Combinação de Classificadores Simbólicos para Melhorar
o Poder Preditivo e Descritivo de *Ensembles*

Flavia Cristina Bernardini

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 22/04/2002

Assinatura: _____

Combinação de Classificadores Simbólicos para Melhorar o Poder Preditivo e Descritivo de *Ensembles*¹

Flavia Cristina Bernardini

Orientadora: *Profa. Dra. Profa. Dra. Maria Carolina Monard*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação — ICMC/USP — como parte dos requisitos necessários à obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP - São Carlos
Abril/2002

¹Trabalho realizado com o apoio da FAPESP (00/00013-5) — <http://www.fapesp.br>

Resumo

A qualidade das hipóteses induzidas pelos atuais sistemas de Aprendizado de Máquina depende principalmente da quantidade e da qualidade dos atributos e exemplos utilizados no treinamento. Frequentemente, resultados experimentais obtidos sobre grandes bases de dados, que possuem muitos atributos irrelevantes, resultam em hipóteses de baixa precisão. Por outro lado, muitos dos sistemas de aprendizado de máquina conhecidos não estão preparados para trabalhar com uma quantidade muito grande de exemplos. Assim, uma das áreas de pesquisa mais ativas em aprendizado de máquina tem girado em torno de técnicas que sejam capazes de ampliar a capacidade dos algoritmos de aprendizado para processar muitos exemplos de treinamento, atributos e classes.

Para que conceitos sejam aprendidos a partir de grandes bases de dados utilizando Aprendizado de Máquina, pode-se utilizar duas abordagens. A primeira realiza uma seleção de exemplos e atributos mais relevantes, e a segunda é a abordagem de *ensembles*. Um *ensemble* é um conjunto de classificadores cujas decisões individuais são combinadas de alguma forma para classificar um novo caso. Ainda que *ensembles* classifiquem novos exemplos melhor que cada classificador individual, eles se comportam como caixas pretas, no sentido de não oferecer ao usuário alguma explicação relacionada à classificação por eles fornecida.

O objetivo deste trabalho é propor uma forma de combinação de classificadores simbólicos, ou seja, classificadores induzidos por algoritmos de AM simbólicos, nos quais o conhecimento é descrito na forma de regras *if-then* ou equivalentes, para se trabalhar com grandes bases de dados. A nossa proposta é a seguinte: 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 simbólicos. Logo após, as regras que constituem os classificadores induzidos por esses algoritmos são combinadas em um único classificador. Para analisar a viabilidade do objetivo proposto, foi implementado um sistema na linguagem de programação lógica Prolog, com a finalidade de (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 classificadores simbólicos. A finalidade (a) é implementada pelo Módulo de Análise de Regras e a finalidade (b) pelo Módulo de Combinação e Explicação. Esses módulos constituem os módulos principais do $\mathcal{R}_{uleSystem}$.

Neste trabalho, são descritos os métodos de construção de ensembles e de combinação de classificadores encontrados na literatura, o projeto e a documentação do $\mathcal{R}_{uleSystem}$, a metodologia desenvolvida para documentar o sistema $\mathcal{R}_{uleSystem}$, a implementação do Módulo de Combinação e Explicação, objeto de estudo deste trabalho, e duas aplicações do Módulo de Combinação e Explicação. A primeira aplicação utilizou uma base de dados artificiais, a qual nos permitiu observar necessidades de modificações no Módulo de Combinação e Explicação. A segunda aplicação utilizou uma base de dados reais.

Abstract

The hypothesis quality induced by current machine learning algorithms depends mainly on the quantity and quality of features and examples used in the training phase. Frequently, hypothesis with low precision are obtained in experiments using large databases with a large number of irrelevant features. Thus, one active research area in machine learning is to investigate techniques able to extend the capacity of machine learning algorithms to process a large number of examples, features and classes.

To learn concepts from large databases using machine learning algorithms, two approaches can be used. The first approach is based on a selection of relevant features and examples, and the second one is the ensemble approach. An ensemble is a set of classifiers whose individual decisions are combined in some way to classify a new case. Although ensembles classify new examples better than each individual classifier, they behave like black-boxes, since they do not offer any explanation to the user about their classification.

The purpose of this work is to consider a form of symbolic classifiers combination to work with large databases. Given a large database, it is equally divided randomly in small databases. These small databases are supplied to one or more symbolic machine learning algorithms. After that, the rules from the resulting classifiers are combined into one classifier. To analyse the viability of this proposal, was implemented a system in logic programming language Prolog, called $\mathcal{R}_{uleSystem}$. This system has two purposes; the first one, implemented by the Rule Analyses Module, is to evaluate rules induced by symbolic machine learning algorithms; the second one, implemented by the Combination and Explanation Module, is to evaluate several forms of combining symbolic classifiers as well as to explain ensembled classification of new examples. Both principal modules constitute the $\mathcal{R}_{uleSystem}$.

This work describes ensemble construction methods and combination of classifiers methods found in the literature; the project and documentation of $\mathcal{R}_{uleSystem}$; the methodology developed to document the $\mathcal{R}_{uleSystem}$; and the implementation of the Combination and Explanation Module. Two different case studies using the Combination and Explanation Module are described. The first case study uses an artificial database. Through the use of this artificial database, it was possible to improve several of the heuristics used by the the Combination and Explanation Module. A real database was used in the second case study.

Conteúdo

Lista de Figuras	iii
Lista de Tabelas	vi
Lista de Algoritmos	viii
Lista de Abreviaturas	ix
1 Introdução	1
2 Aprendizado de Máquina e <i>Ensembles</i> de Classificadores	5
2.1 Aprendizado de Máquina Supervisionado	5
2.1.1 Classificadores Simbólicos	6
2.1.2 Avaliação de Regras	6
2.2 <i>Ensembles</i> de Classificadores	7
2.3 Razões para a Construção de <i>Ensembles</i>	9
2.4 Métodos de Construção de <i>Ensembles</i>	11
2.4.1 Votação Bayesiana: Enumerando as Hipóteses	12
2.4.2 Manipulando os Exemplos de Treinamento	13
2.4.3 Manipulando os Atributos do Conjunto de Treinamento	14
2.4.4 Manipulando os Valores da Classe	14
2.4.5 Inserindo Aleatoriedade no Algoritmo de Aprendizado	15
2.5 Métodos de Combinação de Classificadores	15
2.5.1 Votação sem Peso	15
2.5.2 Votação com Peso	16
2.5.3 Stacking	16
2.6 Considerações Finais	17
3 O Sistema <i>RuleSystem</i>	19

3.1	Arquitetura do $\mathcal{R}_{uleSystem}$	19
3.2	Metodologia para Documentação do $\mathcal{R}_{uleSystem}$	21
3.2.1	Componentes do Diagrama de Fluxo de Execução de Procedimentos	21
3.2.2	Descrição do Fluxo de Execução do $\mathcal{R}_{uleSystem}$	26
3.3	Considerações Finais	28
4	Conversores de Sintaxe	29
4.1	Conversores de Sintaxe	29
4.2	Sintaxe Padrão de Exemplos	29
4.3	Sintaxe Padrão de Regras	31
4.4	Sintaxe Padrão em Prolog de Exemplos	32
4.5	Sintaxe Padrão em Prolog de Regras	34
4.6	Considerações Finais	37
5	Módulo de Combinação e Explicação	39
5.1	Procedimentos Principais	39
5.2	Classificação de Exemplos	40
5.3	Determinação da Matriz de Confusão	43
5.4	Explicação de <i>Ensembles</i>	43
5.5	Combinação de Classificadores Simbólicos	45
5.5.1	Algoritmo $\mathcal{RCO}$	45
5.5.2	Algoritmo k-Fold Cross Validation	47
5.6	Considerações Finais	49
6	Um Exemplo de Aplicação do MCE	51
6.1	Descrição do Conjunto de Exemplos Artificial	51
6.2	Descrição dos Experimentos Realizados	55
6.3	Análise dos Resultados	61
6.4	Considerações Finais	62
7	Estudo de Caso — Processamento de Sêmen Diagnóstico	63
7.1	O Processamento de Sêmen Diagnóstico	63
7.2	Descrição do Conjunto de Dados	65
7.3	Descrição do Experimento	66
7.4	Análise dos Resultados	68
7.5	Considerações Finais	72

8 Conclusão	73
A Análise Qualitativa de Regras	77
A.1 Avaliação de Regras	77
A.2 Medidas de Avaliação de Regras	80
Referências Bibliográficas	82

Lista de Figuras

2.1	<i>Ensemble</i> de Três Classificadores	8
2.2	Probabilidade do Voto Majoritário Estar Errado com L Hipóteses, $L = 1, \dots, 150$, com Taxa de Erro de Cada Hipótese Igual a 0.3, 0.4 e 0.45	9
2.3	Probabilidade do Voto Majoritário Estar Errado com L Hipóteses, $L = 1, \dots, 150$, com Taxa de Erro de Cada Hipótese Igual a 0.7	10
2.4	Três Motivos Fundamentais que Explicam Porquê os <i>Ensembles</i> Funcionam Melhor que um Único Classificador	10
2.5	Funcionamento do Algoritmo Stacking	17
3.1	Arquitetura do $\mathcal{R}_{uleSystem}$	20
3.2	Diagrama de Fluxo de Execução de Procedimentos do $\mathcal{R}_{uleSystem}$	22
3.3	Exemplo de Representação de um Procedimento	23
3.4	Exemplo de Representação de um Conjunto de Condições	23
3.5	Exemplo de Representação de um Subconjunto de Condições	24
3.6	Tipos de Barramento	25
4.1	Conversores do Formato Padrão $\mathcal{PBM}$ para Sintaxe Padrão em Prolog	30
4.2	Gramática do Arquivo de Nomes (Prati et al., 2001a)	31
4.3	Gramática do Formato Padrão de Regras Estendido (Prati et al., 2001a)	33
4.4	Gramática da Sintaxe Padrão em Prolog para Exemplos	33
4.5	Gramática da Sintaxe Padrão em Prolog para Regras	35
5.1	Subconjuntos de Regras — Forma de Avaliação <i>Interclass</i>	42
5.2	Ilustração do Algoritmo $\mathcal{RCO}$	47
6.1	Função Verdadeira f	52
6.2	Hipóteses Induzidas por $\mathcal{CN}2$ e <i>See5</i> Utilizadas nos Experimentos	53

6.3 Representação Gráfica das Hipóteses Induzidas por $\mathcal{CN}^2$	54
6.4 Representação Gráfica das Hipóteses Induzidas por See^5	55
6.5 Ilustração do Experimento 1.	56
6.6 Ilustração do Experimento 2.	56
6.7 Ilustração do Experimento 3.	57
7.1 Ilustração do Experimento Realizado.	67

Lista de Tabelas

2.1	Conjunto de Exemplos no Formato Atributo-Valor	6
2.2	Medidas Genéricas de Avaliação de Regras	7
3.1	Principais Procedimentos do $\mathcal{R}_{uleSystem}$	20
3.2	Condições Pertencentes ao Conjunto de Condições E	23
3.3	Condições Pertencentes ao Conjunto de Condições R	24
3.4	Condições 1, 2, 3, 4 e 5	24
4.1	Mapeamento dos Dias da Semana	36
4.2	Mapeamento dos Meses do Ano	36
4.3	Valores Nominais de Operadores	37
6.1	Resultados Obtidos Utilizando $\mathcal{CN}2$ e $See5$ na Fase 1 dos Experimentos 1, 2 e 3	57
6.2	Resultados Obtidos nas Hipóteses de Entrada $h_1, \dots, h_7$	58
6.3	Resultados Obtidos Utilizando $\mathcal{RCO}$ — Fase 2 do Experimento 1	58
6.4	Resultados Obtidos Utilizando $\mathcal{RCO}$ — Fase 2 do Experimento 3)	59
6.5	Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 4	60
6.6	Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 5	60
6.7	Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 6	60
6.8	Número de Regras Seleccionadas (com a Regra <i>Default</i>) nos Experimentos 4, 5 e 6	62
7.1	Sumário das Características do Conjunto de Dados <i>Proc-a-gmg-d</i>	66
7.2	Descrição dos Atributos do Conjunto de Dados <i>Proc-a-gmg-d</i>	66
7.3	Resultados Obtidos Utilizando o $\mathcal{CN}2$	68
7.4	Resultados Obtidos Utilizando o $\mathcal{RCO}$	69

7.5	Número de Exemplos Classificados por Classe pela Regra <i>Default</i>	70
7.6	Número de Regras Seleccionadas por Classe	70
7.7	Regras Seleccionadas pelo <i>RCO</i>	71
7.8	Motivos do Especialista para Considerar que uma Regra Não Faz Sentido . .	71

Lista de Algoritmos

1	Avaliação <i>Ordered</i>	41
2	Avaliação <i>Unordered</i>	41
3	Matriz de Confusão	43
4	Explicação de <i>Ensembles</i>	44
5	<i>RCO</i>	46
6	k-Fold Cross Validation	48
7	Construção de Partições	49
8	Cálculo de Informações para Avaliação de Regras <i>Unordered</i>	78
9	Cálculo de Informações para Avaliação de Regras <i>Ordered</i>	79

Lista de Abreviaturas

AM	Aprendizado de Máquina
BF	Base de Fatos
ECOC	Error Correct Output Coding
FIV	Fertilização In Vitro
GEFS	Genetic Ensemble Feature Selection
ICSI	Injeção Intracitoplasmática do Espermatozóide no Oócito
IUI	Inseminação Intra Uterina
LABIC	Laboratório de Inteligência Computacional
MA	Módulo Auxiliar
MAR	Módulo de Análise de Regras
MCE	Módulo de Combinação e Explicação
MD	Mineração de Dados
MT	Mineração de Textos
PEFD	Procedures Execution Flow Diagram
<i>RCO</i>	Rule Combination

Introdução

A qualidade das hipóteses induzidas pelos atuais sistemas de Aprendizado de Máquina (AM) depende principalmente da quantidade e da qualidade dos atributos e exemplos (ou casos) utilizados no treinamento. Frequentemente, resultados experimentais obtidos sobre grandes bases de dados, nas quais muitos atributos irrelevantes estão presentes, resultam em hipóteses de baixa precisão. Por outro lado, muitos dos sistemas de aprendizado de máquina conhecidos não estão preparados para trabalhar com uma quantidade muito grande de exemplos. Assim, uma das áreas de pesquisa mais ativas em aprendizado de máquina tem girado em torno de técnicas que sejam capazes de ampliar a capacidade dos algoritmos de aprendizado para processar muitos exemplos, atributos e classes (Dietterich 2000b; Michalski, Bratko, & Kubat 1998). Esses problemas são típicos da área de mineração de dados em grandes bases (Data Mining) (Cabena, Hadjinian, Stadler, Verhees, & Zanasi 1998; Mannila 1996; Mitchell 1999; Weiss & Indurkha 1998).

Para que os conceitos sejam aprendidos a partir de grandes bases de dados, utilizando AM, pode-se utilizar duas abordagens. A primeira realiza uma seleção de exemplos e atributos mais relevantes (Blum & Langley 1997), e a segunda é a abordagem de *ensembles*, a qual é o foco principal de nosso trabalho. Nessa abordagem, primeiramente são retiradas, por exemplo, L amostras (subconjuntos) do conjunto de exemplos disponível para realizar o aprendizado (exemplos de treinamento). Logo após, cada um desses L subconjuntos é submetido a algum algoritmo de AM induzindo assim L classificadores, os quais podem ser construídos em paralelo. Um *ensemble* é um conjunto de classificadores cujas decisões individuais são combinadas de alguma forma para classificar um novo caso. Deve ser observado que a abordagem de *ensembles* pode ser utilizada com pequenas bases de dados. Na realidade, *ensembles* foram inicialmente propostos para melhorar a precisão de algoritmos de aprendizado de máquina.

Apesar de melhorar o poder de predição dos algoritmos de aprendizado, os métodos de construção de *ensembles* normalmente geram classificadores grandes, o que pode ser indesejável. Por exemplo, em (Margineantu & Dietterich 1997), usando o conjunto de dados

Frey-Slater ([Blake, Keogh, & Merz 1998](#)) com 16 atributos numéricos e 16000 exemplos, é possível obter uma boa precisão no conjunto de testes com 4000 exemplos utilizando 200 hipóteses unindo-as com votação. Incluindo o conjunto de treinamento e o conjunto de testes, o conjunto de dados requer aproximadamente 700Kbytes. Todavia, cada classificador ocupa 295Kbytes de memória, o que implica que o *ensemble* de 200 classificadores requer 58Mbytes — mais de 85 vezes o tamanho do conjunto de dados completo!

Quinlan, em ([Quinlan 1999](#)), explora o algoritmo de Boosting com pequenos conjuntos de classificadores, por 2 motivos:

1. É mais fácil analisar o conhecimento de *ensembles* pequenos e assim fazer uma análise do funcionamento interno da técnica de Boosting;
2. *Ensembles* com baixo número de classificadores são mais baratos, computacionalmente falando, fazendo com que haja uma melhora nas taxas de erro de predição de novos exemplos requerendo poucos recursos adicionais.

Neste trabalho, foram considerados *ensembles* com pequeno número de classificadores (nos experimentos realizados, foram utilizados, no máximo, 7 classificadores). Isso se dá porque, tomadas três hipóteses (classificadores) *h* não correlacionadas, se combinadas com uma votação simples sem peso, por exemplo, a taxa de erro das hipóteses combinadas é menor que as taxas de erro de cada uma das três hipóteses. Isso indica que em um pequeno número de hipóteses pode-se ter uma melhora na precisão, desde que essas hipóteses sejam **independentes**. Em Aprendizado de Máquina, o que se faz é construir hipóteses com correlação mínima entre elas.

Ainda que *ensembles* classifiquem novos exemplos melhor que cada classificador individual, eles se comportam como caixas pretas, no sentido de não oferecer ao usuário alguma explicação relacionada à classificação por eles fornecida. Observa-se, também, que alguns algoritmos de aprendizado de máquina induzem classificadores que não têm condições de oferecer uma explicação que justifique a classificação de novos casos. Em algumas situações, essas explicações são obrigatórias. Por exemplo, em certos países, para dizer ao cliente de um banco se seu crédito foi aprovado ou não, existe a necessidade legal de se justificar o motivo de sua aprovação ou negação. Nesses casos, algoritmos de aprendizado de máquina que induzem classificadores com comportamento do tipo caixas pretas não podem ser utilizados. Assim, perante a necessidade da explicação de classificação em situações em que se faz o uso de aprendizado de máquina, surgiu a idéia de, dado um conjunto de classificadores simbólicos, ou seja, classificadores que podem ser escritos em forma de regras *if-then*, construir outro classificador simbólico que consiste das melhores regras desse conjunto de classificadores inicial. Para essa seleção, diferentes critérios podem ser utilizados, como medidas de avaliação de regras, por exemplo.

No LABIC¹ — Laboratório de Inteligência Computacional —, encontra-se em desenvolvimento um projeto de grande porte denominado DISCOVER, inicialmente proposto por [Barranauskas & Batista \(2000\)](#). O projeto DISCOVER tem como objetivo fornecer um ambiente integrado para apoiar as etapas do processo de descoberta de conhecimento, oferecendo

¹<http://labic.icmc.sc.usp.br>

funcionalidades voltadas para Aprendizado de Máquina (AM) (Batista 1997; Caulkins 2000; Milaré 2000; Martins 2001; Pila 2000), Mineração de Dados²(MD) (Batista 2000; Félix 1998; Horst 1999; Lee 2000; Nagai 2000; Pugliesi 2001; Baranauskas 2001) e Mineração de Textos³(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, Baranauskas, & Monard 2001b; Prati, Baranauskas, & Monard 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, dentre elas a idéia de combinar classificadores simbólicos, foi implementado um sistema computacional, denominado $\mathcal{R}_{uleSystem}$, 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 $\mathcal{R}_{uleSystem}$ 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.

Os dados de entrada para o $\mathcal{R}_{uleSystem}$ 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 em Prolog, que é equivalente à sintaxe padrão $\mathcal{PBM}$ de regras e exemplos adotada no projeto DISCOVER.

²Data Mining

³Text Mining

O objetivo deste trabalho 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. Para analisar a viabilidade desta proposta, foram realizadas as seguintes tarefas, descritas neste trabalho:

- Análise das diversas técnicas existentes de construção de *ensembles* e de combinação de classificadores;
- Metodologia proposta para documentar o $\mathcal{R}_{uleSystem}$;
- Projeto, implementação e documentação do $\mathcal{R}_{uleSystem}$;
- Implementação do Módulo de Combinação e Explicação (MCE) do $\mathcal{R}_{uleSystem}$ ⁴; e
- Comparação dos resultados obtidos utilizando o MCE com resultados obtidos utilizando algoritmos simbólicos de AM, utilizando uma base de dados artificiais e uma base de dados reais. Foram utilizadas bases de dados de um tamanho que tornasse os experimentos realizados controláveis.

Esta dissertação está dividida em 8 capítulos. No Capítulo 2 são abordadas as definições relacionadas a aprendizado de máquina supervisionado e a *ensembles* de classificadores, são descritos alguns métodos de construção de *ensembles*, que podem ser utilizados com diferentes algoritmos de AM, e são descritas as formas de se combinar classificadores para gerar *ensembles*. No Capítulo 3, é descrito o $\mathcal{R}_{uleSystem}$, sua documentação e a metodologia desenvolvida para documentá-lo. No Capítulo 4, são descritos os conversores de formato, que convertem conjuntos de regras e exemplos na sintaxe padrão $\mathcal{PBM}$ para a sintaxe padrão em Prolog. No Capítulo 5, é descrito o Módulo de Combinação e Explicação pertencente ao $\mathcal{R}_{uleSystem}$. No Capítulo 6, é abordado um exemplo de utilização do MCE utilizando um conjunto de dados artificial. No capítulo 7, é descrito um estudo de caso utilizando o MCE e um conjunto de dados do mundo real, os experimentos realizados, os resultados obtidos e é feita uma análise desses resultados. Finalmente, no Capítulo 8, é feita uma conclusão sobre o trabalho realizado e são propostos possíveis trabalhos futuros.

⁴O Módulo de Análise de Regras, outro módulo principal do $\mathcal{R}_{uleSystem}$, está descrito em (Gomes & Monard 2002b; Gomes 2002)

Aprendizado de Máquina e *Ensembles* de Classificadores

Neste capítulo, são abordadas algumas definições e conceitos de aprendizado de máquina supervisionado e de *ensembles* de classificadores. *Ensembles* de classificadores, teoricamente, são mais precisos que os classificadores que o compõem. Para que isso ocorra, algumas condições são necessárias, como a independência entre os classificadores que compõem o *ensemble*, e esses classificadores serem precisos. Assim, na Seção 2.3 são discutidas as condições para que um *ensemble* seja mais preciso que seus componentes individuais, e as razões que motivam a construir um *ensemble*. Também, nas Seções 2.4 e 2.5, são abordados os métodos de construção de *ensembles* e os métodos de combinação de classificadores, respectivamente. Ambas as seções foram baseadas em Dietterich (1997), Dietterich (2000a) e Bauer & Kohavi (1999).

2.1 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 $T_i, i = 1, \dots, N$, escolhidos de um domínio X com uma distribuição D fixa, desconhecida e arbitrária, da forma $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ para alguma função desconhecida $y = f(\mathbf{x})$, como mostra a Tabela 2.1 na página seguinte. Os $\mathbf{x}_i$ são tipicamente vetores da forma $\langle x_{i1}, x_{i2}, \dots, x_{iM} \rangle$, 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 X_j do exemplo T_i , como mostra a Tabela 2.1. Os valores y_i referem-se ao valor do atributo Y , freqüentemente denominado classe.

Os valores de y são tipicamente pertencentes a um conjunto discreto de classes $C_v, v = 1, \dots, N_{Cl}$, da forma $\{C_1, \dots, C_{N_{Cl}}\}$, quando se trata de *classificação* ou ao conjunto de números reais em caso de *regressão*. O enfoque deste trabalho é problemas de classificação.

	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 2.1: Conjunto de Exemplos no Formato Atributo-Valor

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 .

2.1.1 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 $B \rightarrow H$. Quando essa transformação acontece, o classificador composto por regras possui uma regra especial — a regra *default*, a qual possui *Corpo* vazio e a *Cabeça* é dada pela classe que possui maior distribuição de exemplos no conjunto de treinamento. Dado um exemplo x e uma regra R , se o *Corpo* de R é verdade para esse exemplo, então R cobre x .

Pode-se medir um conjunto de regras de um classificador simbólico h dado um conjunto de exemplos S . Neste caso, h consiste de um conjunto de regras com NR regras $R_u, u = 1, \dots, NR$, isto é, $h = \{R_1, \dots, R_{NR}\}$. Ou seja, 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 .

2.1.2 Avaliação de Regras

Várias medidas de avaliação de regras têm sido propostas na literatura (Lavrac, Flach, & Zupan 1999; Freitas 1998b; Freitas 1998a; Hilderman & Hamilton 1999). Neste trabalho, serão utilizadas medidas de avaliação de regras propostas por Lavrac, em (Lavrac, Flach, & Zupan 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.

No Anexo A, encontra-se uma descrição das medidas pertencentes a cada uma dessas categorias.

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 2.2,

bem como o nome pelo qual cada uma dessas medidas é referenciada nos procedimentos do MCE.

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 2.2: Medidas Genéricas de Avaliação de Regras

2.2 Ensembles de Classificadores

Uma das áreas ativas em aprendizado de máquina supervisionado estuda métodos de construção de *ensembles* de classificadores. Um *ensemble* é um conjunto de classificadores cujas decisões individuais são combinadas de alguma forma para classificar um novo caso. Um resultado interessante é que *ensembles* de classificadores podem ser mais precisos que os classificadores individuais que compõem o *ensemble*.

Uma condição para que um *ensemble* de classificadores seja mais preciso que seus componentes é que os classificadores que compõem o *ensemble* sejam distintos (Hansen & Salamon 1990). Um classificador preciso é um classificador que faz a predição da classe de um novo exemplo x com uma margem de erro menor do que simplesmente adivinhar o valor de y dado x . Dois classificadores são distintos se cometem erros diferentes em novos conjuntos de exemplos.

Para melhor compreender essa condição, considera-se um *ensemble* de três classificadores h_1, h_2, h_3 e um novo caso (ou exemplo) x , como mostra a Figura 2.1 (Dietterich 2000a). Esse novo exemplo x será classificado por cada classificador h_1, h_2 e h_3 com uma das classes do conjunto discreto de classes $\{C_1, \dots, C_{NCl}\}$.

Seja $h_1(x)$ a classificação atribuída a esse novo exemplo x pelo classificador h_1 , $h_2(x)$ pelo classificador h_2 e $h_3(x)$ pelo classificador h_3 . Se os três classificadores são idênticos então, quando $h_1(x)$ está errado, $h_2(x)$ e $h_3(x)$ também estão. Entretanto, se os erros cometidos pelos classificadores forem não correlacionados, então quando $h_1(x)$ está errado, $h_2(x)$ e $h_3(x)$ podem estar corretos, de forma que o voto majoritário pode classificar corretamente o exemplo x . Em geral, dado um *ensemble* composto por L classificadores $h_1, \dots, h_L$, para cada novo exemplo x a ser classificado por esses classificadores, tem-se uma série de L ensaios. Considerando que cada um desses ensaios é independente e que cada ensaio é um sucesso na classificação de x com probabilidade¹ p ou uma falha com probabilidade $1 - p$, então a probabilidade do número de sucessos ser l em L ensaios é dada por

¹É considerado que essa probabilidade é a taxa de erro do ensaio, ou seja, da hipótese h_l

Figura 2.1: *Ensemble* de Três Classificadores

$$P(Z = l) = \binom{L}{l} p^l (1 - p)^{L-l} \quad (2.1)$$

Mais precisamente, se as taxas de erro de L classificadores $h_1, \dots, h_L$ são todas iguais a $p < \frac{1}{2}$ e se os erros são independentes, então a probabilidade do voto majoritário estar errado, ou seja, a probabilidade do ensemble ter mais de 50% de classificadores que falharam, é dada por

$$P(Z > \frac{L}{2}) = 1 - \sum_{l=0}^{\frac{L}{2}} \binom{L}{l} p^l (1 - p)^{L-l} \quad (2.2)$$

que corresponde à área sob o gráfico da distribuição binomial na qual mais que $\frac{L}{2}$ hipóteses estão erradas.

Na Figura 2.2, são mostrados *ensembles* ideais formados com hipóteses independentes, todas com taxas de erro de 0.3, 0.4 e 0.45 respectivamente. Pode ser observado que quanto maior o valor da taxa de erro das hipóteses, maior é a taxa de erro do *ensemble*, mas ainda assim a taxa de erro do *ensemble* é menor que a taxa de erro de cada hipótese que o compõe. Por exemplo, para um *ensemble* simulado com 21 hipóteses, cada uma delas possuindo uma taxa de erro de 0.3, a área de curva para 11 ou mais hipóteses simultaneamente incorretas é 0.026, o qual é muito menor que a taxa de erro das hipóteses individuais, conforme pode ser observado na Figura 2.2 na próxima página. Observa-se também que quanto maior o número de hipóteses, menor é a taxa de erro do *ensemble*.

Por outro lado, se as hipóteses individuais têm erros não correlacionados mas com taxas de erro excedendo 0.5, então a taxa de erro do *ensemble* aumentará como resultado da votação. A Figura 2.3 na página 10, mostra a curva do erro de um *ensemble* ideal

Figura 2.2: Probabilidade do Voto Majoritário Estar Errado com L Hipóteses, $L = 1, \dots, 150$, com Taxa de Erro de Cada Hipótese Igual a 0.3, 0.4 e 0.45

composto por hipóteses independentes, todas com taxa de erro de 0.7. Pode ser observado que incrementando o número de hipóteses, maior é o erro cometido pelo *ensemble*.

Para concluir, considerando a probabilidade do voto majoritário estar errado no modelo ideal de *ensembles*, definido pela equação 2.2, a chave para o sucesso dos métodos de criação de *ensembles* está em construir classificadores individuais com taxas de erro abaixo de 0.5. Uma observação é que as simulações mostradas tratam situações ideais, nas quais todas as hipóteses que compõem o *ensemble* são ensaios independentes, ou seja, não correlacionados. Em aprendizado de máquina, o que se tenta fazer é tornar os ensaios (as hipóteses) não correlacionados, ou, se a correlação existir — e, na prática, geralmente essa correlação existe —, torná-la mínima.

2.3 Razões para a Construção de *Ensembles*

Os *ensembles* são construídos para tentar melhorar o poder de predição dos algoritmos de AM. Mas quais as razões que levam a construir bons *ensembles*? Uma ilustração dessas razões pode ser vista na Figura 2.4, as quais são fundamentalmente três, segundo Dietterich (2000a):

1. *Estatística*. Um algoritmo de aprendizado pode ser visto como um algoritmo de busca no espaço de hipóteses $\mathcal{H}$ para identificar a melhor hipótese. O problema estatístico aparece quando a quantidade de dados disponíveis para treinamento é muito pequena comparada ao tamanho do espaço de hipóteses. Sem dados suficientes, o algoritmo de aprendizado pode encontrar muitas hipóteses diferentes em $\mathcal{H}$, as quais têm a mesma

Figura 2.3: Probabilidade do Voto Majoritário Estar Errado com L Hipóteses, $L = 1, \dots, 150$, com Taxa de Erro de Cada Hipótese Igual a 0.7

Figura 2.4: Três Motivos Fundamentais que Explicam Porquê os *Ensembles* Funcionam Melhor que um Único Classificador

precisão sobre os dados de treinamento. Construindo um *ensemble* com todos estes classificadores, o algoritmo pode calcular a média de seus votos e reduzir o risco de escolher o classificador errado. A Figura 2.4 na página anterior (topo à esquerda) ilustra essa situação. A curva externa denota o espaço de hipóteses $\mathcal{H}$ e a interna denota o conjunto de hipóteses que têm uma boa precisão sobre o conjunto de treinamento. O ponto rotulado com a letra f é a hipótese verdadeira, onde pode-se observar que tirando a média da precisão das hipóteses, podemos encontrar uma boa aproximação de f .

2. *Computacional.* Muitos algoritmos de aprendizado trabalham fazendo alguma busca local a qual pode parar em algum ótimo local; por exemplo, alguns algoritmos de redes neurais e algoritmos de árvores de decisão. Alguns algoritmos de redes neurais utilizam métodos de busca local — como gradiente descendente — para encontrar pesos localmente ótimos para a rede. Algoritmos de árvore de decisão aplicam regras gulosas de particionamento para induzir as árvores. Nos casos nos quais há quantidade suficiente de dados de treinamento, o que indica que não há problema estatístico, é computacionalmente difícil para o algoritmo de aprendizado encontrar a melhor hipótese. De fato, o problema de encontrar a menor árvore de decisão que seja consistente com um conjunto de treinamento é NP-hard (Hyafil & Rivest 1976). Similarmente, encontrar os pesos para a menor rede neural possível consistente com os exemplos de treinamento é também NP-hard (Blum & Rivest 1988). Por outro lado, construindo um *ensemble* executando várias vezes o algoritmo de busca local partindo, a cada iteração, de diferentes pontos, pode-se obter uma melhor aproximação da verdadeira (e desconhecida) função f que seja mais precisa que qualquer um dos classificadores individuais, como é ilustrado na Figura 2.4 (topo à direita).
3. *Representacional.* Às vezes, não é possível representar a verdadeira função f pelas hipóteses em $\mathcal{H}$. Entretanto, simplesmente unindo as hipóteses ou dando pesos a cada uma delas e unindo-as posteriormente, pode ser possível expandir o espaço das funções representáveis. A Figura 2.4 (inferior) ilustra esta situação.

2.4 Métodos de Construção de *Ensembles*

Muitos métodos de construção de *ensembles* têm sido desenvolvidos. Esses métodos podem ser divididos, basicamente, em cinco métodos de propósitos gerais (Dietterich 2000a), os quais são: enumerando o espaço de hipóteses $\mathcal{H}$, manipulando os exemplos de treinamento S , manipulando o conjunto de atributos X , manipulando o conjunto de classes C , ou inserindo aleatoriedade nos algoritmos de Aprendizado de Máquina. Neste capítulo, são descritos cada um desses métodos, que podem ser aplicados a diferentes algoritmos de Aprendizado de Máquina em problemas de classificação.

2.4.1 Votação Bayesiana: Enumerando as Hipóteses

Na notação probabilística Bayesiana, é considerado que cada hipótese $\mathbf{h} \in \mathcal{H}$ define uma distribuição de probabilidade condicional:

$$\mathbf{h}(\mathbf{x}) = P(f(\mathbf{x}) = y | \mathbf{x}, \mathbf{h}) \quad (2.3)$$

Dado um novo exemplo $\mathbf{x}$ e uma amostra de treinamento S , o problema de prever o valor de $f(\mathbf{x})$ pode ser visto como o problema de computar

$$P(f(\mathbf{x}) = y | S, \mathbf{x}) \quad (2.4)$$

A Equação 2.4 pode ser reescrita como a soma com peso sobre todas as hipóteses $\mathbf{h}$ em $\mathcal{H}$:

$$P(f(\mathbf{x}) = y | S, \mathbf{x}) = \sum_{\mathbf{h} \in \mathcal{H}} \mathbf{h}(\mathbf{x}) P(\mathbf{h} | S). \quad (2.5)$$

Pode-se enxergar esta equação como sendo um método de construção de *ensemble*: o *ensemble* consiste de todas as hipóteses $\mathbf{h} \in \mathcal{H}$, sendo que cada uma delas possui um peso — sua probabilidade posterior $P(\mathbf{h} | S)$. Pelo teorema de Bayes, a probabilidade posterior é proporcional à probabilidade dos dados de treinamento dada à hipótese anterior e sua probabilidade

$$P(\mathbf{h} | S) \propto P(S | \mathbf{h}) P(\mathbf{h}). \quad (2.6)$$

Em alguns problemas de aprendizado, é possível enumerar completamente cada hipótese $\mathbf{h} \in \mathcal{H}$, computando os valores de $P(S | \mathbf{h})$ e de $P(\mathbf{h})$, e (após a normalização), avaliar o *ensemble* (ou comitê) Bayesiano. Além disso, se a verdadeira função f é retirada de $\mathcal{H}$, segundo $P(\mathbf{h})$, então o esquema de votação Bayesiano é ótimo.

O esquema de votação Bayesiano considera primeiramente a componente estatística dos *ensembles*. Quando o conjunto dos exemplos de treinamento é pequeno, muitas hipóteses $\mathbf{h} \in \mathcal{H}$ terão um valor significativamente alto de probabilidade posterior $P(\mathbf{h} | S)$ e o processo de votação pode, na média, aproximar essas hipóteses da função f . Quando o conjunto de treinamento é grande, é típico que somente uma hipótese tenha probabilidade posterior substancial, e o *ensemble* efetivamente tende a conter somente uma hipótese $\mathbf{h}$.

Em problemas complexos onde o conjunto de hipóteses $\mathcal{H}$ não pode ser enumerado, há a possibilidade de aproximar a votação Bayesiana retirando uma amostra randômica de hipóteses, distribuídas de acordo com $P(\mathbf{h} | S)$. Alguns trabalhos em métodos de cadeias de Markov Monte Carlo visam desenvolver um conjunto de ferramentas para resolver este problema (Neal 1993).

O aspecto mais idealizado da análise Bayesiana é a confiança anterior $P(\mathbf{h})$. Se essa confiança captura completamente todo o conhecimento que se tem sobre f antes de obter S , então, pela definição, não se pode fazer melhor do que foi feito — o ótimo do algoritmo

foi alcançado. Mas, na prática, é muitas vezes difícil construir um espaço $\mathcal{H}$ de hipóteses e atribuir a probabilidade $P(h)$ que captura adequadamente nosso conhecimento prévio. De fato, muitas vezes $\mathcal{H}$ e $P(h)$ são escolhidos levando-se em conta somente a conveniência computacional, ainda que essas escolhas são reconhecidamente inadequadas. Nesses casos, o comitê Bayesiano não é ótimo, e outros métodos de construção de *ensembles* podem produzir melhores resultados.

Deve ser ressaltado que o paradigma Bayesiano não trata dos problemas computacionais e representacionais de um modo significativo.

2.4.2 Manipulando os Exemplos de Treinamento

O segundo método de construção de *ensembles* se dá manipulando os exemplos de treinamento para induzir hipóteses. Neste caso, executa-se o algoritmo de aprendizado várias vezes, cada vez com um subconjunto de treinamento diferente. Esta técnica trabalha especialmente bem quando se utilizam algoritmos de aprendizado instáveis. Algoritmos instáveis são aqueles que induzem classificadores bastante diferentes mesmo ocorrendo pequenas alterações no conjunto de treinamento, ou seja, não é necessário que grandes alterações ocorram no conjunto de treinamento para que classificadores diferentes sejam induzidos. Algoritmos de aprendizado de árvores de decisão, de redes neurais e de regras são instáveis, enquanto que algoritmos de regressão linear, vizinho mais próximo (*nearest neighbor*) e *linear threshold* são geralmente bastante estáveis.

As principais técnicas que manipulam o conjunto de treinamento para a construção de um *ensemble* podem ser divididos em duas famílias (Breiman 1996b; Bauer & Kohavi 1999):

1. *Família P & C*. Técnicas P&C — *Perturb and Combine* (Perturbar e Combinar) — “perturbam” o conjunto de treinamento ou o método de construção da hipótese para induzir diferentes hipóteses as quais são combinadas para gerar o *ensemble*. Técnicas pertencentes à esta família são **Bagging** (Breiman 1996a), **Wagging** (Bauer & Kohavi 1999) e **Cross-validated Committees** (Parmanto, Munro, & Doile 1992).
2. *Família ARC*. ARC — *Adaptively Resample and Combine* (Reamostrar e Combinar Adaptativamente) — refere-se à família de algoritmos que combinam e refazem a amostra de exemplos de treinamento adaptativamente. Técnicas pertencentes a esta família são **Boosting**² (Freund & Schapire 1997; Breiman 1996b), **Arcing** (Bauer & Kohavi 1999) e **Windowing** (Quinlan 1988).

²Durante o desenvolvimento deste trabalho, foram feitos alguns experimentos com a técnica de Boosting, para uma melhor compreensão dessa técnica. Esses experimentos estão descritos em (Bernardini & Monard 2002c).

2.4.3 Manipulando os Atributos do Conjunto de Treinamento

O terceiro método para gerar múltiplos classificadores é manipulando o conjunto de atributos disponíveis para o algoritmo de aprendizado. A seguir são descritas sucintamente duas técnicas que manipulam os atributos do conjunto de treinamento.

Dividindo Aleatoriamente o Conjunto de Atributos

Esta técnica consiste em dividir aleatoriamente o conjunto de atributos em diferentes subconjuntos de atributos e, para cada subconjunto de atributos, construir uma hipótese diferente. Os exemplos considerados são os mesmos para cada hipótese. Esta técnica trabalha muito bem quando o conjunto de atributos é altamente redundante (Dietterich 2000a).

Seleção de Atributos Utilizando Algoritmos Genéticos

O algoritmo **GEFS** — *Genetic Ensemble Feature Selection* — criado por Opitz (1999) utiliza algoritmos genéticos para induzir um conjunto preciso e diverso de classificadores. Inicialmente o algoritmo cria uma população de classificadores, cada um deles rotulado com um valor inteiro. Esta população inicial é criada com um subconjunto de atributos. Seja $F = X_1, X_2, \dots, X_M$ o conjunto de M atributos inicial. Para cada classificador $h_l, l = 1, \dots, L$, o tamanho de cada subconjunto de atributos (F_l) é independentemente escolhido de uma distribuição uniforme entre 1 e $2M$. Esses subconjuntos F_l são replicações *bootstrap* (Efron & Tibshirani 1993) do conjunto inicial F . Cada atributo tem sua codificação genética e, assim, os melhores atributos sobrevivem.

2.4.4 Manipulando os Valores da Classe

O quarto método para construir um bom *ensemble* de classificadores é manipulando os valores do atributo classe dado ao algoritmo de aprendizado. Em (Dietterich & Bakiri 1995) é descrita uma técnica chamada correção de erro do valor de saída (*Error Correct Output Coding* — **ECOC**). Considerando que o número de classes N_{Cl} é muito grande, então novos problemas de aprendizado podem ser construídos particionando-se randomicamente o conjunto de classes em dois subconjuntos A_l e B_l . Os dados de entrada podem ser então rotulados como pertencentes ao subconjunto A_l (sua classe é 0 (zero)) ou ao subconjunto B_l (sua classe é 1 (um)). Este conjunto com um novo valor de classe é dado ao algoritmo de aprendizado o qual cria uma hipótese h_l . Repetindo esse processo L vezes (induzindo diferentes conjuntos A_l e B_l), obtém-se um conjunto de L hipóteses $h_1, \dots, h_L$.

Agora, dado um novo exemplo x , ele deve ser classificado por cada hipótese h_l . Se $h_l(x) = 0$ então todas as classes pertencentes ao subconjunto A_l recebem um voto; caso contrário, todas as classes pertencentes ao subconjunto B_l recebem um voto. Ao final, x é rotulado com a classe que tem o maior número de votos.

Para problemas com mais de duas classes, ECOC pode construir bons *ensembles*. Todavia, devido ao ECOC encontrar dificuldades em problemas de classificação com somente duas classes, é importante que o conjunto de treinamento seja bastante expressivo para este algoritmo (Dietterich 2002).

2.4.5 Inserindo Aleatoriedade no Algoritmo de Aprendizado

Este é o último método de propósito geral para gerar *ensembles* de classificadores — inserindo aleatoriedade nos algoritmos de AM. Por exemplo, há dois algoritmos de AM onde se pode inserir aleatoriedade: o de backpropagation, que induz uma função matemática que representa uma rede neural, e o C4.5, que induz árvores de decisão (Dietterich 2000a).

No algoritmo de backpropagation, para gerar uma rede neural, são atribuídos pesos iniciais aleatórios. Se o algoritmo é aplicado ao mesmo conjunto de treinamento mas com pesos iniciais diferentes, podem ser induzidas hipóteses bastante diferentes (Kolen & Pollock 1991). Com estas diferentes hipóteses, pode-se gerar um *ensemble* de redes neurais da forma mais simples possível. Apesar desta técnica ser a mais simples, manipulando o conjunto de treinamento pode-se gerar bons resultados (Dietterich 2000a).

No algoritmo C4.5, quando se faz a seleção dos possíveis atributos para gerar os nós da árvore, seleciona-se aleatoriamente o melhor dentre os m melhores atributos. São induzidas, assim, L hipóteses diferentes, as quais são combinadas para gerar um *ensemble* de classificadores. Esta técnica é denominada **Randomization** (Dietterich 2000b).

2.5 Métodos de Combinação de Classificadores

Dado um conjunto (*ensemble*) de classificadores, muitos métodos têm sido explorados para se combinar as decisões individuais de cada um dos classificadores que constituem o *ensemble*. Para problemas de classificação, esses métodos podem ser divididos em Votação sem Peso, Votação com Peso e Stacking; neste capítulo, são descritos esses três métodos.

2.5.1 Votação sem Peso

Este é o método mais simples de combinação de classificadores. Bagging, ECOC, e muitas outras técnicas utilizam esse método de combinação. Pode parecer que esquemas de votação mais inteligentes poderiam ser melhores, mas na verdade a experiência mostra que a votação sem peso é bastante robusta.

Um refinamento na simples votação pela maioria é apropriado quando cada classificador $h_l, l = 1, \dots, L$ pode produzir uma estimativa da probabilidade da classe ao invés de uma simples decisão de classificação. Uma estimativa de probabilidade de classe para um dado exemplo x é a probabilidade de que a classe verdadeira seja $P(f(x) = C_{NCl} | h_l)$, para $v = 1, \dots, NCl$ e $l = 1, \dots, L$. Pode-se combinar a probabilidade da classe de todas as hipóteses e, assim, a probabilidade do *ensemble* será

$$P(f(x) = v) = \frac{1}{L} \sum_{l=1}^L P(f(x) = v | h_l) \quad (2.7)$$

A classe predita de x é tomada como a classe que tem a maior probabilidade.

2.5.2 Votação com Peso

Muitos métodos diferentes de votação com peso têm sido desenvolvidos para *ensembles*, tanto para problemas de regressão quanto para problemas de classificação.

Para problemas de regressão, [Perrone & Cooper \(1993\)](#) e [Hashem \(1993\)](#) aplicam o método dos mínimos quadrados para encontrar pesos que maximizam a precisão do *ensemble* no conjunto de treinamento. Nesses trabalhos é mostrado que o peso aplicado a h_l deve ser inversamente proporcional à estimativa da variância de h_l . Entretanto, uma das dificuldades encontradas com o uso do método de mínimos quadrados é que as hipóteses $h_l, l = 1, \dots, L$ podem ser altamente correlacionadas. Assim, [Perrone & Cooper \(1993\)](#) e [Hashem \(1993\)](#) propõem alguns métodos para selecionar do conjunto de hipóteses $h_l, l = 1, \dots, L$, um subconjunto com pouca correlação ao qual é aplicado o método dos mínimos quadrados para determinar os pesos desse subconjunto de hipótese que constituirá o *ensemble*.

Para problemas de classificação, os pesos são usualmente obtidos medindo a precisão de cada classificador individual h_l no conjunto de treinamento ou de teste, construindo pesos que são proporcionais a essas precisões. [Ali & Pazzani \(1996\)](#) descrevem um método, chamado *likelihood combination*, o qual aplica o algoritmo de Naive Bayes para aprender pesos dos classificadores. No AdaBoost, o peso de cada classificador h_l é computado utilizando sua precisão no conjunto de treinamento com os pesos que foram usados para induzir h_l . Este método requer a definição de uma probabilidade prévia $P(h_l)$ a qual é multiplicada por $P(S|h_l)$ para estimar a probabilidade de cada h_l .

2.5.3 Stacking

Este método combina diferentes hipóteses induzidas por diferentes algoritmos de AM da seguinte forma: Supondo um conjunto de algoritmos diferentes $A_1, \dots, A_L$ e um conjunto de treinamento S conforme já descrito, aplica-se cada um destes algoritmos $A_l, l = 1, \dots, L$ ao conjunto de treinamento S para induzir o conjunto de hipóteses $\{h_1, \dots, h_L\}$. A meta do método Stacking é encontrar uma boa combinação desse conjunto de hipóteses, denominada h^* . [Wolpert \(1992\)](#) propôs o seguinte esquema para aprender h^* usando uma forma de *leave-one-out cross validation*, conforme mostra a Figura 2.5 na página oposta:

1. Considere $h_l^{(-i)}$ como sendo a hipótese construída pelo algoritmo de aprendizado A_l utilizando como conjunto de treinamento todos os N exemplos de S com exceção do i -ésimo exemplo x . Ou seja, cada algoritmo é aplicado ao conjunto de treinamento N vezes, deixando de fora um exemplo de treinamento de cada vez.
2. Aplique cada classificador $h_l^{(-i)}$ ao exemplo x_i para obter a classe predita $\hat{y}_i^l$. Esse procedimento constrói um novo conjunto de dados contendo exemplos no “nível 2” cujos atributos são as classes preditas por cada um dos L classificadores. Ou seja, cada um desses exemplos é da forma $\hat{y}_i = \langle \hat{y}_i^1, \dots, \hat{y}_i^L \rangle, i = 1, \dots, N$ rotulado com a classe y_i do exemplo x_i . Assim, o novo conjunto de dados no nível 2 é $S^* = \{(\hat{y}_1, y_1), \dots, (\hat{y}_N, y_N)\}$ onde $\hat{y}_i = \langle \hat{y}_i^1, \dots, \hat{y}_i^L \rangle, i = 1, \dots, N$.

3. Finalmente, aplica-se algum algoritmo de aprendizado a este novo conjunto de treinamento S^* para aprender h^* .

Breiman aplicou este método combinando diferentes formas de regressões lineares e obteve bons resultados (Breiman 1996c).

Figura 2.5: Funcionamento do Algoritmo Stacking

2.6 Considerações Finais

Neste capítulo, foram abordadas definições de AM supervisionado e de *ensembles* de classificadores, bem como as principais razões para construir *ensembles*. Também, foram abordados os diferentes métodos de construção de *ensembles* e foram descritos métodos de combinação de classificadores que resolvem problemas de classificação, objeto de estudo deste trabalho. Para problemas de regressão, existem outros métodos interessantes de combinação, tais como Funções de Ganho (Jordan & Jacobs 1994).

Os métodos de construção de *ensembles* e combinação de classificadores descritos são bons métodos quando se utiliza os classificadores como caixas-pretas. A proposta deste trabalho é combinar classificadores simbólicos, utilizando um critério (medida) de avaliação de regra para selecionar as regras que formam o classificador final, bem como para selecionar as regras do *ensemble* que melhor explicam a classificação de novos exemplos. Portanto, há a necessidade de se utilizar as regras dos classificadores individualmente, fazendo com que esses classificadores deixem de ser caixas-pretas. Assim, os métodos descritos neste capítulo não serão utilizados.

Para testar essa idéia, ou seja, a idéia de construir um classificador simbólico, resultante da seleção das melhores regras de classificadores iniciais utilizando um critério de seleção de regras, foi implementado o sistema $\mathcal{R}_{ule}S_{ystem}$, descrito no próximo capítulo.

O Sistema $\mathcal{R}_{uleSystem}$

Para testar as idéias de explicação de *ensembles* e combinação de classificadores simbólicos utilizando medidas de avaliação de regras, foi implementado um sistema na linguagem de programação lógica Prolog, denominado $\mathcal{R}_{uleSystem}$. A seguir, é descrita a metodologia desenvolvida para documentar a arquitetura do $\mathcal{R}_{uleSystem}$ e o sistema propriamente dito.

3.1 Arquitetura do $\mathcal{R}_{uleSystem}$

A Figura 3.1 mostra a arquitetura do $\mathcal{R}_{uleSystem}$. 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 contêm conjuntos de regras e o conjunto de exemplos na sintaxe padrão em 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.

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 $\mathcal{R}_{uleSystem}$ 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 3.1 mostra os procedimentos principais (<nome do procedimento>/<aridade>) acessíveis ao usuário, implementados em cada módulo do $\mathcal{R}_{uleSystem}$.

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

Figura 3.1: Arquitetura do $\mathcal{R}_{uleSystem}$

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 3.1: Principais Procedimentos do $\mathcal{R}_{uleSystem}$

um procedimento quanto pós-condições geradas após a execução de um procedimento, são armazenadas na Base de Fatos.

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

3.2 Metodologia para Documentação do $\mathcal{R}_{uleSystem}$

Quando foi iniciada a implementação do $\mathcal{R}_{uleSystem}$ 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 $\mathcal{R}_{uleSystem}$. O PEFD do $\mathcal{R}_{uleSystem}$ é apresentado na Figura 3.2.

3.2.1 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.

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 $\mathcal{R}_{uleSystem}$, são 3 os módulos — MA, MAR e MCE. A Figura 3.3 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 3.2 que os procedimentos `listOfExamples/1` e `dictionary/2` não são acessíveis ao usuário.

Figura 3.2: Diagrama de Fluxo de Execução de Procedimentos do $\mathcal{R}_{uleSystem}$

Figura 3.3: Exemplo de Representação de um Procedimento

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:

- $\langle \text{letra} \rangle$ nomeia um conjunto de condições;
- $\langle \text{letra} \rangle \langle \text{números} \rangle$ nomeia a condição $\langle \text{números} \rangle$ que pertence ao conjunto de condições nomeado $\langle \text{letra} \rangle$; e
- $\langle \text{número} \rangle$ nomeia uma única condição que não pertence a nenhum conjunto de condições.

No $\mathcal{R}_{uleSystem}$, essas condições são:

$\langle \text{letra} \rangle := E | R$ onde $E = \{E1, E2, E3\}$ e $R = \{R1, R2, R3, R4, R5, R6, R7\}$

$\langle \text{número} \rangle := 1 | 2 | 3 | 4 | 5$

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 em Prolog. A sintaxe padrão em Prolog é tratada com maiores detalhes no Capítulo 4. A Figura 3.4 ilustra a representação do conjunto de condições E .

Figura 3.4: 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 3.2.

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 3.2: Condições Pertencentes ao Conjunto de Condições E

Analogamente, uma descrição dos fatos relacionados a cada condição que pertence ao conjunto de condições R é encontrada na Tabela 3.3.

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 <code>.names</code> relacionado ao conjunto de exemplos utilizado para induzir as regras.
R6: dataFile/2	Identifica o arquivo <code>.data</code> 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 3.3: 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 compõem, separadas por vírgula. Um exemplo pode ser visualizado na Figura 3.5, onde é ilustrado o subconjunto $\{R1, R4 \text{ e } R7\}$ de R .

Figura 3.5: Exemplo de Representação de um Subconjunto de Condições

Finalmente, na Tabela 3.4, encontra-se 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 3.4: Condições 1, 2, 3, 4 e 5

Foram também definidos os seguintes dois símbolos de representação do estado de uma condição, na Base de Fatos, 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 3.2, 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 3.2. 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.

Fluxo de Execução

O fluxo temporal é representado através de linhas de tempo $t_0, t_1, t_2, \dots, t_q$ as quais são somente aplicáveis a procedimentos representados dentro de um retângulo tracejado¹. Procedimentos fora desse retângulo tracejado são *atemporais*, ou seja, podem ser executados a qualquer momento.

No caso do $\mathcal{R}_{uleSystem}$, foram identificadas cinco linhas de tempo, ou seja, t_0, t_1, t_2, t_3, t_4 — Figura 3.2. 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 na linha t_1 devem ser executados antes que os procedimentos na 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 tracejada², denominada *barramento*. Existem três tipos de barramento, ilustrados na Figura 3.6, e explicados a seguir:

Figura 3.6: Tipos de Barramento

¹tracejado do tipo { - - - - }.

²tracejado do tipo { - - - }.

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.

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 3.2, 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`.

3.2.2 Descrição do Fluxo de Execução do $\mathcal{R}_{ule}S_{ystem}$

Para um melhor entendimento do PEFD do $\mathcal{R}_{ule}S_{ystem}$ (Figura 3.2), aqui é apresentada uma descrição do fluxo de execução dos procedimentos principais implementados nesse sistema.

Na parte superior da Figura 3.2 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 $\mathcal{R}_{ule}S_{ystem}$ 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 `tt 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 e 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 `evaluateAllSetOfRulesFrequency/1`, `indAttSurp/1` e `rco/4` podem ser executados em qualquer ordem. O procedimento `evaluateAllSetOfRulesFrequency/1` está fortemente conectado ao conjunto de condições R e fracamente conectado ao conjunto de condições E e às condições 1 e 2; o procedimento `indAttSurp/1` está fortemente conectado ao conjunto R e fracamente conectado ao conjunto E ; e, por fim, `rco/4` está fortemente conectado ao conjunto R e fracamente conectado às condições 1 e 2.

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 $\mathcal{R}_{uleSystem}$ nos instantes t_3 e t_4 é semelhante ao descrito no instante t_2 .

Em qualquer dos instantes $t_0, \dots, t_4$ 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.

3.3 *Considerações Finais*

Neste capítulo, foi detalhada a documentação do $\mathcal{R}_{uleSystem}$ e a metodologia desenvolvida para essa documentação. No próximo capítulo, são descritos os conversores de sintaxe, que tomam como dados de entrada conjuntos de regras e exemplos na sintaxe padrão $\mathcal{PBM}$ do projeto DISCOVER e convertem esses conjuntos, respectivamente, para conjuntos de regras e exemplos na sintaxe padrão em Prolog. A sintaxe padrão em Prolog é a sintaxe padrão de conjuntos de regras e exemplos aceitos como dados de entrada do $\mathcal{R}_{uleSystem}$.

Conversores de Sintaxe

Neste capítulo, é descrita a sintaxe padrão *PBM* do projeto DISCOVER e a forma como se dá a conversão da sintaxe padrão *PBM* para a sintaxe padrão na linguagem de programação lógica Prolog dos conjuntos de regras e exemplos a serem armazenados na Base de Fatos do $\mathcal{R}_{uleSystem}$.

4.1 Conversores de Sintaxe

A conversão de regras e exemplos no formato padrão *PBM* para a sintaxe padrão de regras e exemplos na linguagem de programação lógica Prolog foi implementada através de *scripts* Perl. A Figura 4.1 ilustra a interação desses *scripts* na conversão da sintaxe de diferentes conjuntos de regras e de exemplos, no formato padrão do DISCOVER (*PBM*), para a sintaxe padrão em Prolog.

Estes conversores foram feitos em conjunto com Alan Keller Gomes, integrante do LA-BIC. Uma referência completa sobre a sintaxe padrão em Prolog proposta bem como a implementação por nós realizada é encontrada em (Gomes, Bernardini, & Monard 2002).

4.2 Sintaxe Padrão de Exemplos

A sintaxe padrão de exemplos proposta em (Batista 2001) é uma extensão do formato dos arquivos de nomes e dados do $\mathcal{C}4.5$ (Quinlan 1988) e utiliza arquivos do tipo texto para declarar os nomes dos atributos (e seus respectivos domínios) bem como os valores que esses atributos assumem no conjunto de exemplos.

Os nomes dos atributos são declarados em um arquivo de nomes, designado com a extensão *.names*. A Figura 4.2 mostra a gramática que define a sintaxe do arquivo de nomes. Os valores que esses atributos assumem no conjunto de exemplos são declarados no arquivo de dados, designados com a extensão *.data*. Um conjunto de exemplos somente está na sintaxe padrão se estiverem presentes os dois arquivos (*.names* e *.data*) que devem

Figura 4.1: Conversores do Formato Padrão *PBM* para Sintaxe Padrão em Prolog

possuir o mesmo nome, se diferenciando apenas pela extensão.

Os atributos declarados em um arquivo de nomes podem assumir qualquer um dos seguintes tipos de dados:

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

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

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

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

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

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

Time: O tipo de dado `time` permite declarar um atributo que pode conter um horário

```

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

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

<feature-name> ::= <identifier>

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

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

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

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

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

```

Figura 4.2: Gramática do Arquivo de Nomes (Prati, Baranauskas, & Monard 2001a)

(hora, minuto e segundo). A princípio, os valores dos horários devem estar no formato “hh:mm:ss”.

A primeira declaração em um arquivo de nomes define qual deve ser o nome do atributo classe. O atributo classe pode ser qualquer atributo presente no conjunto de exemplos. Após a declaração do atributo classe, são declarados os demais atributos.

O arquivo de dados correspondente a um arquivo de nomes contém n exemplos, um exemplo em cada linha. Cada exemplo i consiste nos valores dos atributos desse exemplo, separados por vírgula, que devem pertencer ao domínio correspondente aos atributos especificados no arquivo de nomes. Valores “desconhecidos” são representados por um ponto de interrogação (?) e valores “não-se-aplica” são representados com um ponto de exclamação (!).

4.3 Sintaxe Padrão de Regras

No intuito de unificar a linguagem de representação do conhecimento induzido (classificador) por diferentes algoritmos de AM simbólico, foi proposto em (Prati, Baranauskas, & Monard 2001b) o formato padrão de regras $\mathcal{PBM}$, do projeto DISCOVER.

Um classificador simbólico é qualquer classificador que possa ser transformado num conjunto de regras do tipo *if* <condição> *then* <classe = C_v >. Para realizar essa transformação, foi implementada uma biblioteca composta por uma série de *scripts* Perl (Prati, Baranauskas, & Monard 2001b).

Logo após, derivadas de uma tabela de contingência, foi definido um conjunto mínimo de informações que permitem calcular medidas de avaliação de regras. Esse conjunto de

informações é obtido através da avaliação de cada uma das regras do conjunto de regras que constituem o classificador, no formato padrão de regras $\mathcal{PBM}$, utilizando um conjunto de exemplos fornecido pelo usuário. Como uma extensão do formato $\mathcal{PBM}$, após serem calculadas, essas informações são adicionadas ao final de cada uma das regras (Prati, Baranauskas, & Monard 2001a).

Usando a notação de regra $Body \rightarrow Head$ ou resumidamente $B \rightarrow H$, o formato padrão proposto para as informações adicionais é o seguinte:

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

onde n é o número total de exemplos e f representa a frequência relativa de:

hb número de exemplos para os quais B é verdade e H é verdade.

$b\bar{h}$ número de exemplos para os quais B é verdade e H é falso.

$\bar{b}h$ número de exemplos para os quais B é falso e H é falso.

$\bar{b}\bar{h}$ número de exemplos para os quais B é falso e H é verdade.

O formato $\mathcal{PBM}$ estendido é definido pela gramática na Figura 4.3. Esse formato padrão de regras contém as frequências relativas f_{bh} , $f_{b\bar{h}}$, $f_{\bar{b}h}$, e $f_{\bar{b}\bar{h}}$, além do número de exemplos, n , utilizado no cálculo das métricas de avaliação de regras. Deve ser observado que tanto valores conhecidos como valores desconhecidos nos exemplos são considerados no cálculo dessas métricas. Valores absolutos das medidas também podem ser facilmente obtidos multiplicando as frequências relativas por n .

4.4 Sintaxe Padrão em Prolog de Exemplos

Partindo da sintaxe padrão $\mathcal{PBM}$ do projeto DISCOVER, a conversão de um conjunto de exemplos nessa sintaxe padrão para a sintaxe padrão em Prolog é feita através de um *script* Perl. Por sua vez, esse *script* utiliza uma biblioteca que implementa um grupo de sub-rotinas de manipulação de conjuntos de dados na sintaxe padrão. A Figura 4.4 apresenta a gramática que define a sintaxe padrão em Prolog para um conjunto de exemplos.

Como visto anteriormente, um conjunto específico de exemplos na sintaxe padrão está definido através das informações contidas nos arquivos `.data` e `.names`. Assim, o predicado `ex/2` é utilizado para representar em Prolog cada um dos exemplos contidos no arquivo `.data`. Os dois argumentos do predicado `ex/2` são:

- <arg-1> posição (ou número) do exemplo no arquivo `.data`, começando a partir de 0 (zero);
- <arg-2> lista com os valores que cada atributo pode assumir. A ordem desses valores na lista está dada pela ordem em que os atributos estão declarados no arquivo `.names`.

Os predicados `feature/3` e `classFeature/1` são utilizados para representar em Prolog as informações sobre os atributos contidos no arquivo `.names`.

Os três argumentos do predicado `feature/3` são:


```

S ::= <rule>

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

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

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

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

<comparison> ::= <attribute><operator><value>

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

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

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

<class> ::= CLASS = <value>

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

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

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

```

Figura 4.3: Gramática do Formato Padrão de Regras Estendido (Prati, Baranauskas, & Monard 2001a)

```

S ::= <example> | <feature> | <class_feature>

<example> ::= ex(<example_position>, <list_attribute_values>).

<feature> ::= feature(<feature_position>, <feature_identifier>,
<feature_type>).

<class_feature> ::= classFeature(<feature_identifier>).

<feature_type> ::= nominal(<list_feature_type_values>) | numeric.

```

Figura 4.4: Gramática da Sintaxe Padrão em Prolog para Exemplos

<**arg-1**> posição desse atributo no conjunto de dados.

<**arg-2**> nome do atributo.

<**arg-3**> tipo do atributo. Os dois tipos básicos na sintaxe padrão em Prolog: `numeric` e `nominal`. O tipo de dados `numeric` abrange os tipos de dados inteiro e real da sintaxe padrão. O tipo de dados `nominal` abrange os tipo de dados `nominal` e `enumerated` da sintaxe padrão. Os tipos `date` e `time` da sintaxe padrão de exemplos não são suportados na sintaxe padrão em Prolog. Desta forma, o conversor entre as duas sintaxes é responsável por converter os atributos previamente dos tipos `date` e `time` em atributos do tipo `numeric`.

O argumento do predicado `classFeature/1` é simplesmente o nome da feature que corresponde à classe.

4.5 Sintaxe Padrão em Prolog de Regras

A sintaxe padrão em Prolog de regras por nós proposta é definida pela gramática mostrada na Figura 4.5.

Foi implementado um *script* Perl, denominado Conversor de Regras, que aceita como dado de entrada um arquivo contendo regras de classificação no formato *PBM* estendido e o converte para um arquivo de regras na sintaxe padrão em Prolog, segundo a gramática mostrada na Figura 4.5.

No $\mathcal{R}_{uleSystem}$, diversos conjuntos de regras podem ser armazenados na Base de Fatos simultaneamente, conforme dito anteriormente. Assim, é necessário identificar cada um desses conjuntos que provêm de diferentes experiências. Isso é realizado de forma automática pelo Conversor de Regras através do identificador `<id>::=id_X` utilizado como primeiro argumento nas cláusulas unitárias Prolog `inducer/2`, `inputFile/2`, `dAte/3`, `evaluatedAs/3`, `namesFile/2`, `dataFile/2` e `rule/6` na Figura 4.5. O valor de `X` é gerado como o valor da seguinte função hash:

$$X = (Sum1 + Sum2) \bmod 211 \quad (4.1)$$

onde

$$Sum1 = \langle \text{valor_month} \rangle + \langle \text{day} \rangle + \langle \text{year} \rangle + \langle \text{valor_week_day} \rangle.$$

e

$$Sum2 = \langle \text{hour} \rangle + \langle \text{minute} \rangle + \langle \text{second} \rangle.$$

Os valores de `<day>`, `<year>`, `<hour>`, `<minute>` e `<second>` são retirados da terceira linha do arquivo no formato *PBM* estendido que formam as informações que identificam quando foi realizada essa experiência. Já os valores de `<valor\_week\_day>` e `<valor\_month>`

```

S ::= <header> | <rule>

<header> ::= <inducer>
            <input_file>
            <date>
            <evaluated_as>
            <name_file>
            <data_file>

<inducer> ::= inducer(<id>,<inducer_name>).

<input_file> ::= inputFile(<id>,<input_file_name>).

<date> ::= dAte(<id>,day(<week_day>,<month>,<day>,<year>),
               time(<hour>:<minute>:<second>)).

<evaluated_as> ::= evaluatedAs(<id>,<evaluated_form>).

<name_file> ::= nameFile(<id>,<file>).

<data_file> ::= dataFile(<id>,<file>).

<rule> ::= rule(<id>,<rule number>,<list_of_conditions>,
               class(<class_feature>),
               <extension>).

<list_of_conditions> ::= [<condition>(<feature_name>,<feature_value>),
                        <condition>(<feature_name>,<feature_value>), ...].

<extension> ::= [<know values>],[<unknow values>]

<condition> ::= equal|notEqual|greater|less
              |lessORequal|greaterORequal|pertence.

<id> ::= id_X.

<week_day> ::= 'Sun'|'Mon'|'Thu'|'Wed'|'Wed'|'Fri'|'Sat'

<month> ::= 'Jan'|'Feb'|'Mar'|'Apr'|'May'|'Jun'
           '|Jul'|'Aug'|'Sep'|'Oct'|'Nov'|'Dec'

<evaluated_form> ::= 'UNORDERED'|'ORDERED'|'INTER-CLASS ORDERED'

```

Figura 4.5: Gramática da Sintaxe Padrão em Prolog para Regras

são primeiramente mapeados em números inteiros a partir dos valores dos símbolos terminais `<month>` e `<week_day>`, também retirados da terceira linha do arquivo no formato *PBM* estendido, como mostram as Tabelas 4.1 e 4.2, para finalmente calcular o valor de x (Equação 4.1).

<code><week_day></code>	<i>valor</i>
'Sun'	1
'Mon'	2
'Tue'	3
'Wed'	4
'Thu'	5
'Fri'	6
'Sat'	7

Tabela 4.1: Mapeamento dos Dias da Semana

<code><month></code>	<i>valor</i>	<code><month></code>	<i>valor</i>
'Jan'	1	'Jul'	7
'Feb'	2	'Aug'	8
'Mar'	3	'Sep'	9
'Apr'	4	'Oct'	10
'May'	5	'Nov'	11
'Jun'	6	'Dec'	12

Tabela 4.2: Mapeamento dos Meses do Ano

As informações contidas em cada um dos predicados produzidos pelo símbolo não-terminal `<header>` são listadas a seguir:

`inducer/2`: Informa o indutor utilizado para induzir as regras;

`inputFile/2`: Informa o arquivo que contém as regras no formato *PBM*;

`dAte/3`: Informa quando as experiências foram realizadas;

`evaluatedAs/2`: Informa a forma em que as regras devem ser avaliadas (Baranauskas & Monard 2000);

`nameFile/2`: Informa o arquivo `.names` utilizado; e

`dataFile/2`: Informa o arquivo `.data` utilizado.

O predicado `rule/6`, que representa as regras na sintaxe padrão em Prolog, possui os seguintes argumentos:

<arg-2> número da regra.

<arg-3> lista de condições da regra. No formato padrão *PBM*, o `<complex>` de uma regra é uma disjunção de conjunções de testes de atributos na forma:

`<atributo><operador><valor>`

o qual é convertido, na sintaxe padrão em Prolog, para:

`<condition>(<feature_name>,<feature_value>).`

onde `<atributo>` é mapeado para `<feature_name>`, `<operador>` é mapeado, conforme a Tabela 4.3, para `<condition>` e `<valor>` é mapeado para `<feature_value>`.

<arg-4> classe da regra.

<arg-5> lista de informações adicionais calculadas sobre exemplos que possuem valores conhecidos.

<arg-6> lista de informações adicionais calculadas sobre exemplos que possuem valores desconhecidos.

Para a regra *default*¹, os argumentos <arg-3>, <arg-5> e <arg-6> são sempre listas vazias.

<operador>	<condition>
=	equal
>=	greaterOREqual
<=	lessOREqual
>	greater
<	less
in	pertence
<>	notEqual

Tabela 4.3: Valores Nominais de Operadores

Conforme mencionado anteriormente, além do formato *PBM* estendido, o Conversor de Regras também suporta regras no formato *PBM* (Prati, Baranauskas, & Monard 2001b); neste caso, <arg-5> e <arg-6> são sempre listas vazias na sintaxe padrão em Prolog.

4.6 Considerações Finais

Neste capítulo foram descritos os conversores de sintaxe, que convertem conjuntos de regras e exemplos da sintaxe padrão *PBM* do projeto DISCOVER para a sintaxe padrão em Prolog do *RuleSystem*. No próximo capítulo, é apresentado em detalhes um dos principais módulos do *RuleSystem* — o Módulo de Combinação e Explicação, objeto de estudo deste trabalho. O Módulo de Análise de Regras está descrito em (Gomes 2002), como mencionado anteriormente.

¹Regra *default* é uma regra especial que possui o *Corpo* vazio e sua *Cabeça* é definida pela classe que possui a maior quantidade de exemplos no conjunto de exemplos de treinamento. Deve ser lembrado que todo classificador escrito no formato de regras *if-then* possui uma regra *default*; essa regra só é disparada quando nenhuma outra regra do classificador cobre o exemplo a ser classificado.

Módulo de Combinação e Explicação

Neste capítulo, é descrito um dos módulos principais do $\mathcal{R}_{uleSystem}$, o MCE — Figura 3.1 na página 20 — por nós implementado, responsável por:

1. 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 Módulo de Análise de Regras;
2. 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.

5.1 Procedimentos Principais

Para realizar as tarefas de responsabilidade do MCE, 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 instanciação¹ 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.

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

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 x ². O Algoritmo 1 descreve a forma de avaliação *ordered*.

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 vázio cobre o exemplo x , ou seja, não é considerada a regra *default*³. 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*.

¹Na 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.

²Observar que qualquer que seja a forma de avaliação, sempre existirá uma classificação para o exemplo x devido à existência da regra *default*

³Como mencionado anteriormente, 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 frequência no conjunto de exemplos utilizado para a construção dessa hipótese

Algoritmo 1 Avaliação *Ordered*

Require: $h = \{R_1, \dots, R_{NR}\};$

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
```

Algoritmo 2 Avaliação *Unordered*

Require: $h = \{R_1, \dots, R_{NR}\};$

x: exemplo a ser classificado.

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

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

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

Figura 5.1: Subconjuntos de Regras — Forma de Avaliação *Interclass*

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 5.1 mostra que, para cada classe, existe um conjunto de subconjuntos de regras⁴ $\mathbf{h}' = \{\mathbf{h}'_1, \dots, \mathbf{h}'_{NCl}\}$. 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, $\mathbf{h}'_1, \dots, \mathbf{h}'_{NCl}$, 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 $\mathbf{h}'_v \in \mathbf{h}'$, $v = 1, \dots, NCl$. Caso contrário, se todo o conjunto $\mathbf{h}'$ for varrido e nenhum elemento de $\mathbf{h}'$ satisfizer a condição $\textit{Rules} \neq \emptyset$, para algum $v = 1, \dots, NCl$, 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 $\mathbf{x}$.

⁴ $\mathbf{h}$ e $\mathbf{h}'$ diferem unicamente na regra *default*, ou seja, $\mathbf{h} = \mathbf{h}' + \{\textit{default}\}$. Assim, R_u de $\mathbf{h}$, que não aparece na Figura 5.1, é a regra *default*.

5.3 Determinação da Matriz de Confusão

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

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

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

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 $NCl \times NCl$. 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.

Algoritmo 3 Matriz de Confusão

Require: h : hipótese;

$S = \{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$: conjunto de exemplos;

$C = \{C_1, \dots, C_{NCl}\}$: 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:      $Classification := \text{classifyExample}(h, \mathbf{x}_i)$ ;
9:      $m := \text{index}(C, y_i)$ ;
10:     $n := \text{index}(C, Classification)$ ;
11:     $CM_{mn} := CM_{mn} + 1$ ;
12:   end for
13: return  $CM$ 

```

5.4 Explicação de Ensembles

Dado um conjunto de hipóteses $\mathcal{H} = \{h_1, \dots, h_L\}$, 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 $h_1, \dots, h_L$. 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 2.2 na página 7 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 x ;
- ? **<arg-3>** medida de avaliação de regra a ser usada para classificação (*majority* no caso de estar desinstanciado);
- **<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 h_l . Depois, se *Measure* estiver desinstanciado, considera 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 $h_l, l = 1, \dots, L$, 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 .

Algoritmo 4 Explicação de *Ensembles*

Require: $\mathcal{H} = h_l, l = 1, \dots, L$: conjunto de hipóteses;

(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( $\mathcal{H}, (x_i, y_i), Measure$ )
2:   Classes :=  $\emptyset$ ;
3:   Rules :=  $\emptyset$ ;
4:   for  $l := 1$  to  $L$  do
5:     classifyExample( $h_l, x, Class_l, HipRules_l$ );
6:     Classes := Classes  $\cup$  Classl;
7:     Rules := Rules  $\cup$  HipRulesl;
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

```

5.5 Combinação de Classificadores Simbólicos

Os dois procedimentos relacionados com combinação de classificadores são:

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

ambos descritos a seguir.

5.5.1 Algoritmo *RCO*

Dado um conjunto de hipóteses $\mathcal{H} = \{h_1, \dots, h_L\}$, um conjunto de exemplos de treinamento $S = \{(x_i, y_i), i = 1, \dots, N\}$ e uma medida de avaliação de regra *Measure*, o procedimento `rco/4` constrói uma nova hipótese como uma combinação das regras $R \in h_l, l = 1, \dots, L$, 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 3.3 na página 24. 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;
2. Calcula *Measure* para todas as regras $R \in h_l, l = 1, \dots, L$;
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 pára⁵;
5. Adiciona a regra *Rule* à nova hipótese;
6. Volta ao passo inicial.

⁵Se a regra *Rule* não cobre nenhum exemplo, isso significa que *Measure* atingiu seu mínimo (ou máximo) absoluto para todas as regras $R \in h_l, l = 1, \dots, L$ 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.

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 nos 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.

Algoritmo 5 *RCO*

Require: $\mathcal{H} = \mathbf{h}_l, l = 1, \dots, L$: conjunto de hipóteses;
 $S = \{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$: 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: $\mathbf{h}_{new} := \mathbf{h}_{new} \cup \{Rule\}$; {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: $\mathbf{h}_{new} := \mathbf{h}_{new} \cup \{defaultRule\}$; {adicioná-la a $\mathbf{h}_{new}$ }
- 21: **return** $\mathbf{h}_{new}$

O algoritmo *RCO* (*Rule Combination*) pode ser visualizado na Figura 5.2, onde se pode observar que os dados de entrada para o algoritmo *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. Deve ser observado que, para que o algoritmo funcione corretamente, o conjunto de dados de treinamento de ser diferente dos conjuntos de dados de 1 a L e, para que a medida de precisão ou erro do classificador não seja otimista, o conjunto de dados de teste deve ser independente do conjunto de dados de treinamento e do conjunto de dados de 1 a L . No entanto, se o número de exemplos for pequeno, a independência entre os conjuntos de dados pode não ser possível.

Figura 5.2: Ilustração do Algoritmo $\mathcal{RCO}$

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.

5.5.2 Algoritmo k-Fold Cross Validation

Dado um conjunto de hipóteses $\mathcal{H} = \{h_1, \dots, h_L\}$, um conjunto de treinamento $S = \{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$, 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 SD 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;
- + **<arg-3>** número de partições;
- + **<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 aproximada-

mente $\frac{N}{K}$ exemplos em cada partição. Para cada iteração $k = 1, \dots, K$, 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 h_k utilizando o procedimento que implementa o algoritmo $\mathcal{RCO}$ e o conjunto $S_{training}$;
3. Calcula-se a matriz de confusão para h_k utilizando o conjunto S_{test} ;
4. Calcula-se o erro cometido por 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

Require: $\mathcal{H} = \{h_1, \dots, h_L\}$;
 $S = \{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$: conjunto de exemplos;
 K : número de partições;
Measure: medida de avaliação de regra.

- 1: **procedure** kFoldCrossValidation($L, S, K, Measure$)
- 2: $S_{partic} = \text{cria_particoes}(K, S)$;
- 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: $h_k := \text{rco}(L, S_{training}, Measure)$;
- 8: $ConfusionMatrix_k := \text{confusionMatrix}(h_k, S_{test}, C_v)$;
- 9: $Erro_k := \text{calculo_erro}(ConfusionMatrix_k)$;
- 10: $Erro := Erro \cup \{Erro_k\}$
- 11: **end for**
- 12: $Mean = \text{mean}(Erro, K)$;
- 13: $SD = \text{sd}(Erro, Mean, K)$;
- 14: **return** $Mean, SD$

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

Algoritmo 7 Construção de Partições

Require: K : número de partições;
 $S = \{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$: 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: $\mathbf{x} := \text{retira_aleatorio}(S)$;
- 8: $S := S - \{\mathbf{x}\}$;
- 9: $N_{aux} = N_{aux} - 1$;
- 10: $S_k := S_k \cup \mathbf{x}$;
- 11: **end for**
- 12: **end for**
- 13: $S_1 = S$; {Colocar os elementos restantes de S em S_1 }
- 14: **return** $\{S_1, \dots, S_K\}$

5.6 Considerações Finais

Neste capítulo, foram descritos os procedimentos principais do Módulo de Combinação e Explicação do $\mathcal{R}_{ule}S_{system}$ e os algoritmos implementados. Dentre esses algoritmos, destaca-se o $\mathcal{RCO}$, responsável pela geração de um novo classificador através da combinação de classificadores induzidos por algoritmos de AM simbólicos. No próximo capítulo, é mostrado um exemplo de utilização do Módulo de Combinação e Explicação, utilizando um conjunto de dados artificiais.

Um Exemplo de Aplicação do MCE

*P*ara melhor ilustrar os procedimentos do Módulo de Combinação e Explicação, neste capítulo são descritos alguns experimentos realizados sobre um conjunto de exemplos artificial, ou seja, um conjunto de exemplos gerados a partir de uma função verdadeira f . Deve ser ressaltado que os resultados desses experimentos permitiram melhorar o critério de seleção de regras inicialmente implementado no $\mathcal{RCO}$. Esses experimentos encontram-se descritos com maiores detalhes em (Bernardini & Monard 2002b).

6.1 Descrição do Conjunto de Exemplos Artificial

O conjunto de exemplos utilizado corresponde a função verdadeira f , ou hipótese verdadeira, definida por

$$f(att_1, att_2) = \begin{cases} o & \text{se } 10 \leq att_1 \leq 20, 40 \leq att_2 \leq 50, \\ o & \text{se } 20 \leq att_1 \leq 30, 20 \leq att_2 \leq 30, \\ o & \text{se } 40 \leq att_1 \leq 50, 30 \leq att_2 \leq 40, \\ x & \text{se } 10 \leq att_1 \leq 20, 20 \leq att_2 \leq 30, \\ x & \text{se } 30 \leq att_1 \leq 40, 10 \leq att_2 \leq 20, \\ x & \text{se } 50 \leq att_1 \leq 60, 30 \leq att_2 \leq 40. \end{cases} \quad (6.1)$$

para $att_1, att_2 \in \mathbb{R}$, ilustrada na Figura 6.1. Os retângulos contornados por uma linha tracejada correspondem à classe o , e os retângulos contornados por uma linha contínua correspondem à classe x . Essa função verdadeira f foi gerada pensando nas regiões retangulares formadas pelas regras induzidas sobre um domínio com dois atributos e uma classe, as quais pode ser facilmente visualizáveis no espaço bidimensional.

Dada a verdadeira função f , infinitos conjuntos de exemplos podem ser gerados. Para

Figura 6.1: Função Verdadeira f

realizar os experimentos aqui apresentados, foram gerados 8 conjuntos de exemplos diferentes, denominados $S_1, \dots, S_8$. Cada um desses conjuntos possui 120 exemplos, gerados aleatoriamente, onde o número de exemplos em cada classe é balanceado, ou seja, 50% dos exemplos pertencem à classe o e 50% dos exemplos pertencem à classe x . Os sete primeiros conjuntos $S_1, \dots, S_7$ foram utilizados para induzir 7 (sete) hipóteses, denominadas $h_1, \dots, h_7$, com dois algoritmos de aprendizado de máquina: o $\mathcal{CN}2$ e o $See5$. O $\mathcal{CN}2$ (Clark & Niblett 1989) induz hipóteses com regras não-ordenadas e ordenadas (*unordered* e *ordered*), e o $See5$ (Quinlan 2002) induz árvores de decisão e regras não-ordenadas. As sete hipóteses $h_1, \dots, h_7$ foram induzidas como regras não-ordenadas, sendo que $h_1, \dots, h_5$ foram induzidas pelo algoritmo $\mathcal{CN}2$ e h_6 e h_7 foram induzidas pelo algoritmo $See5$, como mostra a Figura 6.2.

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 h composta por NR regras tem suas regras plotadas em ordem inversa, 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

Figura 6.2: Hipóteses Induzidas por $\mathcal{CN}2$ e *See5* Utilizadas nos Experimentos

hipóteses cobrem os exemplos do domínio.

Assim, a forma como as hipóteses induzidas pelo $\mathcal{CN}2$ — $h_1, \dots, h_5$ — se comportam em relação à função verdadeira f pode ser vista na Figura 6.3; a forma como as hipóteses induzidas pelo *See5* — h_6 e h_7 — se comportam em relação à função verdadeira f pode ser vista na Figura 6.4.

Utilizando as hipóteses induzidas $h_1, \dots, h_7$ 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 2.2 na página 7. Essas hipóteses são denominadas h_{Exp_accR} , h_{Exp_covR} , h_{Exp_satR} , h_{Exp_sensR} , h_{Exp_specR} e h_{Exp_supR} , onde *Exp* referencia a experiência em que a hipótese foi construída.

São vários os objetivos desses experimentos, dentre eles:

- Verificar qual o erro aparente¹ 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} ;

¹Erro aparente é o erro cometido por uma hipótese sobre os exemplos utilizados para induzir esta hipótese.

Figura 6.3: Representação Gráfica das Hipóteses Induzidas por $\mathcal{CN}^2$.

Figura 6.4: Representação Gráfica das Hipóteses Induzidas por See5.

- Verificar as diferenças entre os resultados obtidos pelas hipóteses construídas pelo $\mathcal{RCO}$ com as hipóteses induzidas por $\mathcal{CN2}$ e See5 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 $S_1, \dots, S_8$ gerados anteriormente. Para realizar as comparações, os experimentos podem ser divididos em duas fases:

Fase 1: Verificar as hipóteses induzidas por $\mathcal{CN2}$ e See5 utilizando todos os exemplos ($S_1 \cup \dots \cup S_8$) em cada experimento.

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

A seguir, é feita uma descrição dos três experimentos.

6.2 Descrição dos Experimentos Realizados

As duas fases dos experimentos 1, 2 e 3 podem ser visualizadas nas Figuras 6.5, 6.6 e 6.7. Deve ser observado que as hipóteses $h_1, \dots, h_7$ foram induzidas uma única vez, e se aparecem nessas figuras como induzidas uma ou mais vezes, o único motivo é para facilitar a visualização.

Na fase 1 do experimento 1, foram induzidas 2(duas) hipóteses, uma com o $\mathcal{CN2}$ e outra com o See5, utilizando como conjunto de exemplos a união dos conjuntos S_1, S_2, S_3 e S_8 — Figura 6.5. 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

Figura 6.5: Ilustração do Experimento 1.

Figura 6.6: Ilustração do Experimento 2.

Figura 6.7: Ilustração do Experimento 3.

mais precisa nesta primeira fase do experimento. Analogamente, na fase 1 do experimento 2, foram induzidas 2(duas) hipóteses, uma com o $\mathcal{CN}2$ e outra com o $\mathcal{See}5$, mas, como conjunto de exemplos, foi utilizada a união dos conjuntos $S_1, \dots, S_5$ e S_8 — Figura 6.6. De forma semelhante, na fase 1 do experimento 3, foi utilizado como conjunto de exemplos a união dos conjuntos $S_1, \dots, S_8$ para a indução das duas hipóteses utilizando o $\mathcal{CN}2$ e o $\mathcal{See}5$ — Figura 6.7. Os resultados obtidos são mostrados na Tabela 6.1.

Experimento	Algoritmos	Subconjunto/ # Ex. Treinamento	Erro Aparente	$e \pm SD$	# de Regras Induzidas
1	$\mathcal{CN}2$	$S_1 \cup S_2 \cup S_3 \cup S_8 / 480$	1.40%	$4.46\% \pm 0.85\%$	15
	$\mathcal{See}5$	$S_1 \cup S_2 \cup S_3 \cup S_8 / 480$	2.20%	$3.30\% \pm 0.90\%$	7
2	$\mathcal{CN}2$	$S_1 \cup \dots \cup S_5 \cup S_8 / 720$	2.00%	$2.99\% \pm 0.60\%$	15
	$\mathcal{See}5$	$S_1 \cup \dots \cup S_5 \cup S_8 / 720$	2.00%	$3.00\% \pm 1.20\%$	7
3	$\mathcal{CN}2$	$S_1 \cup \dots \cup S_8 / 960$	1.40%	$2.76\% \pm 0.44\%$	15
	$\mathcal{See}5$	$S_1 \cup \dots \cup S_8 / 960$	2.10%	$3.00\% \pm 0.40\%$	8

Tabela 6.1: Resultados Obtidos Utilizando $\mathcal{CN}2$ e $\mathcal{See}5$ na Fase 1 dos Experimentos 1, 2 e 3

Para a realização da fase 2 dos experimentos 1, 2 e 3, foram induzidas as hipóteses $h_1, \dots, h_7$, utilizando como conjuntos de exemplos os conjuntos $S_1, \dots, S_7$ respectivamente, conforme dito anteriormente. 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 6.2.

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

Tabela 6.2: Resultados Obtidos nas Hipóteses de Entrada $h_1, \dots, h_7$

Na fase 2 do experimento 1, foram construídas as hipóteses h_{1_accR} , h_{1_covR} , h_{1_satR} , h_{1_sensR} , h_{1_specR} e h_{1_supR} . Para isso, foram utilizadas as hipóteses h_1 , h_2 e h_3 e o conjunto de exemplos S_8 — Figura 6.5. A Tabela 6.3 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} .

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
h_{1_accR}	0.00%	6.00%	12	03 – 02.50%
h_{1_covR}	0.00%	6.00%	3	60 – 50.00%
h_{1_satR}	0.00%	6.00%	12	03 – 02.50%
h_{1_sensR}	0.00%	6.00%	3	60 – 50.00%
h_{1_specR}	10.00%	7.50%	6	57 – 47.50%
h_{1_supR}	0.00%	6.00%	3	60 – 50.00%

Tabela 6.3: Resultados Obtidos Utilizando $\mathcal{RCO}$ — Fase 2 do Experimento 1

Analogamente, na fase 2 do experimento 2, foram construídas as hipóteses h_{2_accR} , h_{2_covR} , h_{2_satR} , h_{2_sensR} , h_{2_specR} e h_{2_supR} . Para isso, foram utilizadas as hipóteses $h_1, \dots, h_5$ e o conjunto de exemplos S_8 — Figura 6.6. Os resultados obtidos para as hipóteses h_{2_accR} , h_{2_covR} , h_{2_satR} , h_{2_sensR} , h_{2_specR} e h_{2_supR} são idênticos aos obtidos na fase 2 do experimento 1 — Tabela 6.3. Ou seja, foram construídas as mesmas hipóteses que no experimento anterior. Em outras palavras, as regras que constituem as hipóteses h_4 e h_5 nunca foram consideradas melhores por $\mathcal{RCO}$.

Apesar das regras que constituem as hipóteses h_4 e h_5 induzidas por $\mathcal{CN}2$ não terem influência nas hipóteses construídas por $\mathcal{RCO}$, essas duas hipóteses foram mantidas no experimento 3 pois os exemplos utilizados para induzi-las são também utilizados por $\mathcal{CN}2$ e *See5* na fase 1 do experimento, influenciando no cálculo do erro e desvio padrão dos classificadores induzidos por esses dois algoritmos de AM. Assim, foram construídas as hipóteses h_{3_accR} , h_{3_covR} , h_{3_satR} , h_{3_sensR} , h_{3_specR} e h_{3_supR} . Para isso, foram utilizadas as hipóteses $h_1, \dots, h_7$ e o conjunto de exemplos S_8 — Figura 6.7. Os resultados obtidos são mostrados na Tabela 6.4. Observando os conjuntos de regras construídos pelo $\mathcal{RCO}$ nesse experimento, apenas a hipótese h_{3_covR} apresentou uma pequena diferença em relação a h_{1_covR} e h_{2_covR} . Todas as outras hipóteses são idênticas às hipóteses construídas nos experimentos anteriores.

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
h_{3_accR}	0.00%	6.00%	12	03 – 02.50%
h_{3_covR}	0.00%	9.00%	3	60 – 50.00%
h_{3_satR}	0.00%	6.00%	12	03 – 02.50%
h_{3_sensR}	0.00%	6.00%	3	60 – 50.00%
h_{3_specR}	10.00%	7.50%	6	57 – 47.50%
h_{3_supR}	0.00%	6.00%	3	60 – 50.00%

Tabela 6.4: Resultados Obtidos Utilizando $\mathcal{RCO}$ — Fase 2 do 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 construídas 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 $h_1, h_2, \dots, h_L$, 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 $R_u \in h_i$ e $R_q \in h_j$, $i < j$, ambas com a mesma melhor medida segundo o critério implementado, o $\mathcal{RCO}$ escolhe a regra R_u , pois $i < j$. Em outras palavras, se a hipótese h_i precede a hipótese h_j então, em caso de empate, as regras que constituem h_i têm precedência sobre as regras de h_j ;
2. Supondo que existam, em uma iteração do $\mathcal{RCO}$, duas (ou mais) regras R_u e R_q nessa ordem, todas com a mesma melhor medida segundo o critério implementado, então, independente de pertencer à mesma hipóteses ou a hipóteses diferentes 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 $\mathcal{RCO}$ construir a mesma hipótese quando é utilizado o conjunto de hipóteses $\{h_1, h_2, h_3\}$ ou o conjunto $\{h_1, \dots, h_5\}$. Neste caso específico, nenhuma das regras das hipóteses h_4 e h_5 foi considerada por $\mathcal{RCO}$.

Uma forma de diminuir esse *bias*² é escolher aleatoriamente a melhor regra a ser utilizada na hipótese construída pelo $\mathcal{RCO}$. 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.

²O *bias* de um algoritmo de aprendizado de máquina mede o quanto, na média, as hipóteses induzidas pelo algoritmo de aprendizado sobre todos os possíveis conjuntos de treinamento de um dado tamanho se aproxima da hipótese verdadeira.

Com o $\mathcal{RCO}$ revisado, o experimento 3 foi repetido 3(três) vezes, com as mesmas hipóteses de entrada $h_1, \dots, h_7$ e o mesmo conjunto de exemplos S_8 , com o intuito de verificar se as hipóteses construídas são distintas e melhores que as anteriores.

Analogamente à fase 2 do experimento 3, para as hipóteses h_{4_accR} , h_{4_covR} , h_{4_satR} , h_{4_sensR} , h_{4_specR} e h_{4_supR} construídas, a Tabela 6.5 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 h_{5_accR} , h_{5_covR} , h_{5_satR} , h_{5_sensR} , h_{5_specR} e h_{5_supR} são mostrados na Tabela 6.6, e para as hipóteses h_{6_accR} , h_{6_covR} , h_{6_satR} , h_{6_sensR} , h_{6_specR} e h_{6_supR} , são mostrados na Tabela 6.7.

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
h_{4_accR}	0.00%	6.00%	6	20 – 16.67%
h_{4_covR}	3.33%	9.00%	3	60 – 50.00%
h_{4_satR}	0.83%	5.50%	7	05 – 4.17%
h_{4_sensR}	3.33%	9.00%	3	60 – 50.00%
h_{4_specR}	0.83%	6.00%	7	23 – 19.17%
h_{4_supR}	0.00%	6.00%	3	60 – 50.00%

Tabela 6.5: Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 4

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
h_{5_accR}	0.00%	4.50%	8	20 – 16.67%
h_{5_covR}	3.33%	9.00%	3	60 – 50.00%
h_{5_satR}	0.00%	6.00%	11	04 – 33.33%
h_{5_sensR}	3.33%	9.00%	3	60 – 50.00%
h_{5_specR}	0.83%	3.00%	9	25 – 20.83%
h_{5_supR}	3.33%	9.00%	3	60 – 50.00%

Tabela 6.6: Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 5

Classificadores	Erro em S_8	Erro em S_{teste}	# de Regras Induzidas	# Exemplos Não Cobertos
h_{6_accR}	0.00%	6.00%	7	24 – 20.00%
h_{6_covR}	3.33%	9.00%	3	60 – 50.00%
h_{6_satR}	0.00%	4.50%	8	20 – 16.67%
h_{6_sensR}	3.33%	9.00%	3	60 – 50.00%
h_{6_specR}	0.00%	6.00%	7	20 – 16.67%
h_{6_supR}	3.33%	9.00%	3	60 – 50.00%

Tabela 6.7: Resultados Obtidos Utilizando $\mathcal{RCO}$ — Experimento 6

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

- Com os critérios de avaliação de regra *covR* e *sensR*, foram induzidas hipóteses idênticas nos 3 experimentos. Entretanto, foi observado que há uma diferença na ordem em que as regras foram selecionadas.

- 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 6.5, 6.6 e 6.7 — alguns dados apresentados podem ser iguais, mas foi observado que 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.

6.3 Análise dos Resultados

Na Tabela 6.1 na página 57, observa-se que, conforme esperado, o erro e o desvio padrão de $\mathcal{CN}2$ e See5 tendem a se estabilizar quanto maior é o número de exemplos disponíveis para indução de hipóteses. Também confirma que $\mathcal{CN}2$ induz mais regras que See5 .

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 6.5, 6.6 e 6.7 —, 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 See5 na fase 1 do experimento 3, é possível observar que, como esperado, as hipóteses induzidas por $\mathcal{CN}2$ e See5 são melhores que as construídas 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 6.8 —, 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 $h_1, \dots, h_7$ 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, com algumas semelhanças entre as hipóteses h_1 e h_3 e as hipóteses h_2 e h_5 . Quanto às hipóteses h_6 e h_7 induzidas pelo See5 , notou-se uma grande semelhança entre elas. 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. Maiores detalhes sobre os experimentos realizados são encontrados em (Bernardini & Monard 2002b).

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 6.8: Número de Regras Seleccionadas (com a Regra *Default*) nos Experimentos 4, 5 e 6

6.4 Considerações Finais

Neste capítulo, foram relatados os experimentos realizados utilizando o Módulo de Combinação e Explicação e um conjunto de dados artificiais. Esses experimentos mostraram a utilidade de se usar dados artificiais, pois os experimentos são bastante controlados. Com isso, foi possível notar a importância de se reconsiderar casos de empate na seleção de regras, o que talvez não fosse possível com outros conjuntos de dados. Assim, o *RCO* foi melhorado. Porém, ainda devido à utilização de um conjunto de dados artificiais, outras possíveis melhorias podem ser realizadas, as quais serão relatadas nas conclusões deste trabalho, como proposta de trabalhos futuros. No próximo capítulo, é feito um estudo de caso utilizando o MCE e um conjunto de dados do mundo real.

Estudo de Caso — Processamento de Sêmen Diagnóstico

Neste capítulo é descrito um estudo de caso do mundo real utilizando o Módulo de Combinação e Explicação. O estudo de conjuntos de dados reais envolve uma série de problemas que não são enfrentados quando os experimentos são realizados utilizando conjuntos de dados obtidos de repositórios como o Repositório de Dados da UCI (Blake, Keogh, & Merz 1998) ou conjuntos de dados artificiais, como o utilizado no capítulo anterior. Esses problemas envolvem o trabalho de coleta, preparação e limpeza dos dados, trabalhos quais realizados por Huei Diana Lee, ex-integrante e atual colaboradora do LABIC.

7.1 O Processamento de Sêmen Diagnóstico

Na medicina, freqüentemente existem exames e processos importantes, mas apresentam um alto custo para sua realização. Em muitos casos, é desejável que os mesmos resultados, ou alguns resultados parciais, possam ser obtidos por métodos menos custosos. Um exemplo desse caso é o processamento de sêmen.

O estudo de caso sobre processamento de sêmen diagnóstico teve seu início em (Lee 2000), onde é descrita a importância desse processamento no tratamento para a reprodução assistida. A explicação a seguir está baseada nos trabalhos de Huei Diana Lee (Lee & Monard 2000b; Lee & Monard 2000a; Esteves & Bento 1998; Esteves, Sharma, Thomas, & Agarwal 2000; Esteves, Lee, Monard, & Lopes 2001) e foi desenvolvida conjuntamente com Alan Keller Gomes (Gomes & Monard 2002a). Esse exame permite:

1. Quantificar com precisão a qualidade do sêmen (processamento de sêmen diagnóstico);
2. Recuperar a maior quantidade possível de espermatozóides para a utilização na re-

produção assistida (processamento de sêmen terapêutico).

A quantidade de espermatozóides recuperados pelo processamento de sêmen influencia na escolha da técnica que será utilizada no tratamento. São utilizadas três técnicas no tratamento para a reprodução assistida:

1. Inseminação Intra Uterina — IUI;
2. Fertilização In Vitro — FIV e
3. Injeção Intracitoplasmática do Espermatozóide no Oócito — ICSI.

Deve ser observado que o processamento de sêmen é bastante custoso. A realização do processamento de sêmen pode elevar o custo do exame em aproximadamente 80% do valor de um espermograma. Essa elevação de custo se deve principalmente a três fatores: necessidade de equipamentos especiais, mão de obra qualificada e tempo gasto para a realização do exame. Assim, um dos interesses do estudo desse tema é tentar prever qual será a quantidade de espermatozóides recuperados pelo processamento de sêmen antes mesmo da realização desse exame, a partir de exames menos custosos, como o espermograma. Dessa forma, dependendo da qualidade da predição, o especialista poderia decidir por uma técnica sem a necessidade da realização do processamento de sêmen. Um outro interesse para o estudo desse tema é a extração e avaliação do conhecimento adquirido.

O processamento de sêmen, através do fornecimento de melhores condições, permite que o maior número de espermatozóides móveis (motilidade) seja recuperado. Os espermatozóides são classificados em graus A, B, C e D dependendo de sua motilidade:

- grau A: espermatozóides que apresentam o maior grau de motilidade;
- grau B: espermatozóides que apresentam um menor grau de motilidade;
- grau C: espermatozóides que se movem em círculos;
- grau D: espermatozóides que são imóveis.

Os graus de motilidade são atributos medidos tanto no espermograma (exame de baixo custo) quanto no processamento de sêmen (exame de alto custo). Os atributos medidos durante o processamento de sêmen são utilizados para a determinação das classes e os atributos medidos através do espermograma são fornecidos como atributos ao algoritmo de indução. Em outras palavras, os valores dos atributos de cada exemplo (paciente) são medidos utilizando um exame pouco custoso (espermograma, neste caso) enquanto que as classes desses exemplos são determinadas através de um outro exame mais custoso (processamento de sêmen, neste trabalho). A idéia é tentar verificar se é possível descobrir um relacionamento entre eles tal que, pelo menos para novos pacientes, a necessidade de realizar o exame mais custoso seja minimizada. Como já mencionado, este é um procedimento freqüentemente utilizado na área de medicina a fim de tentar diminuir, mas com segurança, o custo de um tratamento.

O especialista sugeriu duas possíveis opções para os valores considerados na classificação dos exemplos:

- A: considerar apenas a percentagem de espermatozóides classificados como de motilidade grau A no processamento de sêmen ou
- AB: considerar a percentagem de espermatozóides classificados como de motilidade grau A e grau B no processamento de sêmen.

Assim, foram definidas três classes, baseadas nas técnicas que são utilizadas no tratamento para a reprodução assistida:

- 1: $x < 1$ — ICSI;
- 2: $1 \leq x < 5$ — FIV e
- 3: $x \geq 5$ — IUI.

onde x representa milhões de espermatozóides por mililitro (ml).

Os casos para a composição desses conjunto de dados foram extraídos de casos reais do Centro de Referência em Infertilidade Masculina — Androfert¹ —, em Campinas, SP, fornecidos pelo Dr. Sandro C. Esteves. Em (Lee 2000), foram calculados para cada caso os valores A e AB, gerando dois conjuntos de dados, *Proc-a* e *Proc-ab*, respectivamente, cada um com 231 exemplos. Diversos experimentos foram conduzidos considerando-se esses dois conjuntos separadamente. Com o decorrer do trabalho, foi necessário criar novos atributos como combinação de atributos originais, criando, assim, diferentes conjuntos de dados (Lee & Monard 2000b). Um desses conjuntos de dados que mostrou um bom resultado com relação à precisão dos classificadores induzidos foi o denominado *Proc-a-gmg-d*, o qual considera somente os exemplos em que o processamento de sêmen é diagnóstico.

Posteriormente, ao conjunto de exemplos *Proc-a* do trabalho inicial foram acrescentados novos casos. Tomando desse conjunto de exemplos maior somente os exemplos em que o processamento de sêmen é diagnóstico, foi obtido um segundo conjunto de exemplos *Proc-a-gmg-d*, semelhante ao construído com os exemplos *Proc-a* do trabalho inicial, mas incluindo os novos casos, o qual é utilizado neste trabalho. Deve ser ressaltado que todo o trabalho de coleta, preparação e limpeza dos dados foi realizada por Huei Diana Lee.

7.2 Descrição do Conjunto de Dados

A Tabela 7.1 sumariza as características do conjunto de dados utilizado neste estudo. Ela mostra o número de exemplos (#Exemplos), número e porcentagem de exemplos com valores duplicados (que aparecem mais de uma vez) ou conflitantes (que possuem o mesmo atributo-valor mas têm diferentes classificações), número de atributos (# Atributos) contínuos e nominais, distribuição de classes, o erro majoritário e se o conjunto de dados tem ao menos um valor desconhecido². Já a Tabela 7.2 descreve os atributos, exceto o atributo classe, do conjunto de dados utilizado — *Proc-a-gmg-d*.

¹<http://www.androfert.com.br/>

²Estas informações foram obtidas utilizando a ferramenta *info* da *MCC++*.

# Exemplos	# Duplicados ou conflitantes (%)	# Atributos (cont.,nom.)	Classe	Classe %	Erro Majoritário	Valores Desconhecidos
240	1 (0.42%)	21 (10,11)	1	42.92%	57.08%	S
			2	22.08%	no valor 1	
			3	35.00%		

Tabela 7.1: Sumário das Características do Conjunto de Dados *Proc-a-gmg-d*

Número da Feature	Nome da Feature	Descrição	#Valores Distintos		
			possíveis	atuais	tipo
#0	idade	Idade do paciente	—	26	contínuo
#1	hora	Período da coleta da amostra de sêmen 1: manhã; 2: tarde	2	3	discreto
#2	processamento	Processamento do sêmen realizado após esta quantidade de minutos após a coleta	—	12	contínuo
#3	tempo-abs	Tempo de abstinência	—	12	contínuo
#4	volume	Volume de sêmen coletado	—	55	contínuo
#5	cor	Cor do sêmen coletado 1: branco-opalescente (normal); 2: amarelo-opalescente; 3: translúcido	3	4	discreto
#6	odor	Odor do sêmen coletado 1: característico; 2: forte; 3: urina	3	3	discreto
#7	pH	pH do sêmen coletado	—	13	contínuo
#8	viscosidade	Viscosidade do sêmen coletado 1: normal; 2: aumentada	2	2	discreto
#9	liquefacao	Liquefação do sêmen coletado 1: completa; 2: incompleta	2	2	discreto
#10	concentracao	Concentração de espermatozóides por ml coletado	—	198	contínuo
#11	concentracao-total	Concentração total de espermatozóides	—	219	contínuo
const #12	motilidade	% de espermatozóides móveis	—	74	contínuo
#13	class-A	Classificação da motilidade - Grau A	—	66	contínuo
#14	class-B	Grau B	—	53	contínuo
#15	class-C	Grau C	—	44	contínuo
#16	class-D	Grau D	—	76	contínuo
#17	vitalidade	% de espermatozóides vivos	—	59	contínuo
#18	num-leu	Número de leucócitos	—	66	contínuo
#19	Kruger	Morfologia Estrita de Kruger - % de formas normais	—	28	contínuo
#20	HP	Teste Hipo-osmótico - % de inchados	—	54	contínuo

Tabela 7.2: Descrição dos Atributos do Conjunto de Dados *Proc-a-gmg-d*

7.3 Descrição do Experimento

O experimento conduzido tem por objetivo gerar um novo classificador simbólico utilizando vários classificadores simbólicos previamente induzidos pelo algoritmo de aprendizado de máquina $\mathcal{CN}2$, através de diferentes critérios implementados pelo $\mathcal{RCO}$ para eleger as regras que farão parte desse novo classificador. Logo após, verificar o erro cometido pelo classificador assim construído com os erros cometidos pelos classificadores individuais, bem como verificar o conjunto de regras que constituem esse novo classificador. Uma ilustração do experimento realizado pode ser visualizada na Figura 7.1.

Inicialmente, foi induzido um único classificador com o $\mathcal{CN}2$, utilizando todo o conjunto de exemplos S disponível, ou seja, os 240 exemplos, e foi medido o erro aparente desse classificador. Após, foi utilizada a técnica de 10-fold cross validation para estimar o erro

Figura 7.1: Ilustração do Experimento Realizado.

verdadeiro e o desvio padrão desse classificador, bem como o erro de classificação em cada classe.

Num passo seguinte, o conjunto total de exemplos S , com 240 exemplos, foi dividido aleatoriamente em quatro subconjuntos disjuntos S_1 , S_2 , S_3 e S_4 de exemplos com o mesmo número de exemplos em cada um deles — 60 exemplos.

Utilizando cada um dos subconjuntos de exemplos S_1 , S_2 e S_3 , foram induzidos três classificadores h_1 , h_2 e h_3 , respectivamente, com o indutor $\mathcal{CN}2$. Para esses três classificadores foi também calculado o erro aparente, a estimativa do erro verdadeiro e desvio padrão, bem como a estimativa do erro e desvio padrão em cada classe através de 10-fold cross validation. Os resultados obtidos são encontrados na Tabela 7.3.

Com as hipóteses induzidas h_1 , h_2 , h_3 e o conjunto de exemplos S_4 , foram construídos seis classificadores utilizando o algoritmo $\mathcal{RCO}$ — Capítulo 5. Cada um desses seis classificadores foi construído utilizando um critério (medida) diferente de avaliação de regra para eleger a melhor regra em cada iteração. As medidas de avaliação de regra utilizadas neste experimento são as mesmas medidas utilizadas nos experimentos com o conjunto de dados artificiais, no capítulo anterior, e constantes na Tabela 2.2 na página 7. As hipóteses construídas pelo $\mathcal{RCO}$ neste experimento são denominadas h_{accR} , h_{covR} , h_{satR} , h_{sensR} , h_{specR} e h_{supR} .

Finalmente, foi calculado o erro para cada um dos seis classificadores construídos por $\mathcal{RCO}$ utilizando os subconjuntos de exemplos S_1 , S_2 e S_3 . Ainda que neste experimento os

Conj. de Dados → Classificador	Erro Aparente	Erro Majoritário	Erro nas Classes ($e \pm SD$ (%))	Erro do Classificador ($e \pm SD$ (%))	# Regras Induzidas
$S \rightarrow h_{CN2}$	12.90%	57.08%	1 – 14.96±3.59 2 – 82.58±5.65 3 – 44.93±5.74	40.83±2.83	33
$S_1 \rightarrow h_1$	3.30%	57.08%	1 – 33.33±10.54 2 – 80.00±13.33 3 – 20.67±7.11	41.66±5.13	12
$S_2 \rightarrow h_2$	1.70%	57.08%	1 – 8.50±4.60 2 – 60.00±16.33 3 – 40.83±12.94	36.66±7.78	13
$S_3 \rightarrow h_3$	0.00%	57.08%	1 – 20.83±7.38 2 – 80.00±13.33 3 – 18.33±8.03	41.68±5.69	16

Tabela 7.3: Resultados Obtidos Utilizando o $CN2$

exemplos contidos em S_1 , S_2 e S_3 não foram utilizados por RCO para construir os classificadores, eles foram utilizados para induzir as hipóteses h_1 , h_2 e h_3 , utilizadas pelo RCO . O ideal seria estimar o erro cometido pelos classificadores construídos por RCO em um conjunto de exemplos independente. Neste trabalho utilizamos a primeira opção devido ao número limitado de exemplos disponíveis. Assim, deve ser ressaltado que o erro por nós estimado utilizando os exemplos contidos nos subconjuntos S_1 , S_2 e S_3 tende a ser otimista. Foi medido o erro aparente com o subconjunto de exemplos S_4 .

Deve ser lembrado que, para gerar um novo classificador, o algoritmo RCO pára em duas condições durante a iteração que executa, como mostra o Algoritmo 5 na página 46:

1. Não há mais exemplos a serem cobertos;
2. As regras restantes atingiram seu mínimo (ou máximo) absoluto, relacionado à medida sendo considerada.

Na Tabela 7.4, são mostrados os resultados obtidos, onde a última coluna mostra o número de exemplos restantes na construção do novo classificador. Ou seja, em todos os casos o segundo critério de parada foi utilizado pelo algoritmo RCO . Isso significa que esses exemplos foram classificados pela regra *default* da hipótese construída, a qual prediz a classe 1.

7.4 Análise dos Resultados

Quanto ao número de regras selecionadas pelo RCO para a construção das hipóteses, pode ser observado que esse número é geralmente menor que o número de regras que constituem cada um dos classificadores h_1 , h_2 e h_3 . Entretanto, este resultado é esperado já que, para este conjunto de exemplos, o segundo critério de parada foi sempre utilizado por RCO .

Com relação ao erro dos classificadores construídos por RCO (erro aparente em S_4 e erro do classificador em $S_1 \cup S_2 \cup S_3$ na Tabela 7.4), todos eles são muito mais altos que

Classificador	Erro Aparente em S_4	Erro Majoritário	Erro nas Classes em $S_1 \cup S_2 \cup S_3$	Erro do Classificador em $S_1 \cup S_2 \cup S_3$	# Regras Selecionadas	# Exemplos Não Cobertos
h_{accR}	36.67%	57.08%	1 – 1.30% 2 – 100.00% 3 – 31.82%	32.78%	12	26 – 43.33%
h_{covR}	36.67%	57.08%	1 – 7.79% 2 – 100.00% 3 – 33.33%	36.11%	7	21 – 35.00%
h_{satR}	33.33%	57.08%	1 – 3.90% 2 – 97.30% 3 – 27.27%	31.67%	14	21 – 35.00%
h_{sensR}	38.33%	57.08%	1 – 2.60% 2 – 100.00% 3 – 45.46%	38.33%	6	24 – 40.00%
h_{specR}	53.33%	57.08%	1 – 0.00% 2 – 100.00% 3 – 95.45%	55.56%	5	49 – 81.66%
h_{supR}	38.33%	57.08%	1 – 0.00% 2 – 100.00% 3 – 51.52%	40.56%	6	25 – 41.67%

Tabela 7.4: Resultados Obtidos Utilizando o RCO

o erro das hipóteses induzidas por $\mathcal{CN}2$ — Tabela 7.3. Ainda que o erro em $S_1 \cup S_2 \cup S_3$ seja semelhante ao erro obtido pelas hipóteses induzidas por $\mathcal{CN}2$ utilizando 10-fold cross validation, essa comparação não é válida, pois o erro em $S_1 \cup S_2 \cup S_3$ é muito otimista, enquanto que o erro obtido utilizando 10-fold cross validation é uma boa estimativa de erro verdadeiro. Porém, deve ser observado que o objetivo de utilizar RCO com as medidas consideradas não é, necessariamente, o de construir um bom classificador no sentido de classificar relativamente bem qualquer exemplo. Na realidade, o classificador construído por RCO é aquele que possui regras selecionadas segundo um critério (medida) utilizado. Dessa forma, é interessante observar qual o comportamento dessas melhores regras.

Para verificar o comportamento das regras que constituem as hipóteses construídas pelo RCO , devem ser analisados o número de regras de cada hipótese e os exemplos cobertos pela regra *default*. Esses dados são mostrados nas Tabelas 7.5 e 7.6. Na Tabela 7.5 consta, para cada hipótese construída pelo RCO , o número de exemplos do conjunto S_4 classificados pela regra *default*, e na Tabela 7.6 consta o número de regras selecionadas por RCO para cada classe do domínio. Como pode ser observado, a classe mais crítica é a classe 2. O conjunto de exemplos S_4 possui 16 exemplos de classe 2. Assim, quando não há nenhuma regra que prediz essa classe, como no caso das hipóteses h_{accR} , h_{sensR} , h_{specR} e h_{supR} , então os 16 exemplos são cobertos pela regra *default*. Como essa regra prediz a classe 1, todos esses exemplos são classificados erroneamente, incrementando o erro dos classificadores construídos pelo RCO . Pode-se assim concluir que não há “melhores” regras para prever a classe 2 utilizando esses quatro critérios.

Já para a classe 3, observa-se em geral uma pequena quantidade de exemplos cobertos pela regra *default*; no entanto, as regras selecionadas que classificam exemplos como sendo pertencentes a essa classe não os cobrem corretamente.

Fazendo uma análise da hipótese h_{specR} , a qual apresentou uma alta taxa de erro, deve ser lembrado que o critério $specR$ é uma medida de avaliação de regra do número (relativo)

Classe	h_{accR}	h_{covR}	h_{satR}	h_{sensR}	h_{specR}	h_{supR}	# Exemplos em S_4
1	6	6	6	6	17	6	26
2	16	13	13	16	16	16	16
3	4	2	2	2	16	3	18
Total	26	21	21	24	49	25	60

Tabela 7.5: Número de Exemplos Classificados por Classe pela Regra *Default*

Classe	h_{accR}	h_{covR}	h_{satR}	h_{sensR}	h_{specR}	h_{supR}	Distribuição dos Exemplos
1	7	2	7	2	3	3	42.92%
2	0	1	1	0	0	0	22.08%
3	4	3	5	3	1	2	35.00%
1 (<i>default</i>)	1	1	1	1	1	1	–
Total	12	7	14	6	5	6	–

Tabela 7.6: Número de Regras Seleccionadas por Classe

de exemplos da classe prevista pela *Cabeça* cobertos pela regra. É definida pela propriedade de o *Corpo* ser falso dado que a *Cabeça* é falsa (Prati, Baranauskas, & Monard 2001a). Isso implica que, se a *Cabeça* da regra não coincide com a classe do exemplo então ela não cobre o exemplo, o que é equivalente dizer que uma regra com alta especificidade, quando cobre um exemplo, ela o faz corretamente. Se for observada a Tabela 7.5, há somente 11 exemplos não cobertos pela regra *default*, dos quais 2(dois) exemplos são da classe 2 e 9(nove) exemplos são da classe 1. Como os exemplos restantes são classificados pela regra *default* e a regra *default* classifica os exemplos como sendo pertencentes da classe 1, o erro cometido por esse classificador foi alto para estes exemplos. Para a classe 2, o erro cometido foi de 100%, e o mesmo não aconteceu para a classe 3 porque existe uma regra na hipótese que cobre corretamente 2 exemplos pertencentes a essa classe (Tabela 7.4). Deve ser observado que o erro para a classe 1 foi de 0% porque os exemplos cobertos pelas regras da hipótese cobrem corretamente os exemplos da classe 1, e o restante dos exemplos é coberto (corretamente) pela regra *default*.

Quanto às regras seleccionadas pelo *RCO* para os 6(seis) critérios utilizados, foi observado que foram seleccionadas um total de 17 regras diferentes que participam dos classificadores construídos pelo *RCO*. Nas hipóteses construídas, essas 17 regras aparecem em ordens diferentes nas hipóteses ou aparecem em algumas hipóteses e não aparecem em outras. Esse fato foi observado quando se deu a comparação das hipóteses construídas com algumas hipóteses analisadas pelo especialista Dr. Sandro C. Esteves. A Tabela 7.7 mostra o corpo e a cabeça (classe) das regras pertencentes às hipóteses construídas pelo *RCO*, na sintaxe padrão em Prolog, o número de vezes que a regra foi seleccionada utilizando esses seis critérios, e a opinião do especialista em relação à regra (**N** se não faz sentido, **OK** se a regra faz sentido e **?** se a regra não foi analisada pelo especialista). Quando a regra foi rotulada por **N** na opinião do especialista, a tabela também contém o índice do motivo que o levou a dizer que a regra não faz sentido. Os motivos do especialista e seus respectivos índices constam na Tabela 7.8.

Observa-se, na Tabela 7.7, que, das regras analisadas pelo especialista, 50% das regras

Corpo	Cabeça (Classe)	# Vezes Seleccionada	Opinião do Especialista	Motivo
greater(idade,36.5),greater(class_B,29.5)	1	3	N	1
greater(tempo_abs,1.5),less(class_C,65),less(vitalidade,53.5)	1	3	N	3
greater(volume,4.1),greater(pH,8.25)	3	3	N	4
less(processamento,32.5),less(volume,4.2),less(concentracao_total,27.95)	1	3	OK	-
greater(pH,7.75),less(pH,8.25),less(concentracao_total,72.7),less(motilidade,12.5)	1	3	OK	-
greater(class_C,34.5),less(kruger,1.5)	1	2	OK	-
greater(idade,29),less(tempo_abs,6.5),less(hP,58.5)	1	2	?	-
less(idade,48.5),less(concentracao,33.65),greater(class_B,7),less(hP,75.5)	1	5	N	1 e 5
greater(vitalidade,70.5),greater(kruger,11.5),greater(hP,60)	3	2	OK	-
greater(concentracao_total,221.25),less(class_B,28.5),less(class_C,70.5)	3	5	OK	-
greater(concentracao,24),greater(motilidade,28.5)	3	2	OK	-
greater(volume,1.35),less(num_leu,1.28),less(hP,69)	1	3	N	-
greater(motilidade,19.5),less(num_leu,0.68)	3	1	N	7
less(pH,8.25),greater(concentracao,63.5),greater(motilidade,0.5)	3	4	OK	Observar Motivo 6
less(class_A,11.5)	2	1	N	1
greater(class_B,25.5),greater(kruger,3.5)	2	1	OK	
greater(idade,31.5),less(processamento,35),greater(concentracao,20.35),less(class_A,21.5)	3	1	N	1

Tabela 7.7: Regras Seleccionadas pelo $\mathcal{RCO}$

Motivos do Especialista	
1:	Quanto maior o valor dos atributos class_A ou class_B, maior a chance da regra prever as classes 2 ou 3
2:	Quanto maior o valor dos atributos class_C ou class_D, menor a chance da regra prever a classe 3
3:	Quanto menor o valor dos atributos class_C ou class_D, menor a chance da regra prever a classe 1
4:	Especialista não utilizaria os atributos da regra
5:	Valor de corte muito alto para o atributo hP
6:	Valor de corte baixo para o atributo motilidade
7:	Quanto menor o valor do atributo num_leu, menor a chance da regra prever a classe 3

Tabela 7.8: Motivos do Especialista para Considerar que uma Regra Não Faz Sentido

segue o raciocínio que o especialista utiliza para analisar um caso. Os outros 50% não fazem sentido ora porque o *Corpo* da classe possui condições falsas, ora porque utiliza atributos que o especialista não utiliza para a análise.

As 3(três) primeiras regras da Tabela 7.7 foram selecionadas pelos critérios `accR`, `satR` e `specR` como as primeiras regras com melhor valor nessas medidas. No entanto, para o especialista, elas não fazem sentido, o que mostra que essas medidas devem ser melhor analisadas para utilizá-las como critério de seleção de regras para a construção de uma hipótese, utilizando o *R_{CO}*.

7.5 Considerações Finais

Com os resultados obtidos nesse experimento e nos experimentos realizados com o conjunto de dados artificiais, foi possível concluir que a idéia de combinar classificadores simbólicos para formar um único classificador é válida, porém muito se pode fazer para melhorar o algoritmo *R_{CO}*. Deve ser observado que esses experimentos encontram-se descritos com maiores detalhes em (Bernardini & Monard 2002a). As idéias que surgiram são descritas no próximo capítulo, onde constam as conclusões de todo o trabalho realizado.

Conclusão

Neste trabalho é proposta a combinação de classificadores induzidos por algoritmos de AM simbólicos para gerar um novo classificador, bem como explicar a classificação de novos exemplos realizada pelo ensemble formado por esses classificadores simbólicos. Para analisar a viabilidade dessa proposta, foi implementado o sistema $\mathcal{R}_{ule}System$ na linguagem de programação lógica Prolog.

A prototipagem rápida oferecida pela linguagem Prolog agilizou o processo de implementação do sistema $\mathcal{R}_{ule}System$. 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 desenvolvida 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.

Em relação 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 realizados utilizando um conjunto de dados artificiais, foi possível observar a necessidade de melhorar o critério de seleção de regras do algoritmo $\mathcal{RCO}$ em caso de empate. Esse fato mostrou a importância de se utilizar bases de dados artificiais pois, dessa forma, os experimentos são bastante controlados. Em experimentos

controlados, há a facilidade de melhorar alguns critérios dos algoritmos envolvidos nos experimentos bem como motivar novas idéias.

Assim, o critério de desempate utilizado por $\mathcal{RCO}$ foi revisto e foi observado que, após a revisão ocorrida no $\mathcal{RCO}$, houve uma melhora significativa na eleição das regras. 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.

Existem inúmeros critérios de seleção da melhor regra de uma hipótese h (Horst & Monard 2000; Gomes 2002). Um dos possíveis critérios consiste em escolher a regra com o melhor grau de adequação (*rule fitness*) para o problema em questão. Dada uma regra R , representada por $B \rightarrow H$, o grau de adequação de R pode ser denotado por $\text{rf}(B \rightarrow H)$. O critério de seleção rf escolhe a melhor regra como sendo aquela que possui maior valor de rf . Existem três possíveis critérios que permitem estimar o grau de adequação de uma regra:

Confiabilidade Positiva: O primeiro é a confiabilidade positiva, a qual utiliza a precisão de regra (accR). Entretanto, a confiabilidade positiva possui uma propriedade indesejada. Por exemplo, dadas duas regras R_u e R_q , supondo que para a regra R_u , o valor de accR seja 1, e para a regra R_q , o valor de accR seja 0.98; segundo o critério atual de $\mathcal{RCO}$, a regra R_u é melhor que a regra R_q . Entretanto, se a regra R_u cobre somente um exemplo e a regra R_q cobre 200 exemplos (portanto, 4 deles são cobertos erroneamente), a regra que deveria ser eleita como a melhor regra seria a regra R_q .

Precisão de Laplace Uma solução para o problema apresentado pela confiabilidade positiva (accR) consiste em substituí-la pela precisão de Laplace (Clark & Boswell 1991), definida pela Equação 8.1, onde NCl é o número de classes do conjunto de exemplos, bh é o número de exemplos cobertos corretamente pela regra e $b\bar{h}$ é o número de exemplos cobertos incorretamente pela regra.

$$\text{rf}(B \rightarrow H) \stackrel{\text{def}}{=} \text{lacc}(B \rightarrow H) = \frac{bh + 1}{b + NCl} = \frac{bh + 1}{bh + b\bar{h} + NCl} \quad (8.1)$$

Considerando esse critério, quanto maior o valor da precisão de Laplace, melhor é a regra. Assim, ainda considerando o exemplo anterior em um problema com $NCl = 2$ classes, tem-se que $\text{lacc}(R_u) = 0.66$ e $\text{lacc}(R_q) = 0.98$, indicando que a regra R_q é melhor que a regra R_u . Assim como a confiabilidade positiva, a precisão de Laplace assume valores no intervalo $0 \leq \text{lacc}(B \rightarrow H) \leq 1$.

Novidade: Esta medida de avaliação de regra está implementada no Módulo de Análise de Regras (Gomes 2002), outro módulo principal do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$.

Futuramente, deverá ser investigado se os dois últimos critérios — precisão de Laplace e novidade — oferecem bons resultados quando utilizados com o $\mathcal{RCO}$.

A idéia de combinar diversos classificadores bem como utilizar melhores heurísticas para seleção de regras é reforçada quando se analisa os resultados obtidos no experimento utilizando o conjunto de dados reais. Das regras selecionadas pelo $\mathcal{RCO}$ neste experimento

e analisadas pelo especialista do domínio, 50% delas não são boas regras do ponto de vista do especialista. Entretanto, 50% delas o são, o que mostra que a idéia é viável mas precisa ser melhorada.

O erro obtido pelos classificadores construídos utilizando a versão atual do $\mathcal{RCO}$ deixa a desejar em todos os experimentos realizados. Deve ser observado que não necessariamente bons classificadores são construídos, mas sim classificadores com as melhores regras induzidas por diferentes algoritmos de aprendizado de máquina, ou induzidas por um mesmo algoritmo de aprendizado de máquina mas com diferentes conjuntos de exemplos, como dito anteriormente.

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 $\mathcal{RCO}$ (Capítulo 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 mais 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 x é selecionar, do conjunto de hipóteses $\mathcal{H}$, um conjunto de regras $\mathbf{R}$ com as regras que cobrem o exemplo 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 x e deve-se tentar criar uma única regra R' a partir do conjunto de regras $\mathbf{R}$. Se R' for construída, essa será a regra fornecida ao usuário para explicar a classificação de x .

Análise Qualitativa de Regras

Sob o aspecto qualitativo, regras são avaliadas com o objetivo de saber quais são aquelas melhor sustentadas pelos dados (Lavrac, Flach, & Zupan 1999). A análise qualitativa de regras pode ser efetuada por meio de procedimentos que implementam medidas de avaliação de regras.

Deve ser observado que o texto deste capítulo foi integralmente baseado em (Gomes 2002), onde há uma descrição mais completa dessas informações, e informações sobre outras medidas de regras.

A.1 Avaliação de Regras

Dada uma regra R , denotada por $B \rightarrow H$, onde B é o corpo da regra e H é a cabeça da regra e dado um exemplo x rotulado por y , a avaliação da regra é feita através de um procedimento que determina se B é verdadeiro ou falso e se H é verdadeiro ou falso para esse exemplo. Por meio desse procedimento é possível determinar, para cada regra R_u de um classificador simbólico $h = \{R_1, \dots, R_{NR}\}$, as seguintes informações:

- hb número de exemplos para os quais H é verdade e B é verdade e
- $\bar{h}b$ número de exemplos para os quais H é falso e B é verdade e
- $\bar{h}\bar{b}$ número de exemplos para os quais H é falso e B é falso e
- $h\bar{b}$ número de exemplos para os quais H é verdade e B é falso.

No cálculo dessas informações deve ser considerado a forma que as regras do classificador simbólico h foram induzidas pelo algoritmo de AM. O Módulo de Análise de Regras calcula essas informações para regras *unordered*, *ordered* e *interclass*, conforme descrito a

seguir.

Unordered

Para um conjunto de regras *unordered*, essas informações são calculadas verificando a cobertura de cada regra do classificador simbólico $h = \{R_1, \dots, R_{NR}\}$ em todo conjunto de exemplos S . O Algoritmo 8 mostra como é realizado o cálculo dessas informações para um conjunto de regras *unordered*.

Algoritmo 8 Cálculo de Informações para Avaliação de Regras *Unordered*

Require: $h = \{R_1, \dots, R_{NR}\}$: Conjunto de Regras *Unordered*

$S = \{(x_1, y_1), \dots, (x_N, y_N)\}$: Conjunto de Exemplos

```

1:  $L := [0, 0, 0, 0]$ ;
2: for all  $R_u \in h$  do
3:   for all  $(x_i, y_i) \in S$  do
4:      $j := \text{cobertura}(R_u, (x_i, y_i))$ ;
5:     Incrementar em 1 o elemento  $j$  da lista  $L$ ;
6:   end for
7: end for
8: return  $L$ 
```

A função $\text{cobertura}(R_u, (x_i, y_i))$ retorna um número inteiro no intervalo $1 \cdot 4$ que é usado como índice para realizar o elemento correspondente da lista L . O índice 1 corresponde a hb , 2 a $\bar{h}b$, 3 a $\bar{h}\bar{b}$, 4 a $h\bar{b}$.

Ordered

No caso de regras *ordered* existe um *else* implícito entre cada regra, assim o exemplo deve ser aplicado a partir da primeira regra, até que uma delas cubra esse exemplo (B é verdade para o exemplo). Esse exemplo deve ser coberto apenas por essa regra. Para as regras seguintes àquela que cubra o exemplo, as informações devem ser atualizadas incrementando-se os valores de $\bar{b}h$ e $\bar{b}\bar{h}$ conforme a classe C_v do exemplo. O Algoritmo 9 mostra como é realizado o cálculo dessas informações para um conjunto de regras *ordered*.

A variável *first* está associada a cada exemplo (x_i, y_i) e recebe *false* se o exemplo não foi coberto por uma regra ou *true* se já foi coberto.

Interclass

As regras *interclass* são separadas em C_v blocos para cada uma das $C_v, v = 1, \dots, NCl$ diferentes classes, existindo um *else* implícito entre cada um desses blocos. Para regras em um mesmo bloco C_v , as informações são calculadas como descrito anteriormente para regras *unordered*. Ao ser encontrada uma regra que cubra o exemplo em um bloco C_v , as informações extraídas das regras subseqüentes devem ser atualizadas como descrito anteriormente para regras *ordered*, nos C_{NCl-v} blocos seguintes.

Por exemplo, a Figura 5.1 na página 42 mostra que, para cada classe, existe um conjunto de subconjuntos de regras $h' = \{h'_1, \dots, h'_{NCl}\}$ ¹. É importante observar que,

¹ $h' = h - \text{default}$. A regra *default* é uma regra especial cujo corpo é vazio e a cabeça é a classe de maior

Algoritmo 9 Cálculo de Informações para Avaliação de Regras *Ordered***Require:** $h = \{R_1, \dots, R_{NR}\}$: Conjunto de Regras *Ordered* $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$: Conjunto de Exemplos

```

1:  $L := [0, 0, 0, 0]$ ;
2:  $first := false$ ;
3: for all  $R_u \in h$  do
4:   for all  $(x_i, y_i) \in S$  do
5:      $j := cobertura(R_u, (x_i, y_i))$ ;
6:     if  $first = true$  and  $(j = 1$  or  $j = 2)$  then
7:        $j := 4 - j + 1$ ;    {o exemplo já foi coberto por uma regra. Assim, o elemento  $j$ 
                             deverá ser igual a 3 ou 4 }
8:     end if
9:     if  $first = false$  and  $(j = 1$  or  $j = 2)$  then
10:       $first := true$ ;
11:    end if
12:    Incrementar em 1 o elemento  $j$  da lista  $L$ ;
13:  end for
14: end for
15: return  $L$ 

```

além do else implícito entre cada um desses subconjuntos de regras, esses subconjuntos são formados conforme a ordem em que o algoritmo de AM induz as classes que compõem a hipótese h . Cada elemento de h' , ou seja, $h'_1, \dots, h'_{N_{Cl}}$, é uma hipótese com as regras que predizem a mesma classe.

Assim, o exemplo deve ser aplicado a todas as regras de h'_1 e as informações são calculadas como descrito anteriormente para regras *unordered*. Se nenhuma regra de h'_1 cobrir o exemplo, ele deve ser aplicado a h'_2 e assim por diante. Ao ser encontrada uma regra em um h'_v que cobre o exemplo, as informações extraídas das regras subsequentes são atualizadas como descrito anteriormente para regras *ordered*, nos $h'_{N_{Cl}-v}$ subconjuntos seguintes.

Após determinar as informações para cada uma das regras R_u ² de um classificador simbólico h , os valores encontrados podem ser expressos sob a forma de frequências relativas, as quais são utilizadas nesse trabalho como uma estimativa de probabilidade. Por exemplo, considerando a informação hb , a probabilidade $P(HB)$ pode ser determinada da seguinte forma:

$$P(HB) = f_{bh} = \frac{hb}{N}$$

onde N é o número total de exemplos utilizados para calcular a informação hb .

De forma semelhante podem ser determinados os valores das probabilidades $P(\overline{HB})$, $P(\overline{HB})$ e $P(H\overline{B})$. Conhecidas essas probabilidades, os valores de $P(B)$, $P(\overline{B})$, $P(H)$ e $P(\overline{H})$ podem ser determinados. Por exemplo:

frequência no conjunto de exemplos. Assim, uma hipótese h sempre irá classificar um exemplo devido à existência da regra *default*.

²Se R_u é a regra *default*, então as informações não são calculadas para essa $R_u \in h$.

$$P(B) = P(\overline{H}B) + P(HB) \text{ ou } P(\overline{B}) = P(H\overline{B}) + P(\overline{H}\overline{B})$$

Neste trabalho, essas probabilidades são utilizadas para calcular as medidas de avaliação de regras.

A.2 Medidas de Avaliação de Regras

No *framework* proposto por Lavrac em (Lavrac, Flach, & Zupan 1999), algumas medidas de avaliação encontradas na literatura foram unificadas, considerando uma regra R_u na forma $B \rightarrow H$. Essas medidas podem ser divididas nas seguintes categorias:

1. Medidas genéricas de avaliação;

Precisão	$Acc(B \rightarrow H) = P(H B) = \frac{hb}{b}$
Erro	$Err(B \rightarrow H) = P(\overline{H} B) = \frac{\overline{h}b}{b}$
Confiança negativa	$NegRel(B \rightarrow H) = P(\overline{H} \overline{B}) = \frac{\overline{h}\overline{b}}{\overline{b}}$
Sensitividade	$Sens(B \rightarrow H) = P(B H) = \frac{hb}{h}$
Especificidade	$Spec(B \rightarrow H) = P(\overline{B} \overline{H}) = \frac{\overline{h}\overline{b}}{\overline{h}}$
Cobertura	$Cov(B \rightarrow H) = P(B) = \frac{b}{N}$
Suporte	$Sup(B \rightarrow H) = P(HB) = \frac{hb}{N}$
Novidade	$Nov(B \rightarrow H) =$ $P(HB) - P(H)P(B) = \frac{1}{N}(hb - \frac{h \cdot b}{N})$
Satisfação	$Sat(B \rightarrow H) = \frac{(P(\overline{H}) - P(\overline{H} B))}{P(\overline{H})} =$ $\frac{\frac{\overline{h}}{N} - \frac{\overline{h}b}{b}}{\frac{\overline{h}}{N}} = 1 - \left(\frac{\overline{h}b}{b} \cdot \frac{N}{\overline{h}} \right)$

2. Medidas relativas de avaliação;

Precisão relativa	$RAcc(B \rightarrow H) = P(H B) - P(H) = \frac{hb}{b} - \frac{h}{N}$
Confiança negativa relativa	$RNegRel(B \rightarrow H) = P(\overline{H} \overline{B}) - P(\overline{H}) = \frac{\overline{h}\overline{b}}{\overline{b}} - \frac{\overline{h}}{N}$
Sensitividade relativa	$RSens(B \rightarrow H) = P(B H) - P(B) = \frac{hb}{h} - \frac{b}{N}$
Especificidade relativa	$RSpec(B \rightarrow H) = P(\overline{B} \overline{H}) - P(\overline{B}) = \frac{\overline{h}\overline{b}}{\overline{h}} - \frac{\overline{b}}{N}$

3. Medidas relativas de avaliação com peso;

Precisão relativa com peso	$WRAcc(B \rightarrow H) = P(B)(P(H B) - P(H)) = \frac{b}{N} \left(\frac{hb}{b} - \frac{h}{N} \right)$
Confiança negativa relativa com peso	$WRNegRel(B \rightarrow H) = P(\overline{B})(P(\overline{H} \overline{B}) - P(\overline{H})) = \frac{\overline{b}}{N} \left(\frac{\overline{h}\overline{b}}{\overline{b}} - \frac{\overline{h}}{N} \right)$
Sensitividade relativa com peso	$WRsens(B \rightarrow H) = P(H)(P(B H) - P(B)) = \frac{h}{N} \left(\frac{hb}{h} - \frac{b}{N} \right)$
Especificidade relativa com peso	$WRSpec(B \rightarrow H) = P(\overline{H})(P(\overline{B} \overline{H}) - P(\overline{B})) = \frac{\overline{h}}{N} \left(\frac{\overline{h}\overline{b}}{\overline{h}} - \frac{\overline{b}}{N} \right)$

Baseada na medida de novidade, Lavrac mostra que as medidas relativas de avaliação com peso são iguais entre si e iguais a medida de novidade da regra, i.e.

$$\begin{aligned}
 WRAcc(B \rightarrow H) &= WRsens(B \rightarrow H) = WRSpec(B \rightarrow H) \\
 &= WRNegRel(B \rightarrow H) = Nov(B \rightarrow H)
 \end{aligned}$$

Geralmente, medidas de avaliação são úteis para determinar quais são as “melhores” regras de um classificador simbólico h . As “melhores” regras podem ser aquelas que apresentam o maior (ou menor) valor para uma determinada medida de avaliação. Por exemplo, pode ser considerado que as “melhores” regras R_u de um classificador simbólico h , são aquelas que possuem maior valor para a medida de Precisão (ou menor valor para a medida de Erro).

Referências

- Ali, K. M. & M. J. Pazzani (1996). Error reduction through learning multiple descriptions. *Machine Learning* 24(3), 173–202.
- Baranauskas, J. A. (2001). Extração automática de conhecimento por múltiplos indutores. Tese de Doutorado, ICMC-USP. <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-08102001-112806/>.
- Baranauskas, J. A. & G. E. A. P. A. Batista (2000). O projeto DISCOVER: Idéias iniciais (comunicação pessoal).
- Baranauskas, J. A. & M. C. Monard (2000, Fev). 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, Out). Um ambiente de avaliação de algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC-USP.
- Batista, G. E. A. P. A. (2000, Mar). 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, Mai). Sintaxe padrão do arquivo de exemplos do projeto DISCOVER. <http://www.icmc.sc.usp.br/~gbatista/SintaxePadraoFinal.htm>.
- Bauer, E. & R. Kohavi (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning* 36(1/2), 105–139.
- Bernardini, F. C. & M. C. Monard (2002a, Abr). Um estudo de caso utilizando o módulo de explicação e combinação do $\mathcal{R}_{uleSystem}$. Technical Report 160, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_160.ps.zip.
- Bernardini, F. C. & M. C. Monard (2002b, Abr). Um módulo de explicação de *ensembles* e combinação de regras. Technical Report 159, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_159.ps.zip.
- Bernardini, F. C. & M. C. Monard (2002c, Abr). Uma comparação experimental de *ensembles* de árvores de decisão e de regras utilizando a técnica de boosting. Technical Report 162, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_161.ps.zip.

- Blake, C., E. Keogh, & C. Merz (1998). UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- Blum, A. & P. Langley (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence* 97, 245–271.
- Blum, A. & R. Rivest (1988). Training a 3-node neural network is NP-complete (extended abstract). In *Proceeding of the 1988 Workshop on Computational Learning Theory*, San Mateo, California, pp. 9–18. Morgan Kaufmann Publishers.
- Bratko, I. (1990). *Prolog Programming for Artificial Intelligence*. Addison-Wesley.
- Breiman, L. (1996a). Bagging predictors. *Machine Learning* 24(2), 123–140.
- Breiman, L. (1996b). Bias, variance and arcing classifiers. Technical Report 460, Statistics Department, University of California. <ftp://ftp.stat.berkeley.edu/pub/users/breiman/>.
- Breiman, L. (1996c). Bias, variance and arcing classifiers. Technical Report 460, Statistics Department, University of California. <ftp://ftp.stat.berkeley.edu/pub/users/breiman/>.
- Cabena, P., P. Hadjinian, R. Stadler, J. Verhees, & A. Zanasi (1998). *Discovering Data Mining: from Concept to Implementation*. Prentice Hall.
- Caulkins, C. W. (2000, Ago). Aquisição de conhecimento utilizando aprendizado de máquina relacional. Dissertação de Mestrado, ICMC-USP.
- Clark, P. & R. Boswell (1991). Rule induction with $\mathcal{CN}^2$: Some recent improvements. In Y. Kodratoff (Ed.), *Proceedings of the 5th European Conference (EWSL 91)*, pp. 151–163. Springer Verlag.
- Clark, P. & T. Niblett (1989). The $\mathcal{CN}^2$ induction algorithm. *Machine Learning* 3(4), 261–283.
- Dietterich, T. G. (1997). Machine learning research: Four current directions (draft). *AI Magazine* 18, 97–136. <http://www.cs.orst.edu/~tgd/>.
- Dietterich, T. G. (2000a). Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems. Lecture Notes in Computer Science*, New York, pp. 1–15. Springer Verlag. <http://www.cs.orst.edu/~tgd/>.
- Dietterich, T. G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning* 40, 139–157. <http://www.cs.orst.edu/~tgd/>.
- Dietterich, T. G. (2002). Ensemble learning. In M. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks, Second Edition*. MIT Press. (To appear). <http://www.cs.orst.edu/~tgd/> [date:04/2002].
- Dietterich, T. G. & G. Bakiri (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2, 263–286.
- Efron, B. & R. Tibshirani (1993). *An Introduction to Bootstrap*. Springer Verlag.
- Esteves, S. & F. Bento (1998). Sperm processing for assisted reproductive technology (art): effects of pentoxifyline on recovery of motile sperm in asthenozoospermic men. *FertilSteril* 70, 5196.

- Esteves, S., H. Lee, M. C. Monard, & L. Lopes (2001, Out). Inteligência artificial aplicada à andrologia: um estudo de caso do processamento de sêmen diagnóstico. In *XXVIII Congresso Brasileiro de Urologia*, Fortaleza, CE, Brasil.
- Esteves, S., R. Sharma, A. Thomas, & A. Agarwal (2000). Improvement in motion characteristics and acrosome status in cryopreserved human spermatozoa by swimup processing before freezing. *HumProd* 15, 2173–2179.
- 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, Set). A multi-criteria approach for the evaluation of rule interestingness. In *Proceedings of the International Conference on Data Mining*, Rio de Janeiro, RJ, pp. 7–20.
- Freitas, A. A. (1998b). On objective measures of rule surprisingness. In Springer Verlag (Ed.), *Proc. 2nd European Symp. on Principles of Data Mining and Knowledge Discovery (PKDD-98). Lecture Notes in Artificial Intelligence*, Volume 1510, pp. 1–9.
- Freund, Y. & R. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1), 119–139.
- Gomes, A. K. (2002). Medidas de avaliação de regras. Dissertação de Mestrado, ICMC-USP.
- Gomes, A. K., F. C. Bernardini, & M. C. Monard (2002, Jan). 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. & M. C. Monard (2002a, Mar). Um estudo de caso utilizando o módulo de análise de regras do $\mathcal{R}_{ule}S_{ystem}$. Technical Report 158, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_158.ps.zip.
- Gomes, A. K. & M. C. Monard (2002b, Fev). 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.
- Hansen, L. & P. Salamon (1990). Neural networks ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(10), 993–1001.
- Hashem, S. (1993). *Optimal Linear Combinations of Neural Networks*. Tese de Doutorado, Purdue University, School of Industrial Engineering, Lafayette, IN.
- Hilderman, R. J. & H. J. Hamilton (1999, Out). Knowledge discovery and interestingness measures: A survey. Technical Report s4s, University of Regina.
- Horst, P. & M. C. Monard (2000, Nov). Um sistema computacional para avaliação de regras induzidas por algoritmos de aprendizado de máquina. In *Proceedings IBERAMIA-SBIA 2000 Open Discussion Track*, Atibaia, SP, Brasil, pp. 167–176.
- Horst, P. S. (1999, Ago). Avaliação do conhecimento adquirido por algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC-USP.
- Hyafil, L. & R. Rivest (1976). Constructing optimal binary decision trees is np-complete. *Information Processing Letters* 5(1), 15–17.

- Imamura, C. Y. (2001). Pré-processamento para extração de conhecimento de bases textuais. Dissertação de Mestrado, ICMC-USP.
- Jordan, M. & R. Jacobs (1994). Hierarchical mixtures of experts and EM algorithm. *Neural Computation* 6, 181–214.
- Kolen, J. & J. Pollack (1991). Back propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems*, Volume 3, San Mateo, California, pp. 860–867. Morgan Kaufmann Publishers.
- Lavrac, N., P. Flach, & B. Zupan (1999). Rule evaluation measures: a unifying view. In Springer Verlag (Ed.), *Proc. 9th International Workshop on Inductive Logic Programming. Lecture Notes in Artificial Intelligence*, Volume 1634, pp. 74–185.
- Lee, H. D. (2000). Seleção e construção de features relevantes para o aprendizado de máquina. Dissertação de Mestrado, ICMC-USP. <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-15032002-113112/>.
- Lee, H. D. & M. C. Monard (2000a, Nov). Indução construtiva guiada pelo conhecimento: Um estudo de caso do processamento de sêmen diagnóstico. In *Proceedings IBERAMIA-SBIA 2000, Open Discussion Track*, Atibaia, SP, Brasil, pp. 167–176.
- Lee, H. D. & M. C. Monard (2000b). A practical approach for knowledge-driven constructive induction. In *Proceedings Argentine Symposium on Artificial Intelligence, ASAI'2000, 29th International Conference SADIO*, Argentina, pp. 71–86.
- Mannila, H. (1996). Data mining: Machine learning, statistics, and database. Technical report, Department of Computer Science, University of Helsinki, FIN-00014 Helsinki, Finland. <http://www.cs.helsinki.fi/~mannila> [date: mar/2001].
- Margineantu, D. & T. G. Dietterich (1997). Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning*, San Mateo, California, pp. 211–218. Morgan Kaufmann Publishers.
- Martins, C. A. (2001). Clustering conceitual em aprendizado de máquina. Exame de Qualificação de Doutorado, ICMC-USP.
- Michalski, R., I. Bratko, & M. Kubat (1998). *Machine Learning and Data Mining: Methods and Applications*. John Wiley & Son.
- Milaré, C. R. (2000, Mar). Extração de conhecimento de redes neurais. Exame de Qualificação de Doutorado, ICMC-USP.
- Mitchell, T. (1999). Machine learning and data mining. *Communications of the ACM* 42(11), 30–36. <http://cs.cmu.edu/user/mitchell/ftp/tomhome.html>.
- Nagai, W. A. (2000). Avaliação do conhecimentos extraído de problemas de regressão. Dissertação de Mestrado, ICMC-USP.
- Neal, R. (1993). Probabilistic inference using markov chain monte carlo methods. Technical report, CRG-TR-93-1. Department of Computer Science, University of Toronto, Toronto, CA.
- Opitz, D. (1999). Feature selection for ensembles. Computer Science Department, University of Montana, Missoula, MT.

- Parmanto, B., P. Munro, & H. Doile (1992). Improving committee diagnosis with resampling techniques. In D. Touretzky, M. Mozer, & M. Hesselmo (Eds.), *Advances in Neural Information Processing Systems*, Volume 8, Cambridge, MA, pp. 882–888. The MIT Press.
- Perrone, M. & L. Cooper (1993). When networks disagree: Ensemble methods for hybrid neural networks. In R. Mamone (Ed.), *Neural Networks for Speech and Image Processing*. Chapman & Hall.
- Pila, A. D. (2000, Ago). Seleção de atributos relevantes para aprendizado de máquina utilizando a abordagem de rough sets. Dissertação de Mestrado, ICMC-USP. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-13022002-153921%/publico/dissertacao_ADP.pdf.
- Prati, R. C., J. A. Baranauskas, & M. C. Monard (2001a, Jul). 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., J. A. Baranauskas, & M. C. Monard (2001b, Mar). 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, Mar). O pós-processamento em extração de conhecimento de bases de dados. Exame de Qualificação de Doutorado, ICMC-USP.
- Quinlan, J. (1988). *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers.
- Quinlan, J. (1999). Miniboosting decision trees. Submitted to Journal of Artificial Intelligence Research (available at <http://www.cse.unsw.EDU.AU/~quinlan/miniboost.ps> [date: 04/2002]).
- Quinlan, J. (2002). See5: An informal tutorial. Available at <http://www.rulequest.com/see5-win.html> [date: Abril/2002].
- Weiss, S. & N. Indurkha (1998). *Predictive Data Mining: a Practical Guide*. Morgan Kaufmann Publishers. <http://www.data-miner.com.or>.
- Weiss, S. M. & C. A. Kulikowski (1991). *Computer Systems that Learn*. San Mateo, CA: Morgan Kaufmann Publishers.
- Westwood, D. (2000a, Abr). LPA-PROLOG for windows programming guide.
- Westwood, D. (2000b, Abr). LPA-PROLOG technical reference.
- Wolpert, D. (1992). Stacked generalization. *Neural Networks* 5, 241–259.