2,
  [less(att_1,49.5),greater(att_2,30.5)],
  class(o),
  [0.157,0.000,0.500,0.343,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_supR211,3,
  [],
  class(x),
  [],[]).
```

B.5 Experimento 5

B.5.1 h_{5_accR}

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,
%30,31,32,33,34,35,36,37,38,39]

inducer(id_accR211,rco). inputFile(id_accR211,-).
dAte(id_accR211,day(-,'Mar',14,2002),time(10 : 5 : 26)).
evaluatedAs(id_accR211,'UNORDERED'). dataFile(id_accR211,-).

rule(id_accR211,1,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,21.5),less(att_2,23.5)],
  class(o),
  [0.018,0.001,0.499,0.482,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,2,
  [greater(att_2,40.5)],
  class(o),
  [0.157,0.000,0.500,0.343,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,3,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,27.5)],
  class(o),
  [0.020,0.002,0.498,0.480,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,4,
  [greater(att_1,20),lessORequal(att_1,49),greater(att_2,19)],
  class(o),
  [0.283,0.006,0.494,0.217,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,5,
```

```
  [greater(att_1,49)],
  class(x),
  [0.167,0.008,0.492,0.333,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,6,
  [less(att_1,49.5),greater(att_2,32.5)],
  class(o),
  [0.008,0.000,0.500,0.492,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,7,
  [lessORequal(att_2,19)],
  class(x),
  [0.160,0.000,0.500,0.340,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,8,
  [],
  class(x),
  [],[]).
```

B.5.2 $\mathbf{h}_{5_\mathtt{covR}}$

Mesma hipótese que $\mathbf{h}_{3_\mathtt{covR}}$ — Seção B.3.2.

B.5.3 $\mathbf{h}_{5_\mathtt{satR}}$

```
%Exemplos não cobertos: [100,112,116,119]

inducer(id_satR211,rco). inputFile(id_satR211,-).
dAte(id_satR211,day(-,'Mar',14,2002),time(10 : 6 : 20)).
evaluatedAs(id_satR211,'UNORDERED'). dataFile(id_satR211,-).

rule(id_satR211,1,
  [greater(att_2,40.5)],
  class(o),
  [0.157,0.000,0.500,0.343,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,2,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,21.5),less(att_2,23.5)],
  class(o),
  [0.018,0.001,0.499,0.482,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,3,
  [greater(att_1,19.5),less(att_1,51),greater(att_2,21)],
  class(o),
```

```
  [0.289,0.016,0.484,0.211,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,4,
  [greater(att_1,19),lessORequal(att_1,48),greater(att_2,19)],
  class(o),
  [0.028,0.008,0.492,0.472,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,5,
  [less(att_1,16),less(att_2,30.5)],
  class(x),
  [0.079,0.000,0.500,0.421,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,6,
  [less(att_1,49.5),greater(att_2,32.5)],
  class(o),
  [0.008,0.000,0.500,0.492,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,7,
  [lessORequal(att_1,19),lessORequal(att_2,28)],
  class(x),
  [0.063,0.000,0.500,0.438,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,8,
  [lessORequal(att_2,19)],
  class(x),
  [0.160,0.000,0.500,0.340,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,9,
  [greater(att_1,49.5),greater(att_2,32)],
  class(x),
  [0.121,0.000,0.500,0.379,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,10,
  [less(att_1,19.5),less(att_2,31.5)],
  class(x),
  [0.014,0.000,0.500,0.486,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,11,
  [],
  class(x),
  [],[]).
```

B.5.4 $\mathbf{h}_{5_sensR}$

Mesma hipótese que $\mathbf{h}_{4_sensR}$ — Seção B.4.4.

B.5.5 $\mathbf{h}_{5_specR}$

```
%Exemplos não cobertos: [11,26,28,60,61,62,63,64,65,66,
%67,68,69,70,71,72,73,74,75,76,77,78,79,102,118]

inducer(id_specR211,rco). inputFile(id_specR211,-).
dAte(id_specR211,day(-,'Mar',14,2002),time(10 : 5 : 45)).
evaluatedAs(id_specR211,'UNORDERED'). dataFile(id_specR211,-).

rule(id_specR211,1,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,27.5)],
  class(o),
  [0.028,0.002,0.498,0.472,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,2,
  [greater(att_1,30.5),less(att_1,47),greater(att_2,26.5)],
  class(o),
  [0.109,0.000,0.500,0.391,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,3,
  [greater(att_2,40.5)],
  class(o),
  [0.149,0.000,0.500,0.351,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,4,
  [greater(att_1,19.5),less(att_1,51),greater(att_2,21)],
  class(o),
  [0.177,0.015,0.485,0.323,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,5,
  [greater(att_1,18),less(att_1,25),less(att_2,21.5)],
  class(o),
  [0.013,0.004,0.496,0.488,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,6,
  [less(att_1,16),less(att_2,30.5)],
  class(x),
  [0.079,0.000,0.500,0.421,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,7,
  [less(att_1,19.5),less(att_2,29.5)],
  class(x),
  [0.068,0.000,0.500,0.432,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,8,
  [greater(att_1,51)],
  class(x),
  [0.140,0.000,0.500,0.360,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,9,
  [],
  class(x),
```

```
  [],[]).
```

B.5.6 $\mathbf{h}_{5_supR}$

Deve ser observado que esta hipótese é idêntica à hipótese $\mathbf{h}_{4_sensR}$.

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,30,
%31,32,33,34,35,36,37,38,39,60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79,100,101,102,103,104,105,106,
%107,108,109,110,111,112,113,114,115,116,117,118,119]

inducer(id_supR211,rco). inputFile(id_supR211,-).
dAte(id_supR211,day(-,'Mar',14,2002),time(10 : 6 : 12)).
evaluatedAs(id_supR211,'UNORDERED'). dataFile(id_supR211,-).

rule(id_supR211,1,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_supR211,2,
  [lessORequal(att_1,48),greater(att_2,28)],
  class(o),
  [0.157,0.020,0.480,0.343,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_supR211,3,
  [],
  class(x),
  [],[]).
```

B.6 Experimento 6

B.6.1 $\mathbf{h}_{6_accR}$

```
%Exemplos não cobertos: [60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79,100,112,116,119]

inducer(id_accR211,rco). inputFile(id_accR211,-).
dAte(id_accR211,day(-,'Mar',14,2002),time(10 : 11 : 12)).
```

```
evaluatedAs(id_accR211,'UNORDERED'). dataFile(id_accR211,-).

rule(id_accR211,1,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,21.5),less(att_2,23.5)],
  class(o),
  [0.018,0.001,0.499,0.482,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,2,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.325,0.024,0.476,0.175,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,3,
  [greater(att_2,40.5)],
  class(o),
  [0.149,0.000,0.500,0.351,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,4,
  [less(att_1,49.5),greater(att_2,32.5)],
  class(o),
  [0.008,0.000,0.500,0.492,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,5,
  [greater(att_1,49.5),greater(att_2,32)],
  class(x),
  [0.121,0.000,0.500,0.379,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,6,
  [less(att_1,19.5),less(att_2,31.5)],
  class(x),
  [0.155,0.000,0.500,0.345,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_accR211,7,
  [],
  class(x),
  [],[]).
```

B.6.2 $\mathbf{h}_{6_covR}$

Mesma hipótese que $\mathbf{h}_{3_covR}$ — Seção B.3.2.

B.6.3 $\mathbf{h}_{6_satR}$

```
%Exemplos não cobertos: [20,21,22,23,24,25,26,27,28,29,
%30,31,32,33,34,35,36,37,38,39]
```

```
inducer(id_satR211,rco). inputFile(id_satR211,-).
dAte(id_satR211,day(-,'Mar',14,2002),time(10 : 11 : 58)).
evaluatedAs(id_satR211,'UNORDERED'). dataFile(id_satR211,-).

rule(id_satR211,1,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,27.5)],
  class(o),
  [0.028,0.002,0.498,0.472,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,2,
  [greater(att_1,19.5),less(att_1,25),greater(att_2,21.5),less(att_2,23.5)],
  class(o),
  [0.018,0.001,0.499,0.482,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,3,
  [greater(att_1,20),lessORequal(att_1,49),greater(att_2,19)],
  class(o),
  [0.283,0.006,0.494,0.217,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,4,
  [greater(att_1,49)],
  class(x),
  [0.167,0.008,0.492,0.333,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,5,
  [greater(att_2,40.5)],
  class(o),
  [0.149,0.000,0.500,0.351,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,6,
  [lessORequal(att_2,19)],
  class(x),
  [0.160,0.000,0.500,0.340,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,7,
  [less(att_1,49.5),greater(att_2,32.5)],
  class(o),
  [0.008,0.000,0.500,0.492,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_satR211,8,
  [],
  class(x),
  [],[]).
```

B.6.4 $\mathbf{h}_{6_sensR}$

Mesma hipótese que $\mathbf{h}_{4_sensR}$ — Seção B.4.4.

B.6.5 h_{6_specR}

```
%Exemplos não cobertos: [60,61,62,63,64,65,66,67,68,69,
%70,71,72,73,74,75,76,77,78,79]

inducer(id_specR211,rco). inputFile(id_specR211,-).
dAte(id_specR211,day(-,'Mar',14,2002),time(10 : 11 : 30)).
evaluatedAs(id_specR211,'UNORDERED'). dataFile(id_specR211,-).

rule(id_specR211,1,
  [greater(att_1,19.5),less(att_1,50.5),greater(att_2,19.5)],
  class(o),
  [0.343,0.025,0.475,0.157,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,2,
  [greater(att_2,40.5)],
  class(o),
  [0.149,0.000,0.500,0.351,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,3,
  [less(att_1,50.5),greater(att_2,33.5)],
  class(o),
  [0.008,0.000,0.500,0.492,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,4,
  [less(att_1,10.5),less(att_2,32)],
  class(x),
  [0.011,0.000,0.500,0.489,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,5,
  [greater(att_1,10.5),less(att_1,19.5),less(att_2,34.5)],
  class(x),
  [0.144,0.000,0.500,0.356,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,6,
  [greater(att_1,50.5)],
  class(x),
  [0.159,0.000,0.500,0.341,960.000],[0.000,0.000,0.000,0.000,0.000]).
rule(id_specR211,7,
  [],
  class(x),
  [],[]).
```

B.6.6 $\mathbf{h}_{6_\mathtt{supR}}$

Mesma hipótese que $\mathbf{h}_{5_\mathtt{supR}}$ — Seção B.5.6.

Referências

Baranauskas, J. A. (2001). Extração automática de conhecimento por múltiplos indutores. Tese de Doutorado, ICMC-USP.

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

Baranauskas, J. A. and Monard, M. C. (2000). 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.

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. (2000). Pré-processamento de dados em aprendizado de máquina supervisionado. Monografia do Exame de Qualificação de Doutorado, ICMC-USP.

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.

Bernardini, F. C. (2001). Combinação de classificadores para melhorar o poder preditivo e descritivo de *ensembles*. Monografia do Exame de Qualificação de Mestrado, ICMC-USP.

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

Caulkins, C. W. (2000). Aquisição de conhecimento utilizando aprendizado de máquina relacional. Dissertação de Mestrado, ICMC-USP.

Clark, P. and Niblett, T. (1989). The $\mathcal{CN}2$ induction algorithm. *Machine Learning*, 3(4):261–283.

Félix, L. C. M. (1998). Data mining no processo de extração de conhecimento de bases de dados. Dissertação de Mestrado, ICMC-USP.

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, RJ.

Freitas, A. A. (1998b). On objective measures of rule surprisingness. In *Proceedings of the Second European Symp. LNAI*, volume 1510, pages 1–9.

Gomes, A. K. (2001). Medidas de avaliação de regras. Exame de Qualificação de Mestrado, ICMC-USP.

Gomes, A. K., Bernardini, F. C., and Monard, M. C. (2002). Uma sintaxe padrão Prolog para classificadores simbólicos. Technical Report 154, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_154.ps.zip.

Gomes, A. K. and Monard, M. C. (2002). Um módulo de análise de regras extraídas de classificadores simbólicos. Technical Report 155, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_155.ps.zip.

Hilderman, R. J. and Hamilton, H. J. (1999). Knowledge discovery and interestingness measures: A survey. Technical Report s4s, University of Regina.

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

Imamura, C. Y. (2001). Pré-processamento para extração de conhecimento de bases textuais. Dissertação de Mestrado, ICMC-USP.

Lavrac, N., Flach, P., and Zupan, B. (1999). Rule evaluation measures: a unifying view. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming. LNAI*, volume 1634, pages 74–185.

Lee, H. D. (2000). Seleção e construção de features relevantes para o aprendizado de máquina. Dissertação de Mestrado, ICMC-USP.

Martins, C. A. (2001). Clustering conceitual em aprendizado de máquina. Exame de Qualificação de Doutorado, ICMC-USP.

Milaré, C. R. (2000). Extração de conhecimento de redes neurais. Exame de Qualificação de Doutorado, ICMC-USP.

Nagai, W. A. (2000). Avaliação do conhecimentos extraído de problemas de regressão. Dissertação de Mestrado, ICMC-USP.

Pila, A. D. (2000). Seleção de atributos relevantes para aprrendizado de máquina utilizando a abordagem de rough sets. Dissertação de Mestrado, ICMC-USP.

Prati, R. C., Baranauskas, J. A., and 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., and Monard, M. C. (2001a). Extração de informações padronizadas para a avaliação de regras induzidas por algoritmos de aprendizado de máquina simbólico. Technical Report 145, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_145.ps.zip`.

Prati, R. C., Baranauskas, J. A., and Monard, M. C. (2001b). 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`.

Pugliesi, J. B. (2001). O pós-processamento em extração de conhecimento de bases de dados. Exame de Qualificação de Doutorado, ICMC-USP.

Quinlan, J. (1997). *S*ee5: An informal tutorial. http://www.rulequest.com/ [data: Agosto/2001].

Weiss, S. M. and Kulikowski, C. A. (1991). *Computer Systems that Learn.* Morgan Kaufmann Publishers, San Mateo, CA.

Westwood, D. (2000a). LPA-PROLOG for windows programming guide.

Westwood, D. (2000b). LPA-PROLOG technical reference.