

Pós-processamento de regras de regressão

Jaqueleine Brighadori Pugliesi

Orientadora: Profa. Dra. Solange Oliveira Rezende

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional..

“VERSAO REVISADA APÓS A DEFESA”

Data da Defesa:	07/05/2004
Visto do Orientador:	<i>Rezende.</i>

USP – São Carlos
Maio/2004

Este documento foi preparado utilizando-se o formatador de textos L^AT_EX. Sua bibliografia é gerada automaticamente pelo BIBT_EX, utilizando o estilo Chicago.

Aos meus pais, Denizar e Maria Ilzi,
meu irmão Denizar e minha irmã Helena.

Agradecimentos

À Solange Oliveira Rezende, minha orientadora no mestrado e doutorado, pelo incentivo, paciência e dedicação, que durante esses anos todos de convivência, passou de simplesmente orientadora à orientadora/amiga. Solange, muito obrigada por tudo!

À Maria Carolina Monard pelo carinho, incentivo e amizade dispensados.

Ao Luís Torgo, pela atenciosa supervisão durante meu estágio em Portugal e valiosas contribuições para a realização desta tese.

Ao Pavel Brazdil, Nitin Indurkhyá e João Gama, pelos comentários e sugestões para o meu trabalho, e todos os integrantes do LIACC pela atenção a mim dedicada durante o período que estive na cidade do Porto. Em especial à Rita Ribeiro pelos bons momentos propiciados e ao Joel Silva que desde antes da minha viagem já se mostrou atencioso. Lá também tive a oportunidade de conhecer novos amigos com os quais compartilhei bons momentos e que tornaram mais fácil minha estadia no Porto.

À Fapesp pelo apoio financeiro, sem o qual não seria possível a realização desta tese.

Ao assessor da Fapesp, pelas contribuições e sugestões nos relatórios encaminhados.

As funcionárias da Pós-Graduação (Beth, Laura e Ana Paula), da biblioteca e à todos os funcionários do ICMC, pela maneira prestativa e educada que sempre me trataram.

A toda a equipe que vem desenvolvendo e tornando concreto o Ambiente DISCOVER pelas valiosas conversas e debates, dos quais surgiram boas idéias.

À Roberta, não só pela maneira competente e responsável com que realizou nossos projetos conjuntos (Roberta, foi muito bom trabalhar com você!), mas também pelo apoio incondicional em vários momentos do meu doutorado. Você é muito mais do que uma colega de trabalho, é uma amiga.

Ao Daniel, pelo auxílio e colaboração dado nesta tese durante o tempo em que realizamos trabalho conjunto e por ser um grande amigo.

Às minhas amigas Ana Elisa, Cristiane e Patricia, pelos grandes momentos também fora do ambiente profissional. A amizade, apoio e alegria de vocês foram essenciais nos meus momentos difíceis e muito importante para eu chegar onde estou hoje.

À Ana Cláudia, Luciane e Ludmila, amigas que me acompanham desde a época de graduação, pela constante amizade e carinho mesmo distantes.

Aos meus amigos (Edson Melanda, Fernanda, Veronica, Marcos Aurélio, Marcos Paula, Marina, Adriano, Walter, Valmir, Camila, Ronaldo, Augusto, Gustavo, Claudia Milaré, Claudia Martins, Katti, Ernesto, Edson Takashi, Marcos Geromini, Thais, José Flávio, Huei, Gedson, Rodrigo, Camilo, Naiara, Vitor e Fernanda Cockell, entre tantos outros), pelos bons momentos compartilhados. Em especial, à Roberta, Veronica e Edson pelo precioso tempo dispensado no auxílio e revisão desta tese.

A todos os amigos do LABIC, pelo companherismo, amizade e momentos de descontração.

Aos meus pais Denizar e Maria Ilzi, meu irmão Denizar e minha irmã Helena, pelo amor, apoio e compreensão pelo tempo que estive ausente. Gostaria também de agradecer a todos os meus familiares, à minha cunhada Silvana e ao meu sobrinho Felipe que trouxe mais alegrias às nossas vidas.

A todos aqueles que, direta ou indiretamente, contribuíram para a realização desta tese.

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

Resumo

O processo de Mineração de Dados inicia-se com o entendimento do domínio da aplicação, considerando aspectos como os objetivos da aplicação e as fontes de dados. Em seguida, é realizado o pré-processamento dos dados e a extração de padrões. Após a etapa de extração de padrões, vem a de pós-processamento, na qual o conhecimento é avaliado quanto a sua qualidade e/ou utilidade a fim de ser utilizado para apoio a algum processo de tomada de decisão. Recentemente, as pesquisas têm se voltado para problemas de regressão, porém a regressão em Mineração de Dados preditiva é uma questão pouco explorada dentro do processo de extração de conhecimento de bases de dados, sendo de grande relevância o estudo de métodos para a exploração de tarefas desse tipo. Alguns trabalhos vêm sendo realizados no Laboratório de Inteligência Computacional (LABIC) em temas relacionados ao processo de Extração de Conhecimento de Bases de Dados e Textos e na construção de um ambiente computacional para extração de conhecimento de dados denominado DISCOVER. Para apoiar a construção de um modelo de regressão simbólico e o pós-processamento de problemas de regressão foi proposto e desenvolvido o Ambiente *REPPE*. Esse ambiente viabiliza a avaliação de regras de regressão, inclusive disponibilizando estratégias para o cálculo da matriz de contingência e consequente utilização de todas as medidas derivadas dessa matriz para avaliação de regras de regressão; a combinação de regressores homogêneos e heterogêneos para melhorar a precisão dos regressores e a integração e poda de regras de regressão obtidas de diferentes amostras ou algoritmos. Essas funcionalidades do Ambiente *REPPE* incrementam a potencialidade do Ambiente DISCOVER quanto ao tratamento de regressão.

Abstract

Data Mining process begins with the understanding of the application domain, considering aspects as application objectives and data sources. Then, the data pre-processing and pattern extraction is realized. After the pattern extraction stage, one proceeds with the post-processing, in which the knowledge is evaluated as regards its quality and/or usefulness in order to use this knowledge to support a decision making process. Recently, much attention has been given to regression problems. However, regression in predictive Data Mining is a little explored subject in the knowledge discovery from database process, what makes the study of exploration methods very relevant. Some work in areas related to the Knowledge Discovery of Data and Text Bases process have been accomplished at LABIC (Laboratório de Inteligência Computacional) which motivated the construction of a computational environment for knowledge extraction called 'DISCOVER'. The *REPPE* environment was proposed and developed to aim the symbolic regression model construction and the regression problems post-processing. This environment makes possible the evaluation of regression rules, providing strategies for contingency table calculation and the subsequent utilization of all measures derived from this table for regression rules evaluation. Moreover, the system also provides the combination of homogeneous and heterogeneous regressors to improve the regressor precision and the integration and pruning of regression rules obtained from different samples or algorithms. These functionalities of *REPPE* increase the DISCOVER potentiality in relation to regression treatment.

Sumário

Agradecimentos	ix
Resumo	xi
Abstract	xiii
Sumário	xv
Lista de Figuras	xix
Lista de Tabelas	xxiii
Lista de Símbolos	xxvii
1 Introdução	1
1.1 Objetivos e Principais Contribuições	5
1.2 Organização do Trabalho	6
2 Mineração de Dados	7
2.1 O Processo de Mineração de Dados	9
2.1.1 Identificação do Problema	11
2.1.2 Pré-Processamento	12
2.1.3 Extração de Padrões	15
2.1.4 Pós-Processamento	18

2.2	Técnicas e Ferramentas Usadas em Mineração de Dados	20
2.3	Considerações Finais	22
3	Régressão	23
3.1	Definições e Formalizações	24
3.2	Métodos Estatísticos	25
3.2.1	Abordagem Paramétrica	25
3.2.2	Abordagem Não-Paramétrica	26
3.2.3	Abordagem Aditiva	29
3.3	Métodos de Aprendizado de Máquina Simbólico	30
3.3.1	Regras de Régressão	31
3.3.2	Árvores de Régressão	32
3.4	Métodos Baseados em Redes Neurais Artificiais	34
3.5	Métodos Baseados em <i>Support Vector Machines</i>	35
3.6	Considerações Finais	36
4	O Ambiente <i>RÉPPE</i>: Novas Funcionalidades no Discover	37
4.1	Contextualização	38
4.1.1	Notação e Terminologia	38
4.1.2	O Ambiente DISCOVER	41
4.1.3	Ferramentas de Régressão e Base de Dados	49
4.2	Sintaxe Padrão para Representar Regras de Régressão	57
4.2.1	Gramática da Sintaxe Padrão	58
4.2.2	Conversão dos Regressores para o Formato Padrão	62
4.3	Funcionalidades do Ambiente <i>RÉPPE</i>	67
4.3.1	Avaliação de Regras de Régressão no <i>RÉPPE</i>	68
4.3.2	Combinação de Regressores	72
4.3.3	Integração de Regras de Régressão	72
4.4	Considerações Finais	73
5	Avaliação de Regras de Régressão	75
5.1	Medidas Derivadas da Matriz de Contingência	76

5.2	Medidas Não Derivadas da Matriz de Contingência	81
5.3	Estratégias para Determinar Matriz de Contingência para Regras de Re-	
	gressão	86
5.3.1	Ajustamento	87
5.3.2	Pseudo-Classe	89
5.3.3	Discretização Baseada em Entropia	90
5.4	Avaliação Experimental	91
5.4.1	Descrição dos Experimentos	92
5.4.2	Resultados	93
5.5	Considerações Finais	95
6	Combinação de Regressores	101
6.1	Combinando Preditores Homogêneos	103
6.1.1	<i>Bagging</i>	104
6.1.2	<i>Boosting</i>	106
6.1.3	<i>Windowing</i>	107
6.1.4	<i>Wagging</i>	108
6.1.5	<i>Arcing</i>	111
6.1.6	<i>Randomization</i>	111
6.1.7	<i>Error-Correcting Output Codes</i>	113
6.2	Combinando Preditores Heterogêneos	113
6.2.1	<i>Stacking</i>	113
6.2.2	<i>Meta Learning</i>	116
6.3	Métodos para Combinação de Preditores	116
6.3.1	Métodos <i>Voting</i> versus Não- <i>Voting</i>	116
6.3.2	Métodos Lineares versus Não-Lineares	117
6.4	Algoritmos para Combinação de Regressores	118
6.4.1	<i>Bagging</i>	119
6.4.2	<i>Boosting</i>	119
6.5	Avaliação Experimental	123
6.5.1	Descrição dos Experimentos	123

6.5.2	Resultados	124
6.6	Considerações Finais	128
7	Integração de Regras de Regressão	131
7.1	Abordagens na Integração de Regras de Regressão	133
7.2	Metodologia para Integração de Regras de Regressão	134
7.3	Avaliação Experimental Utilizando a Metodologia <i>IRR</i>	136
7.3.1	Descrição dos Experimentos	136
7.3.2	Resultados	136
7.4	Avaliação Experimental Utilizando Seleção das Melhores Regras	146
7.4.1	Descrição dos Experimentos	147
7.4.2	Resultados	147
7.5	Considerações Finais	148
8	Conclusões	151
8.1	Contribuições desta Tese	152
8.2	Trabalhos Futuros	154
Referências Bibliográficas		157

Lista de Figuras

2.1	Etapas do processo de Mineração de Dados.	10
2.2	Tarefas de Mineração de Dados.	15
3.1	Exemplo de regressão linear global.	26
3.2	Exemplos mais similares do conjunto de treinamento.	27
3.3	Exemplo de uma <i>Model Tree</i>	33
3.4	Exemplo de uma rede neural artificial.	34
4.1	Funcionalidades do Ambiente DISCOVER.	42
4.2	Metodologia para obtenção de regras nas sintaxes padrão.	44
4.3	Exemplo de arquivo <code>.names</code> na sintaxe padrão.	45
4.4	Exemplo de arquivo <code>.data</code> na sintaxe padrão.	46
4.5	Exemplo de um conjunto de regras de regressão padronizadas.	57
4.6	Produções da gramática da sintaxe padrão de regras de regressão.	60
4.7	Regra de regressão gerada pelo algoritmo Cubist.	61
4.8	Regra de regressão da Figura 4.7 na sintaxe padrão.	61
4.9	Exemplo de arquivo de regras de regressão na sintaxe padrão.	61
4.10	Exemplo de arquivo de regras de regressão na sintaxe padrão estendida.	62
4.11	Produções que completam a gramática da sintaxe padrão de regras de regressão estendida.	62
4.12	Conversão das saídas dos algoritmos de regressão utilizados no DISCOVER e no <i>RÉPPE</i>	63

4.13	Algumas regras geradas pelo Cubist utilizando a base de dados Abalone.	64
4.14	Regras da Figura 4.13 na sintaxe padrão de regras de regressão.	64
4.15	Conjunto de regras geradas pelo RT utilizando a base de dados Auto-MPG.	65
4.16	Conjunto de regras da Figura 4.15 na sintaxe padrão de regras de regressão.	65
4.17	<i>Model Tree</i> gerada pelo M5 utilizando a base de dados Housing.	66
4.18	Algumas regras da Figura 4.17 na sintaxe padrão de regras de regressão.	67
4.19	Funcionalidades do Ambiente <i>REPPE</i> .	68
4.20	Regras da Figura 4.18 na sintaxe padrão de regras de regressão com tratamento semântico.	71
5.1	Desempenho de acordo com o número de pseudo-classes.	90
5.2	Gráfico de quatro medidas para as regras da base de dados Abalone.	94
5.3	Gráfico de quatro medidas para as regras da base de dados Auto-MPG.	95
5.4	Gráfico de quatro medidas para as regras da base de dados Housing.	96
5.5	Boxplot da base de dados Abalone.	97
5.6	Boxplot da base de dados Auto-MPG.	98
5.7	Boxplot da base de dados Housing.	99
6.1	Três razões do porque um <i>ensemble</i> pode trabalhar melhor que um preditor simples.	102
6.2	Combinação de preditores homogêneos.	104
6.3	<i>Bagging</i> — Geração paralela de preditores.	105
6.4	<i>Boosting</i> — Geração de preditores em série.	107
6.5	Combinação de preditores heterogêneos.	113
6.6	<i>Stacking</i> (Arquitetura de duas camadas) — Treinamento.	114
6.7	<i>Stacking</i> — Aplicação.	115
6.8	Esquema dos experimentos — <i>Bagging</i> .	125
6.9	Esquema dos experimentos — <i>Boosting</i> .	125
7.1	Esquema da metodologia para integração de regras de regressão.	134
7.2	Gráficos da base de dados Auto-MPG.	139
7.3	Gráficos da base de dados Housing.	140
7.4	Gráficos da base de dados Abalone.	141

7.5	Gráficos da base de dados Wisconsin Breast Cancer.	142
7.6	Gráficos da base de dados Servo.	143
7.7	Gráficos da base de dados MV.	144
7.8	Gráficos da base de dados Delta Ailerons.	145
7.9	Esquema da metodologia utilizando a seleção das melhores regras.	147

Lista de Tabelas

4.1	Conjunto de exemplos no formato atributo-valor	39
4.2	Matriz de contingência para a regra $B \rightarrow H$	41
4.3	Matriz de contingência expressa em freqüências relativas para a regra $B \rightarrow H$. .	41
4.4	Descrição da base de dados Abalone.	52
4.5	Descrição da base de dados Auto-MGP.	52
4.6	Descrição da base de dados Housing.	52
4.7	Descrição da base de dados CPU Performance.	53
4.8	Descrição da base de dados Pole.	53
4.9	Descrição da base de dados Servo.	54
4.10	Descrição da base de dados Computer Activity.	54
4.11	Descrição da base de dados Kinematics.	54
4.12	Descrição da base de dados Ailerons.	55
4.13	Descrição da base de dados Delta Ailerons.	55
4.14	Descrição da base de dados Elevators.	56
4.15	Descrição da base de dados Census.	56
4.16	Descrição da base de dados Wisconsin Breast Cancer.	56
4.17	Algoritmos de regressão utilizados.	63
5.1	Exemplos cobertos pela regra.	88
5.2	Exemplos não cobertos pela regra.	88
5.3	Limiares ϵ para a Regra [R0001].	89

5.4	Valores para as diferentes medidas de erro para regras da base de dados Abalone.	93
5.5	Valores para as diferentes medidas de erro para regras da base de dados Auto-MPG.	94
5.6	Valores para as diferentes medidas de erro para regras da base de dados Housing.	95
6.1	Resultados dos experimentos com Cubist para a base de dados Abalone.	126
6.2	Resultados dos experimentos com Cubist para a base de dados CPU Performance.	126
6.3	Resultados dos experimentos com Cubist para a base de dados Housing.	126
6.4	Resultados dos experimentos com Weka para a base de dados Abalone.	127
6.5	Resultados dos experimentos com Weka para a base de dados CPU Performance.	127
6.6	Resultados dos experimentos com Weka para a base de dados Housing.	127
7.1	Resultados para a base de dados Auto-MPG com <i>10-fold cross-validation</i> .	139
7.2	Resultados para a base de dados Auto-MPG integrando os 3 algoritmos.	139
7.3	Resultados da intersecção para a base de dados Auto-MPG.	139
7.4	Resultados para a base de dados Housing com <i>10-fold cross-validation</i> .	140
7.5	Resultados para a base de dados Housing integrando os 3 algoritmos.	140
7.6	Resultados da intersecção para a base de dados Housing.	140
7.7	Resultados para a base de dados Abalone com <i>10-fold cross-validation</i> .	141
7.8	Resultados para a base de dados Abalone integrando os 3 algoritmos.	141
7.9	Resultados da intersecção para a base de dados Abalone.	141
7.10	Resultados para a base de dados Wisconsin Breast Cancer com <i>10-fold cross-validation</i> .	142
7.11	Resultados para a base de dados Wisconsin Breast Cancer integrando os 3 algoritmos.	142
7.12	Resultados da intersecção para a base de dados Wisconsin Breast Cancer.	142
7.13	Resultados para a base de dados Servo com <i>10-fold cross-validation</i> .	143
7.14	Resultados para a base de dados Servo integrando os 3 algoritmos.	143
7.15	Resultados da intersecção para a base de dados Servo.	143

7.16 Resultados para a base de dados MV com <i>10-fold cross-validation</i>	144
7.17 Resultados para a base de dados MV integrando os 3 algoritmos.	144
7.18 Resultados da intersecção para a base de dados MV.	144
7.19 Resultados para a base de dados Delta Ailerons com <i>10-fold cross-validation</i> .	145
7.20 Resultados para a base de dados Delta Ailerons integrando os 3 algoritmos.	145
7.21 Resultados da intersecção para a base de dados Delta Ailerons.	145
7.22 Resultados para as bases de dados Servo, Auto-MPG, Housing, Abalone e Elevators.	148
7.23 Resultados para as bases de dados Computer Activity, Kinematics, Pole, Ailerons e Census.	148

Lista de Símbolos

Os símbolos freqüentemente referenciados no texto são apresentados a seguir.

- x_i é o i -ésimo exemplo do conjunto de dados.
 X_i é um atributo de entrada (variável independente).
 X é o conjunto dos atributos de entrada.
 Y é um atributo meta (variável dependente).
 y é a média do atributo meta.
 ϵ é um ruído aleatório ou limiar.
 $h(x)$ é a função que calcula o valor predito para o exemplo de treinamento com o atributo x .
 f é uma função.
 h é uma hipótese.
 A é um conjunto de atributos.
 a é o número de atributos.
 m é o número de atributos.
 E é o domínio dos atributos.
 D é um conjunto de dados.
 n é o número total de exemplos do conjunto de dados.
 S é o conjunto de treinamento.
 C é o conjunto de teste.
 I é o algoritmo de aprendizado ou indutor.
 T é o número de amostras *bootstrap*.
 w é o peso.
 P é o preditor.
 R é uma regra qualquer.
 B é quando o corpo (*body*) de uma regra assume o valor verdadeiro.
 B é quando o corpo de uma regra assume o valor falso.
 H é quando a cabeça (*head*) de uma regra assume o valor verdadeiro.
 H é quando a cabeça de uma regra assume o valor falso.
 $n(X)$ é a cardinalidade do conjunto X .
 $p(X)$ é a frequência relativa associada a X .

Introdução

Aera da informação tem desbravado caminhos que vislumbram uma nova era, a do conhecimento. A popularização da principal e maior rede de computadores do planeta, a Internet, os avanços ocorridos na coleta e armazenamento de dados, o constante desenvolvimento das empresas no que concerne a seu parque tecnológico e, principalmente, a evolução das necessidades de informação fazem com que qualquer esforço para facilitar a manipulação e o entendimento humano em relação a esse grande leque de informações ganhe importância na mesma proporção.

A preocupação cada vez maior das empresas públicas e privadas em adquirir novas tecnologias de processamento e armazenamento, além do entendimento da informação e do conhecimento como seu maior patrimônio, tem direcionado várias pesquisas para o estudo do processo de transformar em conhecimento os dados armazenados em seus sistemas, o que pode proporcionar um auxílio de forma efetivamente inteligente à tomada de decisão.

Métodos manuais para análise e interpretação de dados têm sido utilizados na transformação de dados em conhecimento. Isso, muitas vezes, torna o processo de extrair conhecimento de bases de dados caro, lento e subjetivo, além de inviável quando se trabalha com grande volume de dados. O interesse em automatizar esse processo de análise tem fomentado várias pesquisas na área de Extração de Conhecimento de Bases de Dados (*Knowledge Discovery in Databases (KDD)*) (Fayyad, Piatetsky-Shapiro, & Smith, 1996; Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996; Netz, Chaudhuri, Bernhardt, & Fayyad, 2000).

Diante desse contexto, os pesquisadores da área de Inteligência Artificial (IA) estão, cada vez mais, investindo na investigação do processo de Extração de Conhecimento de Bases de Dados com a finalidade de apresentar soluções úteis em diferentes domínios.

A extração de conhecimento a partir de grande quantidade de dados é vista como um processo interativo e iterativo, e não como um sistema de análise automática, sendo centrado na interação entre usuários, especialistas do domínio e responsáveis pela aplicação. Dessa maneira, não se pode esperar a extração de conhecimento útil simplesmente submetendo um conjunto de dados a uma “caixa preta” (Mannila, 1997).

Assim, o processo de Extração de Conhecimento de Bases de Dados, ou Mineração de Dados (MD)¹, inicia-se com o entendimento do domínio da aplicação, considerando aspectos como os objetivos dessa aplicação e as fontes de dados (base de dados da qual se pretende extrair conhecimento). Em seguida, é realizada uma seleção de dados a partir dessas fontes, de acordo com os objetivos do processo. Os conjuntos de dados resultantes dessa seleção são, então, pré-processados, ou seja, recebem um tratamento para poderem ser submetidos aos métodos e ferramentas para a extração de padrões.

A etapa de extração de padrões tem o objetivo de encontrar modelos (conhecimento) a partir de dados. Para induzir uma hipótese, utiliza-se algum algoritmo de indução (ou indutor), que recebe como entrada um conjunto de dados e gera um modelo que representa o conhecimento extraído desses dados. Após a etapa de extração de padrões, vem a de pós-processamento, na qual o conhecimento é avaliado quanto a sua qualidade e/ou utilidade a fim de ser utilizado para apoio a algum processo de tomada de decisão.

É importante notar que, por ser um processo iterativo, as etapas da Mineração de Dados não são estanques, ou seja, a correlação entre as técnicas e métodos utilizados nas várias etapas é considerável, a ponto da ocorrência de pequenas mudanças em uma delas afetar substancialmente o sucesso de todo o processo. Portanto, os resultados de uma determinada etapa podem acarretar mudanças a quaisquer das etapas posteriores ou, ainda, o recomeço de todo o processo (Fayyad, Haussler, & Stolorz, 1996; Rezende, Pugliesi, Melanda, & Paula, 2003).

Uma vez que o processo de Mineração de Dados está sendo cada vez mais utilizado para solucionar problemas da vida real, o pós-processamento dos resultados torna-se muito importante (Liu & Hsu, 1996). Assumido que o conhecimento extraído será utilizado diretamente por um Sistema Inteligente ou por algum usuário humano para inferir soluções para problemas específicos dentro de um domínio, concorda-se que a obtenção do conhecimento não é o final do processo.

Porém, pesquisas em algoritmos de aprendizado têm focado muito em técnicas para

¹Os termos Extração de Conhecimento de Bases de Dados e Mineração de Dados serão utilizados indistintamente neste trabalho.

a geração de conceitos e regras a partir de dados. Pouca ênfase vem sendo dada a pesquisas referentes ao que acontece após um conhecimento ter sido adquirido, ou seja, o pós-processamento do conhecimento, especialmente com relação à compreensibilidade, qualidade e integração do conhecimento extraído.

Uma das motivações para a realização do pós-processamento do conhecimento está no fato de que utilizar algoritmos de aprendizado em bases de dados não significa que o usuário não tenha conhecimento sobre o domínio e a base. Isso é particularmente verdade se o usuário é um especialista. Geralmente, o usuário humano tem alguma noção ou conhecimento prévio sobre o domínio de aprendizado. Assim, quando o conhecimento é gerado a partir de uma base de dados, o usuário pode querer saber: se o conhecimento representa o que ele sabe; se não representa, qual a parte do seu conhecimento prévio está ou não correta ou é interessante; ou ainda, de que maneira esse novo conhecimento difere de seu conhecimento. Anteriormente, pesquisas assumiam que era de responsabilidade do usuário analisar o conhecimento. Entretanto, quando o número de regras é grande, é muito difícil para este analisá-las manualmente.

Em se tratando de grandes bases de dados, mesmo após as etapas de seleção e preparação dos dados, pode-se continuar com uma quantidade inviável de dados para ser submetida a um algoritmo de aprendizado na busca de padrões, devido às limitações desses algoritmos. Para resolver este problema, faz-se necessária a seleção de amostras representativas desses dados (Glymour, Madigan, Pregibon, & Smyth, 1997). Mesmo tentando-se selecionar amostras representativas, o que se torna complicado em determinados domínios, acaba-se descartando parte dos dados, o que pode ocasionar a perda de conhecimento relevante, ou gerando diferentes bases de conhecimento (BC) a partir de diferentes amostras dos dados. Portanto, torna-se importante uma integração dos resultados obtidos de diferentes amostras, de modo a “aproveitar” todo o conhecimento relevante que foi extraído da base de dados. Por outro lado, com a mesma amostra, pode ser extraído conhecimento utilizando diferentes algoritmos com diferentes *bias*², e diferentes linguagens de descrição de conceitos. Este é um dos grandes problemas em aberto referente ao processo de Mineração de Dados.

Em geral, tarefas de Mineração de Dados podem ser agrupadas em duas categorias: descriptivas e preditivas. A primeira descreve o conjunto de exemplos (também chamados de casos ou dados) de uma maneira concisa e sumarizada, apresentando algumas propriedades gerais dos dados; a segunda constrói uma hipótese fazendo inferências acerca dos exemplos disponíveis, na tentativa de prever o comportamento de novos casos. Quando o objetivo é construir um modelo preditivo, existe um atributo, denominado atributo meta, que atribui a cada exemplo uma categoria, a qual pode ser discreta ou contínua. Quando

²*Bias* é qualquer critério de preferência de uma hipótese sobre outra, com exceção de consistência com os dados.

o atributo meta assume valores discretos, tem-se um problema de classificação e quando é contínuo tem-se um problema de regressão, foco deste trabalho.

Os métodos de regressão já são estudados pela comunidade estatística há bastante tempo. Porém, nas áreas de Aprendizado de Máquina e Mineração de Dados, a maioria das pesquisas é voltada para os problemas de classificação, que são mais comumente encontrados na vida real do que problemas de regressão (Weiss & Indurkhy, 1995). Recentemente, as pesquisas têm se voltado também para problemas de regressão, já que muitos problemas são desse tipo. Assim, a regressão em Mineração de Dados preditiva é uma questão pouco explorada dentro do processo de extração de conhecimento de bases de dados, sendo de grande relevância o estudo de métodos para a exploração de tarefas desse tipo.

Os métodos de regressão geram modelos³ que expressam o conhecimento obtido em diferentes formatos de representação. Dependendo dos objetivos a serem alcançados ao final do processo de Mineração de Dados, a comprehensibilidade dos modelos gerados é considerada muito importante. A necessidade de modelos capazes de fornecerem soluções interpretáveis leva à utilização dos métodos de aprendizado simbólico. Os métodos de regressão simbólicos, normalmente, induzem hipóteses no formato de regras e árvores de regressão. As regras são diferentes das árvores, e mesmo as árvores geradas por diferentes algoritmos são geralmente distintas umas das outras. Desse modo, ao submeter um conjunto de dados a vários algoritmos de regressão, diferentes tipos de regressores são obtidos. Com isso, torna-se difícil avaliar o conhecimento extraído por diferentes algoritmos porque a avaliação precisa ser feita de maneira diferente para cada algoritmo, uma vez que cada modelo é representado de uma forma diferenciada. Essa avaliação dos modelos simbólicos extraídos faz parte do pós-processamento do conhecimento.

O primeiro ponto que deve ser avaliado na etapa de pós-processamento é o desempenho da predição (ou precisão) do modelo frente a novos exemplos. Além da precisão, existem ainda outros fatores que devem ser avaliados, como a comprehensibilidade, que caracteriza o quanto compreensível esse modelo é ao usuário, e o grau de interesse, que consiste em verificar se o conhecimento obtido por esse modelo é novo ou não. Além disso, no caso de se tratar de um novo conhecimento, deve-se verificar se ele é interessante e útil para os usuários do processo de Mineração de Dados.

Alguns trabalhos vêm sendo realizados no Laboratório de Inteligência Computacional (LABIC)⁴ do Instituto de Ciências Matemáticas e de Computação (ICMC) da USP São Carlos em temas relacionados ao processo de Extração de Conhecimento de Bases de Dados e Textos e na construção de um ambiente computacional para extração de conhecimento, denominado DISCOVER.

³Os modelos de regressão são também conhecidos como regressores.

⁴<http://labic.icmc.usp.br/>

O objetivo do DISCOVER consiste em fornecer um ambiente integrado para apoiar as etapas do processo de descoberta do conhecimento, oferecendo funcionalidades voltadas para Aprendizado de Máquina e Mineração de Dados e de Textos. Essas funcionalidades são baseadas em sintaxes padrões utilizadas para representar dados e conhecimento.

1.1 *Objetivos e Principais Contribuições*

A extração de padrões e modelos de grandes bases de dados é feita utilizando-se várias metodologias, técnicas e algoritmos de aprendizado. Entretanto, esses algoritmos trabalham bem em conjuntos relativamente pequenos de dados, de forma que o seu uso deve ser adaptado para a nova realidade de grandes bases de dados. Além disso, outros aspectos devem ser considerados, como: (i) estudo de métodos para avaliação da precisão e qualidade do conhecimento obtido a partir de uma ou mais amostras aplicadas a um ou vários algoritmos de aprendizado; (ii) avaliação do conhecimento extraído a partir da utilização de algoritmos de combinação de regressores usados para melhorar a precisão dos resultados, ou seja, análise do conhecimento contido na “caixa preta”; (iii) descoberta de conhecimento que não seja apenas correto, mas que também seja interpretável e traga novidades para os usuários; e (iv) pesquisa de métodos para integração de conhecimento a partir de diferentes amostras e algoritmos.

Um ponto que merece ser destacado é a necessidade, em alguns casos, de se compreender o porquê de um determinado valor ter sido escolhido frente aos demais, como por exemplo, quando da aprovação ou não de uma apólice de seguros, na qual o cliente tem por direito legal uma explicação. Portanto, não basta utilizar o regressor como uma “caixa preta”, tendo-se a necessidade de “abri-la” e investigar o seu conteúdo.

Assim, esta tese tem como objetivo principal propor e avaliar técnicas para pós-processamento em Mineração de Dados para problemas de regressão. Para tanto nesta tese é abordada a avaliação do conhecimento extraído, a combinação de regressores e a integração de conhecimento.

As principais contribuições são: i) disponibilização de diferentes métricas para avaliar o conhecimento de problemas de regressão, inclusive com proposta e implementação de diferentes estratégias para o cálculo dos valores da matriz de contingência para regras de regressão; ii) exploração de métodos e algoritmos para combinação de regressores, visando melhorar a precisão dos modelos; iii) proposta de técnicas para integrar o conhecimento obtido e métricas que avaliem o desempenho da integração do conhecimento obtido com diferentes métodos e diferentes amostras; e iv) viabilização dos métodos para pós-processamento de problemas de regressão no Ambiente DISCOVER. Ressalta-se que essas contribuições foram disponibilizadas no Ambiente *REPPE* (*Regression Post-Processing*

Environment) aqui proposto e desenvolvido.

Como citado, a maioria das pesquisas realizadas nas áreas de Aprendizado de Máquina e Mineração de Dados são voltadas para problemas de classificação. No DISCOVER, a situação não é muito diferente. A maioria dos trabalhos já desenvolvidos são voltados para problemas desse tipo. Assim, o Ambiente *REPPE*, que é um ambiente computacional para pós-processamento de regras de regressão, incrementa a potencialidade do DISCOVER com relação à disponibilização de funcionalidades para tratar a tarefa de regressão.

1.2 Organização do Trabalho

Esta tese está organizada da seguinte maneira. Neste capítulo foi apresentado o contexto em que se insere esta tese, bem como os objetivos e algumas contribuições da mesma.

No Capítulo 2, é apresentada uma revisão do processo de Mineração de Dados, detalhando as principais etapas desse processo.

No Capítulo 3, são definidos os problemas de regressão, apresentando as definições, formalizações e métodos existentes para tratar tarefas de regressão.

No Capítulo 4, é apresentado o Ambiente *REPPE* e suas funcionalidades, e como ele está integrado ao Ambiente DISCOVER. Além disso, também são relatados todos os trabalhos desenvolvidos e em desenvolvimento do DISCOVER. São também apresentadas as sintaxes padrão definidas e utilizadas no *REPPE*, bem como as bases de dados (BD) e os algoritmos utilizados na realização dos experimentos.

As medidas de avaliação, tanto as que utilizam a matriz de contingência para o cálculo quanto as que não utilizam, são apresentadas no Capítulo 5. Nesse capítulo, também são apresentadas as estratégias propostas para o cálculo dos valores da matriz de contingência para regras de regressão. Além disso, são realizados experimentos para validação das estratégias propostas.

No Capítulo 6, são relatadas as técnicas para combinação de preditores, tanto homogêneos quanto heterogêneos. Ainda são apresentados os algoritmos implementados para combinação de regressores, bem como os experimentos realizados.

No Capítulo 7, são apresentadas as abordagens utilizadas na integração de conhecimento para problemas de regressão, tanto os que visam a seleção das melhores regras para formar a nova base de conhecimento, quanto os que eliminam as piores regras (poda de regras). Experimentos foram realizados nas duas linhas em várias base de dados.

No Capítulo 8, são apresentadas as conclusões desta tese, as principais contribuições e limitações, e as propostas de trabalhos futuros.

Por fim, são apresentadas as referências bibliográficas utilizadas nesta tese.

Mineração de Dados

A evolução da computação possibilitou um aumento na capacidade de processamento e armazenamento de dados. A facilidade atual que uma aplicação científica e/ou comercial possui para gerar *gigabytes* ou *terabytes* de dados em poucas horas excede em muito a capacidade de pesquisadores e analistas de mercado em fazer análises sobre os mesmos.

Os aplicativos essencialmente utilizados para consultas (por exemplo, planilhas eletrônicas) foram projetados para gerar relatórios simplificados de atividades como: listar o número de itens vendidos no dia, verificar estoques de vendas e desenhar gráficos da evolução das vendas obtidas. Consultas dessa natureza são capazes de satisfazer as necessidades rotineiras de uma empresa.

Entretanto, atualmente os analistas de negócios precisam usar ferramentas capazes de responder a perguntas complexas como: “qual produto de alta lucratividade venderia mais com a promoção de um item de baixa lucratividade, analisando os dados dos últimos dez anos de vendas?”. Esse tipo de informação pode ser fundamental para a sobrevivência de uma empresa nos tempos de hoje. Dessa maneira, novas ferramentas de análise e extração de conhecimento devem ser usadas no processo decisório.

O *Data Warehousing* (DW) (Gardner, 1998) é considerado um dos primeiros passos para tornar factível a análise de grande quantidade de dados no apoio ao processo decisório. O objetivo básico é criar um repositório, conhecido por *Data Warehouse*, que contenha dados limpos, agregados e consolidados que possam ser analisados por ferramentas OLAP (*On-Line Analytical Processing*). Essas ferramentas apresentam facilidades para a

realização de consultas complexas em bases de dados multidimensionais. As ferramentas utilizadas para analisar um *Data Warehouse*, normalmente, são orientadas a consultas, ou seja, são dirigidas pelos usuários, os quais possuem hipóteses que gostariam de comprovar ou, simplesmente, executam consultas aleatórias. Essa abordagem dependente do usuário pode impedir que padrões escondidos nos dados sejam encontrados de forma “inteligente”, uma vez que o usuário não terá condições de imaginar todas as possíveis relações e associações existentes em grande volume de dados. Por isso, faz-se necessária a utilização de técnicas de análise dirigidas por computador que possibilitem a extração automática (ou semi-automática) de novos conhecimentos a partir de um grande repositório de dados (Bradley, Fayyad, & Mangasarian, 1998).

Diante da deficiência para analisar e compreender grande volume de dados, diversos estudos têm sido direcionados ao desenvolvimento de tecnologias de extração automática de conhecimento de bases de dados. Esse campo de pesquisa é chamado de Extração de Conhecimento de Base de Dados, geralmente referenciado na literatura como *Knowledge Discovery in Database* (KDD), *Data Mining* (DM) ou Mineração de Dados (MD). Alguns autores consideram os termos KDD e MD distintos, sendo MD uma fase no processo do KDD (Fayyad, Piatetsky-Shapiro, & Smyth, 1996b). Entretanto, neste trabalho, os termos KDD e MD serão tratados indistintamente referenciando o processo de extrair conhecimento a partir de dados.

O processo de Extração de Conhecimento de Bases de Dados tem o objetivo de encontrar conhecimento a partir de um conjunto de dados para ser utilizado em um processo decisório. Portanto, um requisito importante é que esse conhecimento descoberto seja comprehensível por humanos, além de útil e interessante para os usuários finais do processo, de forma que forneça suporte a esses usuários na tomada de decisão (Fayyad, Piatetsky-Shapiro, & Smyth, 1996a; Freitas, 1998b). A Mineração de Dados é uma área multidisciplinar que incorpora técnicas utilizadas em diversas áreas como Base de Dados, Inteligência Artificial e Estatística. Por isso, as técnicas utilizadas em MD não devem ser vistas como substitutas de outras formas de análises (por exemplo, OLAP), mas como práticas para melhorar os resultados das explorações feitas com as ferramentas atualmente usadas.

A definição aceita por diversos pesquisadores de MD foi elaborada por Fayyad, Piatetsky-Shapiro, & Smyth (1996a) como sendo: “Extração de Conhecimento de Base de Dados é o processo de identificação de padrões válidos, novos, potencialmente úteis e comprehensíveis embutidos nos dados”. Para compreender melhor o conteúdo dessa definição, deve-se olhar individualmente cada componente da mesma:

Dados Conjunto de fatos ou casos em um repositório de dados. Por exemplo, os dados correspondem aos valores dos campos de um registro de vendas em uma base de

dados qualquer;

Padrões Denota alguma abstração de um subconjunto dos dados em alguma linguagem descritiva de conceitos;

Processo A Extração de Conhecimento de Base de Dados envolve diversas etapas como a preparação dos dados, busca por padrões e avaliação do conhecimento;

Válidos Os padrões descobertos devem possuir algum grau de certeza, ou seja, devem satisfazer funções ou limiares que garantam que os exemplos cobertos e os casos relacionados ao padrão encontrado sejam aceitáveis;

Novos Um padrão encontrado deve fornecer novas informações sobre os dados. O grau de novidade serve para determinar quão novo ou inédito é um padrão. Pode ser medido por meio de comparações entre as mudanças ocorridas nos dados ou no conhecimento anterior;

Úteis Os padrões descobertos devem ser incorporados e utilizados;

Compreensíveis Um dos objetivos de realizar MD é encontrar padrões descritos em alguma linguagem que possa ser compreendida pelos usuários permitindo uma análise mais profunda dos dados;

Conhecimento O conhecimento é definido em termos dependentes do domínio que estão relacionados fortemente com medidas de utilidade, originalidade e compreensão.

Todo o processo de MD é orientado em função de seu domínio de aplicação e dos repositórios de dados inerentes aos mesmos. Para usar os dados é necessário que eles estejam estruturados de forma a serem consultados e analisados adequadamente.

2.1 *O Processo de Mineração de Dados*

Existem diversas abordagens para a divisão das etapas do processo de Extração de Conhecimento de Bases de Dados. Inicialmente, foi proposto em Fayyad (1996) uma divisão do processo em nove etapas. Já em Weiss & Indurkhy (1998), essa divisão é composta por apenas quatro etapas. No entanto, neste capítulo é considerada a divisão do processo em três grandes etapas: pré-processamento, extração de padrões e pós-processamento (Rezende, Puggiesi, Melanda, & Paula, 2003). Foram incluídas nessa divisão uma fase anterior ao processo de Mineração de Dados, que se refere à identificação do problema e conhecimento do domínio, e uma fase posterior ao processo, que se refere à utilização do conhecimento obtido. Observa-se que normalmente esse processo é iterativo. A Figura 2.1 ilustra essas etapas.



Figura 2.1: Etapas do processo de Mineração de Dados (Rezende, Pugliesi, Melanda, & Paula, 2003).

O processo de MD é centrado na interação entre as diversas classes de usuários, e o seu sucesso depende, em parte, dessa interação. Os usuários do processo podem ser divididos em três classes: especialista do domínio, que deve possuir amplo conhecimento do domínio da aplicação e deve fornecer apoio para a execução do processo; analista, que deve conhecer profundamente as etapas que compõem o processo de Mineração de Dados e ser responsável por sua execução; e usuário final, que representa a classe de usuários que utiliza o conhecimento extraído no processo para auxiliá-lo em um processo de tomada de decisão.

É importante ressaltar que pode haver situações em que o especialista do domínio também é o usuário final, ou que este auxilie ou execute funções pertinentes ao analista. Entretanto, é pouco provável que o analista encontre conhecimento útil a partir dos dados sem a opinião do especialista sobre o que é considerado interessante em um domínio de aplicação específico.

2.1.1 Identificação do Problema

O estudo do domínio da aplicação e a definição de objetivos e metas a serem alcançadas no processo de Mineração de Dados são identificados nesta etapa.

O sucesso do processo de Mineração de Dados depende, em parte, da participação dos especialistas do domínio da aplicação no fornecimento de conhecimento sobre o domínio e apoio aos analistas em sua tarefa de encontrar os padrões. Assim, antes do início das tarefas do processo é imprescindível a realização de um estudo a fim de adquirir um conhecimento inicial do domínio (Fayyad, Piatetsky-Shapiro, & Smyth, 1996b).

Algumas questões importantes devem ser respondidas nessa etapa de identificação do problema, como:

- Quais são as principais metas do processo?
- Quais critérios de desempenho são importantes?
- O conhecimento extraído deve ser compreensível a seres humanos ou um modelo do tipo “caixa preta” é apropriado?
- Qual deve ser a relação entre simplicidade e precisão do conhecimento extraído?

Além dessa análise inicial para definição das principais metas, objetivos e restrições, o conhecimento sobre o domínio fornece um subsídio para todas as etapas do processo de Mineração de Dados. Mais especificamente, na etapa de pré-processamento, esse conhecimento pode ajudar os analistas na escolha do melhor conjunto de dados para realizar a extração de padrões, e saber os valores válidos para os atributos, os critérios de preferência entre os possíveis atributos, as restrições de relacionamento ou as informações para geração de novos atributos.

Na etapa de Extração de Padrões, o conhecimento sobre o domínio pode ajudar os analistas na escolha de um critério de preferência entre os modelos gerados, no ajuste dos parâmetros do processo de indução, ou mesmo na geração de um conhecimento inicial a ser fornecido como entrada do algoritmo de mineração para aumentar a eficiência no aprendizado dos conceitos e melhorar a precisão ou a compreensibilidade do modelo final.

Na etapa de pós-processamento, o conhecimento extraído pelos algoritmos de Extração de Padrões deve ser avaliado. Alguns critérios de avaliação utilizam o conhecimento do especialista para saber, por exemplo, se o conhecimento extraído é interessante ao usuário (Piatetsky-Shapiro & Matheus, 1994; Liu & Hsu, 1996; Hilderman & Hamilton, 2001).

Entender o domínio dos dados é naturalmente um pré-requisito para extrair algo útil: o usuário final deve ter algum grau de entendimento sobre a área de aplicação antes

de qualquer informação valiosa ser obtida. Por outro lado, se existem especialistas com profundo conhecimento sobre domínio, a obtenção de um novo conhecimento com o uso de ferramentas semi-automáticas se torna mais difícil. Esse pode ser o caso em domínios bastante estáveis, no qual os seres humanos tiveram tempo para adquirir o conhecimento especializado em detalhes. Um exemplo ocorre em algumas áreas de venda em que os produtos e os clientes são os mesmos por um longo período de tempo. As áreas mais promissoras para Mineração de Dados parecem ser aquelas nas quais as propriedades reais dos dados mudam, e existem especialistas com conhecimentos genéricos abrangentes, como no caso da área de telecomunicações, na qual os operadores das redes têm uma idéia geral das características dos sistemas. Porém, mudanças em equipamentos e softwares implicam que novos especialistas sejam encontrados, o que pode ser um tarefa difícil.

2.1.2 Pré-Processamento

Normalmente, os dados disponíveis para análise não estão em um formato adequado para a Extração de Conhecimento. Além disso, em razão de limitações de memória ou tempo de processamento, muitas vezes não é possível a aplicação direta dos algoritmos de extração de padrões aos dados. Dessa maneira, torna-se necessária a aplicação de métodos para tratamento, limpeza e redução do volume de dados antes de iniciar a etapa de Extração de Padrões. É importante salientar que a execução das transformações deve ser guiada pelos objetivos do processo de extração a fim de que o conjunto de dados gerado apresente as características necessárias para que os objetivos sejam cumpridos (Batista, 2003)

Diversas transformações nos dados podem ser executadas na etapa de pré-processamento, entre elas: Extração e Integração, Transformação, Limpeza, Seleção e Redução de Dados.

Os dados disponíveis podem estar em diferentes formatos, como arquivos-texto, arquivos no formato de planilhas, Banco de Dados ou *Data Warehouse*. Assim, é necessária a obtenção desses dados e sua unificação, formando uma única fonte de dados no formato atributo-valor, que será utilizada como entrada para o algoritmo de Extração de Padrões.

Após a extração e integração dos dados, estes devem ser adequados para serem utilizados nos algoritmos de extração de padrões. Algumas transformações comuns que podem ser aplicadas aos dados são: resumo, por exemplo, quando dados sobre vendas são agrupados para formar resumos diários; transformação de tipo, por exemplo, quando um atributo do tipo data é transformado em um outro tipo para que o algoritmo de Extração de Padrões possa utilizá-lo mais adequadamente; e normalização de atributos contínuos, colocando seus valores em intervalos definidos, por exemplo, entre 0 e 1. As transformações de dados são extremamente importantes em alguns domínios, por exemplo, em

aplicações que envolvem séries temporais como previsões no mercado financeiro.

Os dados disponíveis para aplicação dos algoritmos de Extração de Padrões podem apresentar problemas advindos do processo de coleta. Esses problemas podem ser erros de digitação ou erro na leitura dos dados pelos sensores. Como o resultado do processo de extração possivelmente será utilizado em um processo de tomada de decisão, a qualidade dos dados é um fator extremamente importante. Por isso, técnicas de limpeza devem ser aplicadas aos dados a fim de garantir sua qualidade.

A limpeza dos dados pode ser realizada utilizando o conhecimento do domínio. Por exemplo, pode-se encontrar registros com valor inválido em algum atributo, granularidade incorreta ou exemplos errôneos. Pode-se também efetuar alguma limpeza independente de domínio, como decisão da estratégia de tratamento de atributos incompletos, remoção de ruído e tratamento de conjunto de exemplos não balanceados (Batista, Carvalho, & Monard, 2000).

O número de exemplos e de atributos disponíveis para análise pode inviabilizar a utilização de algoritmos de Extração de Padrões. Como solução para este problema, pode ser necessária a aplicação de métodos para redução dos dados antes de iniciar a busca pelos padrões. Essa redução pode ser feita de três maneiras (Weiss & Indurkha, 1998):

- redução do número de exemplos;
- redução do número de atributos;
- redução do número de valores de um atributo.

A redução do número de exemplos deve ser feita a fim de manter as características do conjunto de dados original, isto é, por meio da geração de amostras representativas dos dados (Glymour, Madigan, Pregibon, & Smyth, 1997). A abordagem mais utilizada para redução do número de exemplos é a amostragem aleatória (Weiss & Indurkha, 1998), pois este método tende a produzir amostras representativas.

Se a amostra não for representativa, ou se a quantidade de exemplos for insuficiente para caracterizar os padrões embutidos nos dados, os modelos encontrados podem não representar a realidade, não tendo, portanto, valor. Além disso, com uma quantidade relativamente pequena de exemplos, pode ocorrer *overfitting*, isto é, o modelo gerado pode “decorar” os dados do conjunto de treinamento, não se adequando aos novos exemplos (Fayyad, 1996).

A redução do número de atributos pode ser utilizada para reduzir o espaço de busca pela solução. O objetivo é selecionar um subconjunto dos atributos existentes de forma que isto não tenha grande impacto na qualidade da solução final. Essa redução pode ser realizada com o apoio do especialista do domínio, uma vez que, ao remover um atributo

potencialmente útil para o modelo final, a qualidade do conhecimento extraído pode diminuir consideravelmente. Além disso, por não se saber inicialmente quais atributos serão importantes para atingir os objetivos, deve-se remover somente aqueles atributos que, com certeza, não têm nenhuma importância para o modelo final.

Outra maneira de reduzir o número de atributos pode ser por meio de indução construtiva, na qual um novo atributo é criado a partir do valor de outros. Caso os atributos originais utilizados na construção do novo atributo não estejam presentes em um novo modelo, eles podem ser descartados, reduzindo assim o número de atributos. A utilização de indução construtiva pode aumentar consideravelmente a qualidade do conhecimento extraído (Lee, 2000).

A terceira forma de redução dos dados consiste na redução do número de valores de um atributo. Isso é feito, geralmente, por discretização ou suavização dos valores de um atributo contínuo.

Discretização de um atributo consiste na substituição de um atributo contínuo (inteiro ou real) por um atributo discreto, por meio do agrupamento de seus valores. Essencialmente, um algoritmo de discretização aceita como entrada os valores de um atributo contínuo e gera como saída uma pequena lista de intervalos ordenados. Cada intervalo é representado na forma $[V_{in\acute{e}rior} : V_{superior}]$, de modo que $V_{in\acute{e}rior}$ e $V_{superior}$ são, respectivamente, os limites inferior e superior do intervalo. Os métodos de discretização podem ser classificados em supervisionados ou não-supervisionados, locais ou globais, e parametrizados ou não-parametrizados (Félix, Rezende, Monard, & Caulkins, 2000).

Na suavização dos valores de um atributo, o objetivo é diminuir o número de valores do mesmo sem discretizá-lo. Nesse método, os valores de um determinado atributo são agrupados mas, ao contrário da discretização, cada grupo de valores é substituído por um valor numérico que o represente. Esse novo valor pode ser a média, a mediana ou mesmo os valores de borda de cada grupo (Weiss & Indurkhy, 1998).

As transformações descritas devem ser realizadas criteriosamente e com o devido cuidado, uma vez que é fundamental garantir que as informações presentes nos dados brutos continuem presentes nas amostras geradas, para que os modelos finais representem a realidade dos dados brutos.

A etapa de pré-processamento é realizada antes da Extração de Padrões. Mas, como Mineração de Dados é um processo iterativo, algumas atividades de pré-processamento podem ser realizadas novamente após a análise dos padrões encontrados na etapa de Extração de Padrões. Por exemplo, pode-se desejar acrescentar algum atributo, reduzir ou aumentar o volume de dados, fazer uma outra transformação no tipo de algum atributo, para que o indutor possa utilizá-lo de forma mais eficiente, e melhorar a qualidade do conhecimento extraído.

2.1.3 Extração de Padrões

A etapa de Extração de Padrões é direcionada ao cumprimento dos objetivos definidos na Identificação do Problema. Nesta etapa é realizada a escolha, a configuração e a execução de um ou mais algoritmos para extração de conhecimento. Como esta etapa é iterativa, pode ser necessário que ela seja executada diversas vezes para ajustar o conjunto de parâmetros visando a obtenção de resultados mais adequados aos objetivos preestabelecidos. Ajustes podem ser necessários, por exemplo, para a melhoria da precisão ou da comprehensibilidade do conhecimento extraído.

A etapa de Extração de Padrões compreende a escolha da tarefa de Mineração de Dados a ser empregada, a escolha do algoritmo e a extração dos padrões propriamente dita.

Escolha da Tarefa

Com o grande número de sistemas de Mineração de Dados desenvolvidos para os mais diferentes domínios, a variedade de tarefas para MD vem se tornando cada vez mais diversificada. Essas tarefas podem extrair diferentes tipos de conhecimento, sendo necessário decidir já no início do processo de MD qual o tipo de conhecimento que o algoritmo deve extrair. A escolha da tarefa é feita de acordo com os objetivos desejáveis para a solução a ser encontrada. As tarefas possíveis de um algoritmo de Extração de Padrões podem ser agrupadas em atividades preditivas e descritivas, como ilustrado na Figura 2.2.

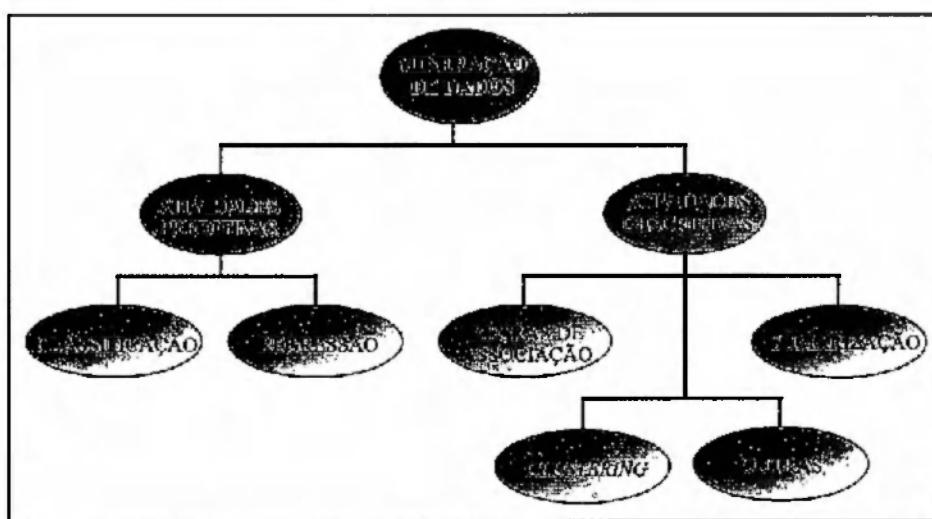


Figura 2.2: Tarefas de Mineração de Dados.

Atividades de predição, ou Mineração de Dados preditivo, consistem na generalização de exemplos ou experiências passadas com respostas conhecidas em uma linguagem capaz

de reconhecer a classe de um novo exemplo. Os dois principais tipos de tarefas para predição são classificação e regressão. A classificação consiste na predição de um valor categórico como, por exemplo, predizer se o cliente é bom ou mau pagador. Na regressão, o atributo a ser predito consiste em um valor contínuo como, por exemplo, predizer o lucro ou a perda em um empréstimo.

A tarefa de classificação é uma função de aprendizado que mapeia dados de entrada, ou conjuntos de dados de entrada, em um número finito de categorias. Nela, cada exemplo pertence a uma classe, entre um conjunto predefinido de classes. Os exemplos consistem de um conjunto de atributos e um atributo-classe discreto. O objetivo de um algoritmo de classificação é encontrar algum relacionamento entre os atributos e uma classe, de modo que o processo de classificação possa usar esse relacionamento para predizer a classe de um exemplo novo e desconhecido.

A tarefa de regressão é conceitualmente similar à de classificação. A principal diferença é que o atributo a ser predito é contínuo em vez de discreto.

Atividades de descrição, ou Mineração de Dados descritivo, consistem na identificação de comportamentos intrínsecos do conjunto de dados, sendo que estes dados não possuem uma classe especificada. Algumas das tarefas de descrição são *clustering*, regras de associação e sumarização.

Clustering procura identificar um conjunto finito de *clusters*, ou agrupamentos, a partir dos dados. Isso é feito, geralmente, de maneira que exemplos com valores de atributos similares são agrupados em um mesmo *cluster*. De forma geral, uma regra de associação caracteriza o quanto a presença de um conjunto de atributos nos registros de uma base de dados implica a presença de algum outro conjunto distinto de atributos nos mesmos registros (Agrawal & Srikant, 1994). Já a sumarização envolve métodos para encontrar uma descrição compacta para um subconjunto de dados. Um exemplo de sumarização é a definição da média e do desvio padrão de todos os campos. Os métodos mais sofisticados envolvem a derivação de regras resumidas e técnicas de visualização (Fayyad, 1996).

As atividades de predição envolvem o uso dos atributos de um conjunto de dados para prever o valor futuro da variável meta, ou seja, essas atividades visam principalmente a tomada de decisões. Já as atividades de descrição procuram padrões interpretáveis pelos humanos que descrevem os dados antes de realizar a previsão. Essa tarefa visa o suporte à decisão.

Uma vez eleita a tarefa a ser empregada, existe uma variedade de algoritmos para executá-la. A seleção do algoritmo de extração e a posterior configuração de seus parâmetros também são realizadas nesta etapa.

Escolha do Algoritmo

A escolha do algoritmo é realizada de forma subordinada à linguagem de representação dos padrões a serem encontrados. Podem-se utilizar algoritmos indutores de árvores de decisão ou regras de produção, por exemplo, se o objetivo é realizar uma predição. Entre os tipos mais freqüentes de representação de padrões, destacam-se: árvores de decisão, regras de produção, modelos lineares, modelos não-lineares (Redes Neurais Artificiais), modelos baseados em exemplos (*K-Nearest Neighbor* (KNN), Raciocínio Baseado em Casos) e modelos de dependência probabilística (Redes Bayesianas).

Um aspecto que merece atenção é a complexidade da solução encontrada pelo algoritmo de extração. A complexidade da solução está diretamente relacionada com a capacidade de representação do conceito embutido nos dados. O problema é que, quando os parâmetros do algoritmo estão ajustados para encontrar soluções mais complexas que o conceito efetivamente existente nos dados, o modelo resultante poderá ter uma precisão boa para o conjunto de treinamento, mas um desempenho ruim para novos exemplos. Diz-se então que o modelo é específico para o conjunto de treinamento, ocorrendo *overfitting* (Mitchell, 1997).

Por outro lado, se a solução que está sendo buscada não for suficiente para adequar o conceito representado nos dados, por exemplo, ao se definir um número insuficiente de neurônios para uma Rede Neural ou um fator de poda muito alto na geração de uma árvore de decisão, o modelo induzido pode não ser representativo. Isto é o que se chama de *underfitting* (Mitchell, 1997). Nesse caso, é provável que o modelo encontrado não tenha bom desempenho nem nos exemplos disponíveis para o treinamento nem para novos exemplos.

Portanto, a configuração dos parâmetros do algoritmo deve ser feita de forma bastante criteriosa. Em (Kearns & Vazirani, 1994) é fornecida uma sugestão para escolher uma determinada função: o modelo mais apropriado é aquele mais simples que seja consistente com todas as observações. Normalmente, soluções mais complexas são preferidas por pesquisadores, enquanto os práticos tendem a preferir modelos mais simples em virtude de sua fácil interpretação (Fayyad, 1996).

Em (Kohavi, Sommerfield, & Dougherty, 1996) é mostrado experimentalmente que não existe um único bom algoritmo para todas as tarefas de Mineração de Dados. Por isso, a escolha de vários algoritmos para realizar a tarefa desejada pode ser feita, levando a obtenção de diversos modelos que, na etapa de pós-processamento, são tratados para fornecer o conjunto de padrões mais adequado ao usuário final.

Extração dos Padrões

A extração dos padrões propriamente dita consiste da aplicação dos algoritmos de mineração escolhidos para a Extração dos Padrões embutidos nos dados. É importante ressaltar que dependendo da função escolhida pode ser necessária a execução desses algoritmos diversas vezes.

Técnicas de combinação de preditores têm sido pesquisadas com o objetivo de construir um preditor mais preciso pela combinação de vários outros. O resultado dessa combinação é chamado de *ensemble*. A utilização de *ensembles* tem obtido melhores resultados que a utilização de um único preditor (Dietterich, 2000a; Breiman, 2000b).

Estas técnicas podem ser divididas em combinação de preditores homogêneos ou heterogêneos. Na combinação de preditores homogêneos, um mesmo algoritmo de aprendizado é aplicado a diversas amostras de dados. Na combinação de preditores heterogêneos, uma mesma amostra de dados é submetida a diversos algoritmos de aprendizado.

Outro aspecto importante é que com o objetivo de fazer uma avaliação mais precisa da taxa de erro de um preditor, métodos de *resampling* têm sido normalmente utilizados na extração dos padrões.

A disponibilização do conjunto de padrões extraídos nesta etapa ao usuário ou a sua incorporação a um Sistema Inteligente ocorre após a análise e/ou o processamento dos padrões na etapa de pós-processamento.

2.1.4 Pós-Processamento

A obtenção do conhecimento não é o passo final do processo de Extração de Conhecimento de Bases de Dados. O conhecimento extraído pode ser utilizado na resolução de problemas da vida real, seja por meio de um Sistema Inteligente ou de um ser humano como apoio a algum processo de tomada de decisão. Para isso é importante que algumas questões sejam respondidas aos usuários (Liu & Hsu, 1996):

- O conhecimento extraído representa o conhecimento do especialista?
- De que maneira o conhecimento do especialista difere do conhecimento extraído?
- Em que parte o conhecimento do especialista está correto?

Os algoritmos utilizados para a extração dos padrões podem gerar uma quantidade enorme de padrões, muitos dos quais podem não ser importantes, relevantes ou interessantes para o usuário. Fornecer ao usuário uma grande quantidade de padrões descobertos não é produtivo pois, normalmente, ele procura uma pequena lista de padrões interessantes. Portanto, é de vital importância desenvolver algumas técnicas de apoio no sentido

de fornecer aos usuários apenas os padrões mais interessantes (Silberschatz & Tuzhilin, 1995).

Diversas medidas para avaliação de conhecimento têm sido pesquisadas com a finalidade de auxiliar o usuário no entendimento e na utilização do conhecimento adquirido. Estas medidas podem ser divididas em medidas de desempenho e medidas de qualidade.

Desempenho pode ser visto como um conjunto de características ou de possibilidades de atuação de um processo. Algumas medidas de desempenho são precisão, erro, confiança negativa, sensitividade, especificidade, cobertura, suporte, satisfação, velocidade, tempo de aprendizado, etc. (Lavrac, Flach, & Zupan, 1999).

Já a qualidade é uma medida que permite avaliar e, consequentemente, aprovar, aceitar ou recusar o conhecimento obtido, visto que um dos objetivos principais do processo de Mineração de Dados é que o usuário possa compreender e utilizar o conhecimento descoberto. Entretanto, podem ocorrer casos em que os modelos são muito complexos ou não fazem sentido para os especialistas (Pazzani, 2000a; Pazzani, Mani, & Shankle, 1997). Assim, a comprehensibilidade do conhecimento extraído é um aspecto bastante importante para o processo de Mineração de Dados.

A comprehensibilidade de um dado conjunto de regras está relacionada com a facilidade de interpretação dessas regras por um ser humano. A comprehensibilidade de um modelo pode ser estimada, por exemplo, pelo número de regras e o número de condições por regra. Nesse caso, quanto menor a quantidade de regras de um dado modelo e menor o número de condições por regra, maior será a comprehensibilidade das regras descobertas (Fertig, Freitas, Arruda, & Kaestner, 1999). Em (Pazzani, 2000a; Pazzani, Mani, & Shankle, 1997) é discutido que outros fatores, além do tamanho do modelo, são importantes na determinação da comprehensibilidade de um conhecimento. Um fator citado é que os usuários especialistas possuem tendência a compreender melhor modelos que não contradizem seu conhecimento prévio.

O grau de interesse é uma maneira de avaliar a qualidade tentando estimar o quanto de conhecimento interessante (ou inesperado) existe e deve combinar fatores numa medida que reflete como o especialista julga o padrão (Piatetsky-Shapiro & Matheus, 1994). As medidas de grau de interesse estão baseadas em vários aspectos, principalmente na utilidade que as regras representam para o usuário final do processo de Extração de Conhecimento (Dong & Li, 1998). Estas medidas podem ser divididas em objetivas e subjetivas (Silberschatz & Tuzhilin, 1995; Piatetsky-Shapiro & Matheus, 1994; Freitas, 1998a).

Medidas objetivas são aquelas que estão relacionadas somente com a estrutura dos padrões e do conjunto de dados. Elas não levam em consideração fatores específicos do usuário nem do conhecimento do domínio para avaliar um padrão. Algumas medidas

objetivas de grau de interesse são: modelos de regras, cobertura de regras mínimas, custo da classificação incorreta e tamanho do disjunto.

Como diferentes usuários finais do processo de Extração de Conhecimento podem ter diferentes graus de interesse para um determinado padrão, medidas subjetivas são necessárias. Essas medidas consideram que fatores específicos do conhecimento do domínio e de interesse do usuário devem ser tratados ao selecionar um conjunto de regras interessantes ao usuário. Algumas medidas subjetivas são inesperabilidade e utilidade (Silberschatz & Tuzhilin, 1995).

Em um ambiente para avaliação de conhecimento, aspectos objetivos do grau de interesse podem ser utilizados como um primeiro filtro para selecionar regras potencialmente interessantes ao usuário. Os aspectos subjetivos podem ser utilizados como um filtro final para selecionar regras realmente interessantes.

Após a análise do conhecimento, caso este não seja de interesse do usuário final ou não cumpra com os objetivos propostos, o processo de extração pode ser repetido ajustando-se os parâmetros ou melhorando o processo de escolha dos dados para a obtenção de resultados melhores em uma próxima iteração.

2.2 *Técnicas e Ferramentas Usadas em Mineração de Dados*

Uma parte importante de MD são as técnicas e os algoritmos empregados na Mineração de Dados. São várias as técnicas que podem ser usadas nesse processo, entre elas Análise Estatística, Aprendizado de Máquina, Algoritmos Genéticos e Redes Neurais.

É importante notar que as técnicas descrevem um paradigma de extração de conhecimento e vários algoritmos podem seguir esse paradigma. Por exemplo, o aprendizado simbólico que gera regras de decisão, muito utilizadas para extração de conhecimento, pode ser realizado utilizando diferentes algoritmos, como *C4.5 rules* (Quinlan, 1993) e *CN2* (Clark & Niblett, 1989).

Entre as técnicas mais utilizadas em Mineração de Dados estão as regras e árvores de decisão, as Redes Neurais que apesar de não gerarem conhecimento explícito são bastante empregadas para diversas tarefas de Mineração de Dados, aplicações de Algoritmos Genéticos que fazem parte da computação evolutiva, e Lógica *Fuzzy*. A combinação dessas técnicas, que constitui os Sistemas Híbridos também tem obtido êxito na Mineração de Dados.

O progresso da área de Mineração de Dados e sua utilização nos mais variados domínios e organizações têm motivado o desenvolvimento de diversas ferramentas comerciais além da elaboração de muitos protótipos de pesquisa. O processo de Mineração de Dados é facilitado consideravelmente se for usada uma ferramenta que ofereça suporte para uma

variedade de técnicas, com diferentes algoritmos disponíveis e voltadas para várias tarefas de Mineração de Dados.

O desenvolvimento de ferramentas comerciais de Mineração de Dados tem como objetivo principal fornecer aos tomadores de decisão das organizações, que são usuários geralmente não especialistas em Mineração de Dados, ferramentas intuitivas e amigáveis. É interessante que estas ferramentas ofereçam suporte às várias etapas do processo de Mineração de Dados assim como disponibilizem apoio para diversas técnicas e tarefas.

Inúmeras universidades e laboratórios de pesquisa têm direcionado seus projetos para o desenvolvimento de protótipos de ferramentas de MD com funcionalidades inovadoras. De acordo com essas funcionalidades esses protótipos podem ser agrupados nas seguintes categorias (Thuraisingham, 1999):

Novos Modelos Funcionais Essencialmente, na maioria destes projetos, o objetivo é integrar métodos de Mineração de Dados com o gerenciamento de Banco de Dados.

A idéia principal desses trabalhos é desenvolver novas técnicas para otimizar as consultas e permitir um melhor suporte aos métodos de Mineração de Dados.

Tratamento de Novos Tipos de Dados O tratamento de diferentes tipos de dados, como dados multimídia, é o objetivo dos protótipos pertencentes a esta categoria. Além de ferramentas que tratam dados multimídia, também pertencem a esta categoria as ferramentas que apóiam o processo de Mineração de Textos (Baeza-Yates & B., 1999; Habn & Mani, 2000; Vasileios, Gravano, & Maganti, 2000).

Escalabilidade Com o desenvolvimento da tecnologia de armazenamento, o volume de dados disponível para análise tem crescido exponencialmente. Dessa maneira, diversos projetos de pesquisa têm sido desenvolvidos com o objetivo de tratar grandes bases de dados. Alguns trabalhos que tratam da escalabilidade dos algoritmos de extração de conhecimento estão sendo desenvolvidos.

Compreensibilidade dos Resultados Um outro problema identificado no final do processo de Mineração de Dados é a comprehensibilidade do resultado obtido. Os padrões extraídos pelos algoritmos podem ser muito complexos ou não fazerem sentido para os usuários do processo de extração de conhecimento, dificultando sua compreensão. A visualização dos padrões extraídos tem sido utilizada em muitos projetos como principal técnica para auxiliar a sua compreensão.

O foco principal desta tese está relacionado com o projeto e desenvolvimento de um ambiente, com funcionalidades para o pós-processamento de conhecimento, que é inovador para tarefas de regressão.

2.3 Considerações Finais

Neste capítulo foi apresentada uma visão geral do processo de Mineração de Dados, bem como uma descrição das etapas que compõe esse processo. O processo de MD apresenta dois tipos de atividades principais: a predição e a descrição. Na predição, os exemplos fornecidos possuem um rótulo associado, e o objetivo é predizer o valor do rótulo de novos exemplos não rotulados. Dependendo dos valores que esses rótulos assumem, o problema é denominado de classificação (categórico) ou regressão (numérico).

Visando apoiar o processo de MD, está sendo desenvolvido em nosso laboratório o Ambiente DISCOVER, do qual faz parte o Ambiente *REPPE* desenvolvido neste trabalho de doutorado e voltado para mineração de problemas de regressão. Ambos os ambientes são descritos no Capítulo 4.

O foco principal desta tese é em Mineração de Dados para problemas de regressão, que tem como objetivo encontrar uma relação entre um conjunto de atributos de entrada e um atributo meta contínuo. Existem vários métodos para se realizar uma tarefa de regressão, que geram modelos em diferentes formatos. Esses métodos, bem como os tipos de modelos gerados, são descritos no próximo capítulo.

Regressão



regressão é também conhecida por predição funcional, predição de valor real, função de aproximação, ou ainda, aprendizado de classes contínuas (Uysal & Güvenir, 1999).

Por exemplo, suponha que se deseja predizer a porcentagem de gordura que uma pessoa possui no corpo. Fornecidos exemplos contendo algumas características de pessoas como: sexo, idade, peso e altura, mais a taxa de gordura (atributo meta), o objetivo da regressão é predizer a porcentagem de gordura do corpo de outras pessoas baseado nesses exemplos.

Outro caso seria tentar predizer quantos carros passam em um determinado pedágio. Fornecidos alguns exemplos contendo informações como: rodovia em que o pedágio está localizado, cidades mais próximas, preço do pedágio, dia da semana e período do dia, mais a quantidade de carros (atributo meta), o objetivo da regressão é predizer quantos automóveis passam naquele pedágio.

A grande maioria dos problemas de regressão faz a predição de um único atributo meta. Porém, a regressão também pode ser realizada para predizer mais de um atributo meta. Por exemplo, as Redes Neurais Artificiais permitem trabalhar com tarefa de regressão com vários atributos meta.

Muitos problemas importantes do mundo real são de regressão. Além disso, métodos de regressão podem ser usados para resolver problemas de classificação (Weiss & Indurkhya, 1995).

Os diferentes métodos de regressão podem ser divididos de muitas maneiras. Os cri-

térios normalmente utilizados na divisão dos métodos são baseados no tipo de modelo gerado. Alguns desses critérios levam em consideração se os modelos são paramétricos ou não-paramétricos; realizam um particionamento dos dados ou não; são globais ou locais; são baseados em regressão linear ou não; são aditivos ou não; e são estatísticos ou não.

Os métodos de regressão são estudados pela comunidade estatística há muito tempo. Porém, nas áreas de Aprendizado de Máquina e Mineração de Dados, a maioria das pesquisas trata de problemas de classificação, que são mais comumente encontrados na vida real do que problemas de regressão (Weiss & Indurkhy, 1995). Recentemente, o foco das pesquisas nessas áreas tem se voltado para problemas de regressão, já que muitos problemas são desse tipo, por exemplo, problemas relacionados com habilidades esportivas e controle dinâmico de robôs.

O capítulo encontra-se organizado da seguinte maneira: na Seção 3.1 são descritas as definições e padronizações aqui utilizadas para problemas de regressão. Nas seções seguintes são descritos diferentes métodos para problemas de regressão. Na Seção 3.2 é descrito o método estatístico, sendo detalhada a abordagem paramétrica, que é uma das mais utilizadas pela comunidade estatística; a abordagem não paramétrica, que utiliza os exemplos mais similares do conjunto de treinamento para fazer previsões; e a aditiva, que decompõe uma função de regressão complexa em várias funções mais simples. Na Seção 3.3 são descritos os modelos baseados em aprendizado simbólico, com destaque para as regras e árvores. Na Seção 3.4 são apresentadas as Redes Neurais Artificiais, modelos computacionais inspirados no cérebro humano. Na Seção 3.5 são apresentadas as *support vector machines*, que tentam encontrar um hiperplano para tornar linear um problema complexo. Por fim, na Seção 3.6 são feitas as considerações finais deste capítulo.

3.1 Definições e Formalizações

Em regressão, também conhecida como função de aproximação, o objetivo é predizer um número real ou números ordenados e não categorias e rótulos. Regressão é normalmente mais difícil que classificação, o que incentiva formular os problemas como de classificação sempre que possível. Entretanto, para muitos problemas, como previsão de lucro ou perda, uma formulação de regressão é indispensável (Weiss & Indurkhy, 1998).

Dada uma amostra de um conjunto de atributos de entrada X_1, \dots, X_n junto com o atributo meta Y , tenta-se aproximar uma função, mostrada na Equação 3.1.

$$Y = f(X_1, \dots, X_n) \tag{3.1}$$

Um problema de regressão é, às vezes, descrito como um problema de sinal e ruído.

Assim, o modelo é estendido para incluir um componente estocástico ou limiar ϵ . Nesse caso, a função real não pode produzir uma distância de erro zero. Ao contrário da classificação, na qual os rótulos são assumidos corretos, para regressão os valores preditos de Y podem ser explicados por alguns fatores incluindo um componente de ruído aleatório, ϵ , no sinal, Y (Equação 3.2).

$$Y = f(X_1, \dots, X_n) \pm \epsilon \quad (3.2)$$

Nas próximas quatro seções estão descritos os diferentes métodos de regressão, os quais podem ser divididos em diversas categorias, entre elas: métodos estatísticos, métodos de Aprendizado de Máquina simbólico, métodos baseados em Redes Neurais Artificiais e métodos baseados em *Support Vector Machine*.

3.2 *Métodos Estatísticos*

Regressão é um dos maiores tópicos estudados em análise de dados estatística. Existe uma enorme quantidade de trabalhos que utilizam várias abordagens para esse problema. Essa seção apresenta uma visão geral dos maiores paradigmas estatísticos sem descrever as variantes existentes.

3.2.1 *Abordagem Paramétrica*

A abordagem paramétrica é um dos métodos mais utilizados pela comunidade estatística para resolver problemas de regressão. Uma abordagem desse tipo tenta adaptar todos os dados de treinamento fornecidos a uma única função paramétrica (Uysal & Güvenir, 1999).

Utilizar uma abordagem desse tipo implica em fazer uma forte suposição a respeito da forma da função de regressão não conhecida, podendo levar a uma baixa precisão dependendo do caso. Apesar disso, a abordagem paramétrica tem sido amplamente utilizada e tem fornecido bons resultados de predição quando a função real existente nos dados pode ser representada corretamente por uma função paramétrica. Além disso, esses modelos são mais interpretáveis e possuem soluções computacionais rápidas (Torgo, 1999).

Um exemplo clássico de abordagem paramétrica e também um dos mais utilizados é o modelo paramétrico global utilizando o critério de erro dos mínimos quadrados. Esse critério, mostrado na Equação 3.3 na página seguinte, tenta encontrar um vetor de parâmetros

β que minimize a soma dos erros quadrados.

$$\sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d))^2 \quad (3.3)$$

O modelo de regressão linear utilizando o critério dos mínimos quadrados tem se mostrado eficiente para muitas aplicações do mundo real (Weiss & Indurkhy, 1995). Na Figura 3.1 é mostrado um exemplo em que os dados se adaptam muito bem ao modelo de regressão linear gerado. Esse exemplo utiliza um atributo de entrada x para predizer o valor do atributo meta y .

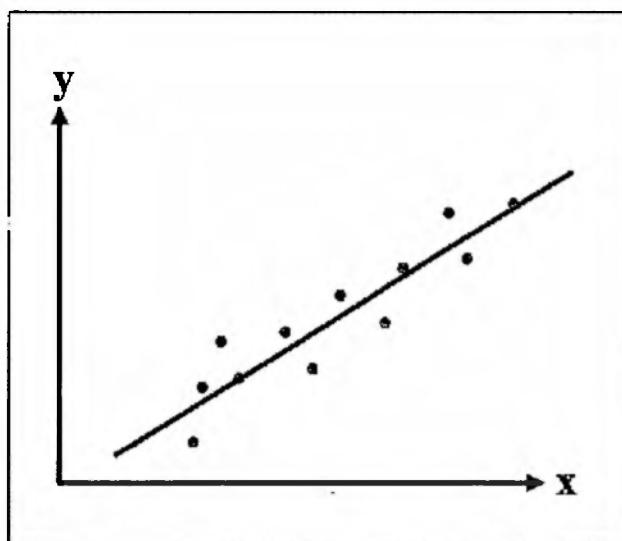


Figura 3.1: Exemplo de regressão linear global.

3.2.2 Abordagem Não-Paramétrica

A modelagem não-paramétrica caracteriza-se por fazer previsões baseadas nos exemplos do conjunto de treinamento mais semelhantes a um determinado exemplo fornecido. Os métodos de modelagem não-paramétrica utilizam alguma medida de similaridade para identificar os exemplos mais similares do conjunto de treinamento, como apresentado na Figura 3.2, e então utilizam esses exemplos para prever o valor do atributo meta.

Esses métodos possuem um custo computacional de treinamento mínimo, pois apenas armazenam os dados na memória. Porém, a fase de previsão apresenta um custo maior, uma vez que para cada novo exemplo fornecido é necessário encontrar os exemplos de treinamento mais similares.

A seguir serão descritos dois métodos baseados na abordagem não-paramétrica: aprendizado baseado em exemplos e *Locally Weighted Regression*.

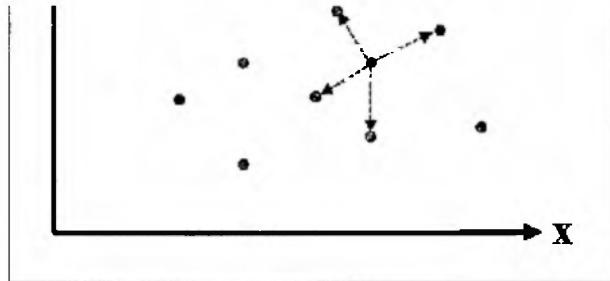


Figura 3.2: Exemplos mais similares do conjunto de treinamento.

Aprendizado Baseado em Exemplos

Os métodos de aprendizado baseado em exemplos (IBL – *Instance Based-Learning*) foram desenvolvidos pela comunidade de Aprendizado de Máquina e são também conhecidos como abordagens baseadas na memória, ou ainda, métodos de aprendizado *lazy*.

De uma maneira geral, o aprendizado baseado em exemplos é composto de (Mitchell, 1997):

1. um conjunto de exemplos de treinamento armazenado na memória;
2. determinação de uma métrica para realizar comparações entre um exemplo fornecido e os mais próximos a ele;
3. determinação de um número k de exemplos mais próximos ou similares que serão utilizados para predizer o valor do atributo meta do novo exemplo.

O aprendizado baseado em exemplos simplesmente armazena todos os exemplos de treinamento na memória (por isso é também chamado de abordagem baseada na memória) sem fazer qualquer tipo de generalização dos dados fornecidos. Portanto, não existe uma fase de treinamento, sendo que todo o trabalho do algoritmo é feito durante a predição. Depois de armazenar os exemplos na memória, utiliza-se alguma métrica para encontrar os exemplos mais similares.

Pelo fato desses algoritmos fazerem suas predições baseadas apenas nos exemplos armazenados na memória, a qualidade da predição pode ser comprometida devido a ruídos nos dados. Para melhorar o desempenho de predição desses algoritmos, o conjunto de dados deve ser selecionado e preparado adequadamente, podendo ser removidos exemplos ou

até mesmo atributos do conjunto de treinamento, desde que isso não afete o desempenho preditivo do algoritmo.

A grande desvantagem do aprendizado baseado em exemplos é que ele não produz abstrações ou modelos que permitam a interpretação do conhecimento contido no conjunto de dados.

O algoritmo do vizinho mais próximo (*Nearest Neighbor* - NN) é um dos mais simples e mais utilizados algoritmos de aprendizado baseado em exemplos. Encontrados os exemplos mais semelhantes, esse algoritmo prediz o valor do atributo meta de um novo caso calculando a média dos valores dos atributos meta dos exemplos mais similares do conjunto de treinamento. O NN compara um novo exemplo x_1 com um outro x_2 previamente armazenado na memória baseado na distância euclidiana. A distância euclidiana, descrita na Equação 3.4, é limitada pelo fato de classificar somente atributos numéricos.

$$\Delta(x_1, x_2) = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + \dots + (x_{1d} - x_{2d})^2} \quad (3.4)$$

em que x_{ij} é o valor do j -ésimo atributo do i -ésimo exemplo e d é o número de atributos.

O algoritmo do vizinho mais próximo puro classifica um caso baseado em um único exemplo similar, o que pode acarretar em erros prematuros. O *k-Nearest Neighbor* é uma versão mais elaborada do NN que classifica um novo exemplo calculando a média entre os k exemplos mais próximos a ele.

O algoritmo *k-NN* assume que todos os exemplos mais similares encontrados são equivalentemente relevantes. Por esse motivo, a precisão da predição do modelo pode ser deteriorada. Uma solução para esse problema, descrita na próxima subseção, é fazer com que, entre os exemplos mais similares encontrados, os que estiverem mais próximos do novo exemplo possuam pesos maiores (Uysal & Güvenir, 1999).

Locally Weighted Regression

O método *Locally Weighted Regression* (LWR), que também faz parte dos métodos de aprendizado *lazy*, é bastante similar à abordagem *Nearest Neighbor* descrita anteriormente, principalmente em três aspectos:

1. a fase de treinamento desses algoritmos consiste apenas em armazenar os exemplos de treinamento na memória, sendo todo o trabalho realizado durante a predição de novos exemplos;
2. os novos exemplos preditos são fortemente influenciados pelos exemplos mais similares previamente armazenados;

3. os exemplos são representados como pontos reais em um espaço d -dimensional.

A principal diferença entre os dois métodos está no modo como eles predizem o valor de um novo exemplo fornecido. Enquanto o k -NN prediz o valor de novos exemplos apenas fazendo a média dos exemplos mais próximos do conjunto de treinamento, o LWR constrói modelos locais adaptando um plano aos exemplos mais próximos do conjunto de treinamento (Atkeson, Moore, & Schaal, 1997). Para construir esses modelos, que geralmente são funções paramétricas lineares ou não-lineares, o LWR utiliza uma abordagem denominada *distance weighted regression*, em que os exemplos mais próximos possuem um peso maior, enquanto que os mais distantes possuem um peso menor. O LWR depende bastante da função de distância utilizada para calcular os exemplos mais próximos (Uysal & Güvenir, 1999).

Como os modelos construídos são locais, depois que um exemplo tem seu valor predito, o modelo utilizado é apagado, e para cada exemplo apresentado, um novo modelo é construído.

3.2.3 Abordagem Aditiva

Um método aditivo é aquele que se “aproveita” do fato de que uma função de regressão complexa pode ser decomposta em partes, sendo que cada uma dessas partes representa uma função mais simples. Assim, um modelo com uma alta dimensão (com muitos atributos) pode ser visto como a soma de funções de dimensões menores (Torgo, 1999). *Project Pursuit Regression* e *Adaptive Regression Splines* são exemplos de métodos aditivos.

Project Pursuit Regression

Project Pursuit Regression (PPR) constitui um modelo refletindo o conjunto de treinamento em projeções de dimensões menores, como uma solução quando o conjunto de dados fornecido possui uma alta dimensão. PPR funciona da seguinte maneira: em cada passo da construção do modelo, a melhor aproximação dos dados encontrada é selecionada, adicionada ao modelo que está sendo gerado e essa parte é removida do conjunto de exemplos. A busca pela melhor aproximação continua com os dados restantes do conjunto de treinamento até que todos façam parte do modelo final (Uysal & Güvenir, 1999).

A desvantagem apresentada por esse método é a complexidade computacional. Quando se possui um conjunto grande de funções candidatas, é necessário escolher uma entre elas e refinar os parâmetros da função selecionada de modo que ela se adapte da melhor maneira aos dados.

Adaptive Regression Splines

Adaptive Regression Splines pode ser visto como uma generalização das árvores de regressão, apresentadas na Seção 3.3.2, e foram desenvolvidas para superar algumas de suas limitações. Algumas dessas limitações são: (i) a lacuna de continuidade apresentada pelas árvores de regressão, que afeta a capacidade preditiva do modelo, e (ii) a incapacidade das árvores de regressão de fornecerem boas aproximações para algumas funções (Uysal & Güvenir, 1999). A lacuna de continuidade corresponde ao fato de que os valores de um nó-folha não são contínuos em relação aos valores dos outros nós-folha.

Uma adaptação paramétrica *piecewise* aproxima uma função por meio de várias funções paramétricas simples (geralmente polinômios de ordens menores), cada uma definida sobre diferentes sub-regiões do conjunto de treinamento. Esses polinômios precisam ser contínuos em cada ponto, isto é, cada função definida sobre uma sub-região do conjunto de treinamento deve ser contínua com relação à função definida sobre a próxima sub-região deste conjunto.

O mais popular entre os procedimentos de adaptação paramétrica *piecewise* são aqueles baseados em *splines*, em que as funções paramétricas globais são polinômios. O procedimento é implementado por intermédio da construção de um conjunto de funções-base definidas globalmente.

O algoritmo MARS (*Multivariate Adaptive Regression Splines*), desenvolvido no início dos anos 90 por Jerry Friedman, é um algoritmo de particionamento recursivo que possui algumas modificações para resolver os problemas discutidos nesta seção, principalmente no que diz respeito à descontinuidade. A metodologia MARS é detalhada em (Friedman, 1991; Hastie, Tibshirani, & Friedman, 2001).

3.3 Métodos de Aprendizado de Máquina Simbólico

A comprehensibilidade dos modelos gerados é considerada muito importante quando se realiza uma tarefa de regressão. A necessidade de modelos capazes de fornecerem soluções interpretáveis levou a comunidade de Aprendizado de Máquina a desenvolver os métodos de aprendizado simbólico. Esses métodos variam de acordo com a linguagem escolhida para representar as hipóteses. A lógica proposicional é uma linguagem de representação de hipóteses bastante utilizada pelos métodos de aprendizado simbólico (Mitchell, 1997).

Os métodos de Aprendizado de Máquina simbólico proposicionais são aqueles que utilizam a lógica proposicional para representar suas hipóteses. A lógica proposicional permite determinar a validade de proposições compostas por meio de conectivos a partir da validade de fatos simples e da interpretação desses conectivos. Os conectivos utilizados

para compor as proposições podem ser do tipo AND, OR, NOT, etc. Se a hipótese é representada por uma disjunção (conectivo OR) de conjunções (conectivo AND), a notação proposicional é denominada Forma Normal Disjuntiva (FND), e se a hipótese é representada por uma conjunção de disjunções, então a notação é chamada de Forma Normal Conjuntiva (FNC). Dentre os modelos que fornecem soluções na FNC, destacam-se, no caso de regressão, as regras e árvores de regressão.

As regras e as árvores são bastante semelhantes, diferenciando-se, entre outros, pelo fato de que as árvores são mutuamente exclusivas, e as regras nem sempre. Portanto, para cada exemplo, uma ou mais regras podem ser satisfeitas, ao contrário das árvores. As regras e as árvores, dependendo da aplicação, oferecem vantagens e desvantagens, que serão descritas nas próximas subseções.

Pelo fato de fornecerem soluções comprehensíveis, este trabalho tem como foco principal os modelos de regressão baseados em Aprendizado de Máquina simbólico proposicional.

3.3.1 Regras de Regressão

Uma regra na Forma Normal Conjuntiva (FNC) geralmente é representada da seguinte forma:

$$\text{if } \underbrace{\langle \text{condição} \rangle}_{\text{Body} - B} \text{ then } \underbrace{\langle \text{conclusão} \rangle}_{\text{Head} - H}$$

ou, na forma simplificada:

$$B \rightarrow H$$

Assim, uma regra na FNC é composta de duas partes:

- a *<condição>*, cauda ou corpo da regra – *B*, que consiste de uma conjunção de pares no formato atributo-valor
- a *<conclusão>* ou cabeça da regra – *H*, que é uma função ou um valor numérico real, é responsável pela predição do valor do atributo meta.

Neste trabalho, são utilizados algoritmos cujo poder de suas linguagens de representação é equivalente ao da lógica proposicional. Para esses algoritmos, o *Body* é tipicamente uma conjunção de restrições sobre o domínio *E* de um atributo de *X*, ou, em outras palavras, o *Body* é uma conjunção de condições para o atributo de *X*. Cada condição geralmente é representada da forma:

$$X_i; op \ valor,$$

na qual X_i representa o i -ésimo atributo, op é um operador pertencente ao conjunto $\{\neg, \neq, \in, \notin, <, \leq, >, \geq\}$, e $valor$ é um valor (ou uma faixa de valores para os operadores \in e \notin) condicionante do atributo X_i .

Quando se trata de uma regra de regressão, a predição pode ser feita por meio da média dos valores do atributo meta, de uma equação linear, da utilização do método *k-Nearest Neighbor*, entre outros.

Como exemplo de um algoritmo de indução de regras de regressão destaca-se o Cubist, descrito na Seção 4.1.3.

3.3.2 Árvores de Regressão

As árvores são compostas por dois tipos de nós:

- os nós internos da árvore, sendo que cada um desses nós corresponde a um teste feito em um dos atributos de entrada do conjunto de exemplos;
- os nós-folha, nos quais são feitas as predições do atributo meta.

Os nós-folha de uma árvore de regressão possuem uma função matemática (que no caso mais simples pode ser a simples média dos valores que caem em cada nó-folha) para predizer o atributo meta.

As árvores que predizem o valor de um atributo meta contínuo são conhecidas como *Regression Tree* ou *Model Tree*, dependendo de como os nós-folha calculam o valor desse atributo meta. Quando os nós-folha da árvore apenas calculam a média de todos os exemplos de treinamento que caem naquele nó, então a árvore é conhecida como *Regression Tree*. Por outro lado, se os nós-folha possuem algum modelo de regressão linear para prever o valor do atributo meta, então a árvore é conhecida como *Model Tree*. Um exemplo de uma *Model Tree* é mostrado na Figura 3.3. Essa árvore foi construída utilizando dois atributos de entrada (X_1 e X_2) do conjunto de dados.

Um dos pontos mais importantes a serem levados em consideração quando uma árvore de regressão está sendo construída é a seleção do atributo que será utilizado para fazer o particionamento dos dados em cada nó da árvore. Além de selecionar o atributo, é muito importante escolher um valor adequado para esse atributo de modo a partitionar os dados da melhor maneira possível.

As principais vantagens apresentadas pelos modelos de árvores de regressão são: (i) seleção dinâmica de atributos: os métodos de indução de árvores são extremamente efetivos

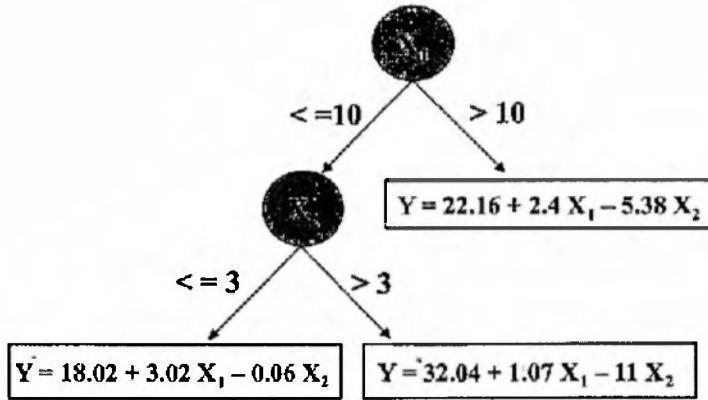


Figura 3.3: Exemplo de uma *Model Tree*.

em encontrar os atributos-chave (os mais importantes) em aplicações de alta dimensão, e (ii) sua capacidade exploratória, porém, à medida que a árvore cresce, sua interpretabilidade diminui. Quanto ao desempenho, as árvores de regressão se equiparam aos demais métodos de regressão (Weiss & Indurkhya, 1995).

Os algoritmos de indução de árvores constroem modelos fazendo um particionamento recursivo do conjunto de treinamento da seguinte maneira: o nó-raiz da árvore contém um atributo que divide o conjunto de treinamento em sub-regiões distintas (geralmente duas sub-regiões). Os filhos do nó-raiz podem ser nós-folhas, ou então, são selecionados novos atributos com seus respectivos valores, dividindo assim, o conjunto de treinamento em novas subdivisões. O número de nós de uma árvore e a profundidade da mesma variam de acordo com o tamanho do conceito representado nos dados e com o mecanismo de inferência utilizado pelo algoritmo responsável pela construção da árvore. Uma árvore muito grande provavelmente causa um “*overfitting*” dos dados, enquanto que uma árvore muito pequena geralmente não oferece bons resultados (Hastie, Tibshirani, & Friedman, 2001).

Em geral, os algoritmos de indução de árvores apresentam as seguintes características em comum: (i) particionam o conjunto de treinamento em regiões disjuntas recursivamente, na qual a partição final é determinada pelos nós-folha da árvore, e (ii) utilizam estratégias de poda para evitar o *overfitting* (Torgo, 1999).

Os diversos algoritmos de indução de árvores de regressão diferenciam-se pelas estratégias de poda utilizadas, pela maneira como selecionam os atributos que particionarão os dados, e principalmente, pelo tipo de função que utilizam para predizer o valor do atributo meta (Torgo, 1997). Alguns exemplos de algoritmos desse tipo são o RETIS (*Regression Tree Induction System*), o M5 e o CART (*Classification And Regression Trees*), descritos na Seção 4.1.3.

3.4 Métodos Baseados em Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados no cérebro humano. Elas são compostas por várias unidades de processamento (neurônios), interligadas por um grande número de conexões (sinapses).

Graficamente, uma RNA pode ser vista como um conjunto de nós (unidades de processamento) e arcos, representando respectivamente, os neurônios e as conexões entre os mesmos (Braga, Carvalho, & Ludermir, 2000). Uma estrutura genérica de Rede Neural Artificial é composta por uma camada de entrada, uma ou mais camadas intermediárias (escondidas) e uma camada de saída. Cada uma dessas camadas é composta por um número variado de neurônios, dependendo da aplicação. Os nós da camada de entrada estão relacionados com os atributos de entrada e contêm os valores dos exemplos fornecidos à RNA. Os nós das camadas intermediárias recebem um conjunto de entradas provenientes de outros nós, computam uma função (conhecida como função de ativação) sobre essas entradas e então enviam o resultado para outro conjunto de nós. Cada arco que conecta dois nós possui um peso associado. A camada de saída da RNA representa os valores de saída da rede, que são calculados por intermédio dos atributos de entrada e dos pesos associados às conexões, que para regressão representa o atributo meta y .

Um exemplo genérico de uma Rede Neural Artificial, com uma camada intermediária e uma saída (atributo meta), é mostrado na Figura 3.4.

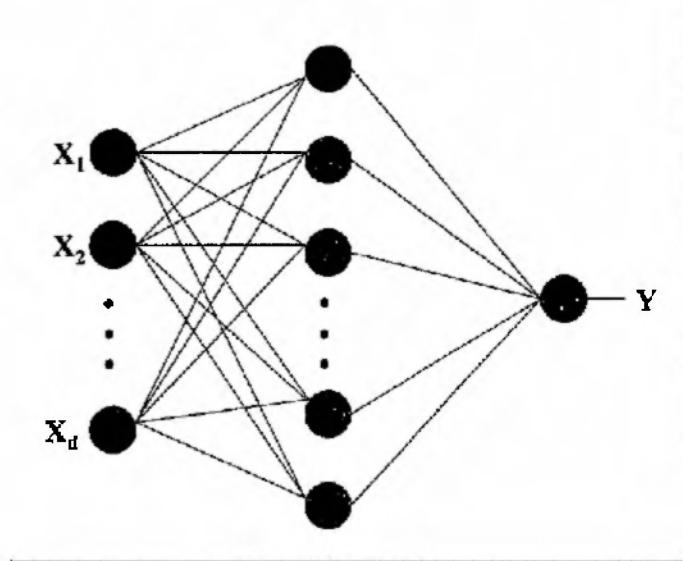


Figura 3.4: Exemplo de uma rede neural artificial.

O aprendizado em uma Rede Neural Artificial consiste em encontrar os valores dos pesos de modo a fornecer a saída correta. A rede começa com um vetor de pesos aleatórios, e então, à medida que lhe são fornecidos exemplos de treinamento, os pesos vão sendo

modificados até que as previsões se tornem satisfatórias.

Quando se trata de um problema de regressão, a Rede Neural Artificial é utilizada para predizer o valor dos atributos contínuos (as saídas fornecidas pela RNA são valores contínuos). Alguns exemplos em que as RNAs tem sido utilizadas para fazer regressão são: previsão de séries temporais, aproximação de funções e fusão de sensores (Braga, Carvalho, & Ludermir, 2000, 2003; Faceli, Carvalho, & Rezende, 2002).

A grande vantagem das RNAs sobre os outros métodos é que elas não são restritas à um único atributo de saída, como acontece na maioria dos casos. Portanto, podem ser realizadas várias regressões em uma RNA. Além disso, as RNAs são conhecidas pela alta precisão na previsão dos valores e, geralmente, são robustas diante de dados com ruído.

Porém, as RNAs também apresentam algumas desvantagens. A primeira é que, dependendo do modelo de rede e do algoritmo de aprendizado, as Redes Neurais podem apresentar lenta convergência para uma solução, pois a rede pode precisar ser treinada até que os pesos estejam corretos de modo a fornecer a saída esperada. Outra desvantagem é que as soluções por ela fornecidas não são facilmente interpretadas pelos usuários, pois o conhecimento está embutido nos pesos e conexões da rede.

3.5 *Métodos Baseados em Support Vector Machines*

A teoria *Support Vector Machines* (SVM), ou Máquinas de Suporte Vetoriais, foi proposta no final da década de 60 por V. Vapnik e A. Chervonenkis. Uma descrição detalhada desse método pode ser encontrada em (Vapnik, 1998).

O princípio das SVMs consiste em encontrar um hiperplano ótimo que separe membros e não-membros de uma classe em um espaço abstrato, denominado *feature space*. Nesse espaço, as classes presentes no conjunto de treinamento se tornam linearmente separáveis, e o hiperplano ótimo é definido como aquele para o qual a margem de separação entre as mesmas é maximizada.

Uma propriedade importante das SVMs está na utilização de *kernels*. Os *kernels* são produtos internos das coordenadas de dois vetores, e são utilizados para construção do hiperplano ótimo no *feature space* sem a necessidade de considerar a forma explícita desse, geralmente bastante complexa (Haykin, 1999). Alguns exemplos de *kernel* utilizados são: o polinomial, o radial e o sigmoidal.

Algumas vantagens apresentadas pelas SVMs são:

- trabalham bem quando o conjunto de dados possui uma alta dimensão;
- costumam apresentar uma alta precisão na previsão de valores;

- não existe o risco de encontrarem mínimos locais, um problema que ocorre bastante quando se trabalha com Redes Neurais Artificiais.

Uma desvantagem apresentada pelas SVMs é que os modelos fornecidos não são facilmente compreensíveis ao ser humano.

As SVMs têm despertado grande interesse devido a obtenção de resultados que superam os obtidos por outras abordagens de Aprendizado de Máquina na solução de alguns problemas, por exemplo, na categorização de textos e no reconhecimento de faces.

3.6 Considerações Finais

Neste capítulo foi apresentada uma definição de problemas de regressão, o qual prediz um valor numérico contínuo. Além disso, foram descritos alguns dos diversos métodos existentes para se realizar uma tarefa de regressão, que geram modelos em diferentes formatos de representação, como métodos estatísticos, métodos de AM simbólico, métodos baseados em Redes Neurais Artificiais e métodos baseados em *Support Vector Machines*.

Nesta tese, utiliza-se principalmente os modelos baseados em Aprendizado de Máquina simbólico, pois possuem como principal característica a comprehensibilidade, isto é, fornecem soluções mais facilmente interpretáveis ao ser humano. As regras e árvores são modelos desse tipo e são as abordagens mais utilizadas pela comunidade de Aprendizado de Máquina.

No próximo capítulo será apresentado um ambiente para Mineração de Dados para problemas de regressão denominado *REPPE*, que visa auxiliar o usuário principalmente na etapa de pós-processamento do conhecimento.

O Ambiente REPPE: Novas Funcionalidades no Discover

A extração de padrões e modelos de grandes bases de dados geralmente é feita utilizando-se metodologias, técnicas e algoritmos de aprendizado. Entretanto, outros aspectos desse processo de extração devem ser considerados, como o estudo de métodos para avaliação da precisão, qualidade e comprehensibilidade do conhecimento obtido de uma ou mais amostras submetidas a algoritmos de aprendizado.

O processo de Mineração de Dados, como apresentado no Capítulo 2, é composto por uma série de etapas, sendo que as etapas de pré e pós-processamento consomem a maior parte do esforço despendido ao longo do processo (Rezende, 2003).

Na fase de pré-processamento, uma série de atividades são necessárias, como adequações de formato, tratamento de valores ausentes e transformações. Em especial, a falta de um formato único para os conjuntos de exemplos utilizados causa um grande transtorno às comunidades de Mineração de Dados e Aprendizado de Máquina. Por exemplo, com relação aos métodos de regressão, associação e classificação simbólicos, cada algoritmo utilizado para induzir uma hipótese possui um formato diferente para seus dados de entrada. Dessa forma, cada vez que é preciso executar algum desses algoritmos, é necessário converter o conjunto de dados para o formato de entrada do algoritmo, além de efetuar outras operações, como limpeza nos dados, tratamento de valores desconhecidos e outras inconsistências.

Já na fase de pós-processamento, o panorama não é muito diferente. Ainda com rela-

ção ao aprendizado simbólico, são distintas as hipóteses induzidas pelos algoritmos. Em consequência, qualquer operação que necessite ser feita sobre a hipótese induzida (como construção da matriz de contingência e o cálculo de medidas para avaliação do conhecimento) precisa ser implementada diversas vezes, uma para cada formato de hipótese.

A regressão ainda é uma tarefa pouco explorada dentro da área de Aprendizado de Máquina e Mineração de Dados, visto que a maioria dos trabalhos trata de problemas de classificação. Nesse sentido, neste capítulo é apresentado o Ambiente *REPPE* (*Regression Post-Processing Environment*) que visa o pós-processamento de conhecimento para problemas de regressão o qual está integrado ao DISCOVER, que é um projeto de grande porte para apoiar a extração de conhecimento. Esse projeto vem sendo desenvolvido no LABIC pelos alunos das professoras Dra. Maria Carolina Monard e Dra. Solange Oliveira Rezende.

Este capítulo está organizado da seguinte maneira: na Seção 4.1 é feita uma contextualização para a apresentação do Ambiente *KEPPE* em que é descrito o Ambiente DISCOVER, as ferramentas utilizadas para a geração dos regressores e as bases de dados utilizadas nos experimentos. Na Seção 4.2 é apresentada a sintaxe padrão definida para representar regras de regressão. Para isso é mostrada a gramática da sintaxe padrão e a conversão dos regressores para o formato padrão. Na Seção 4.3 é mostrado o Ambiente *KEPPE* que foi projetado e desenvolvido para apoiar o pós-processamento de regras de regressão. Finalmente, na Seção 4.4 são apresentadas as considerações finais deste capítulo.

4.1 Contextualização

A principal finalidade do Ambiente *REPPE*, aqui proposto e implementado, é apoiar os usuários do processo de Mineração de Dados para problemas de regressão. As funcionalidades implementadas nesse ambiente são responsáveis pelo tratamento de problemas de regressão no Ambiente DISCOVER, ou seja, apesar de ser desenvolvido de forma independente o *REPPE* complementa o DISCOVER. Dessa maneira, é apresentada nessa seção a contextualização das notações, o Ambiente DISCOVER e as ferramentas e bases de dados utilizadas na realização de toda a avaliação empírica dos módulos que compõem o *REPPE*.

4.1.1 Notação e Terminologia

Em geral, tarefas de Mineração de Dados podem ser agrupadas em duas categorias: descritivas e preditivas. A primeira descreve o conjunto de exemplos (também chamados de casos ou dados) de uma maneira concisa e sumarizada, apresentando algumas propri-

edades gerais dos dados; a segunda constrói uma hipótese fazendo inferências acerca dos exemplos disponíveis na tentativa de prever o comportamento de novos casos.

Exemplos

Os algoritmos utilizados nessas duas tarefas geralmente usam como entrada um conjunto de exemplos D , contendo n exemplos. Cada um desses exemplos é descrito por um vetor x , sendo que cada um dos elementos desse vetor representa um dos possíveis m atributos distintos. Cada um desses atributos pode assumir um dos possíveis valores correspondente ao seu domínio E . Esse domínio pode ser tanto discreto quanto contínuo.

Conjuntos de exemplos são geralmente representados como uma matriz composta de n exemplos e m atributos, como mostrado na Tabela 4.1. Nessa matriz, a linha x_i refere-se ao i -ésimo exemplo (para $i = 1, 2, \dots, n$) e cada entrada x_{ij} refere-se ao valor do j -ésimo (para $j = 1, 2, \dots, m$) atributo X_j do exemplo x_i , ou seja, cada exemplo x_i é um elemento do conjunto $E(X_1) \times E(X_2) \times \dots \times E(X_m)$. Esse formato de representar exemplos é conhecido na literatura como formato atributo-valor.

A última coluna da tabela corresponde aos valores do atributo meta (y_i é o valor do atributo meta do exemplo x_i). Cada exemplo corresponde a uma tupla $(x_{i1}, x_{i2}, \dots, x_{im}, y_i)$, e $y_i = h(X_i)$ é a função que calcula o valor do atributo meta em função dos demais atributos.

Tabela 4.1: Conjunto de exemplos no formato atributo-valor.

Exemplo	X_1	X_2	...	X_m	Y
x_1	x_{11}	x_{12}	...	x_{1d}	y_1
x_2	x_{21}	x_{22}	...	x_{2d}	y_2
x_3	x_{31}	x_{32}	...	x_{3d}	y_3
\vdots	\vdots	\vdots	...	\vdots	\vdots
x_n	x_{n1}	x_{n2}	...	x_{nm}	y_n

Assim, para algoritmos cujo objetivo é construir um modelo preditivo, existe um atributo especial, o qual se tenta inferir baseado nos demais atributos. Em outras palavras o que se deseja é construir um modelo que aproxime uma função desconhecida $f(X) = Y$, por uma hipótese $h(X) \approx Y$. Se $E(Y)$ é discreto, tem-se um problema de classificação. Se $E(Y)$ é contínuo, tem-se um problema de regressão.

Regras

Os algoritmos utilizados em Mineração de Dados geralmente representam o conhecimento no formato de regras. Alguns algoritmos de aprendizado preditivo também induzem árvores de decisão. Como sempre é possível reescrever uma árvore de decisão como

um conjunto de regras disjuntas, o termo *regra* também refere-se a uma regra extraída de uma árvore de decisão (regras disjuntas). Regras são implicações entre um *Body* ou *Corpo* e um *Head* ou *Cabeça*, assumindo o seguinte formato, como já apresentado com mais detalhes na Seção 3.3.1.

$$Body \rightarrow Head,$$

ou resumidamente

$$B \rightarrow H.$$

Em algoritmos descritivos, o *Head* pode assumir o mesmo formato que o antecedente de uma regra. Em algoritmos preditivos, o *Head* ou é a designação de uma classe (um dos possíveis valores no domínio discreto $E(Y)$) que prediz o atributo Y para problemas de classificação ou é uma equação que representa uma regressão feita com os exemplos pertencentes a C que prediz o atributo Y para problemas de regressão.

Exemplos positivos (negativos) previstos são aqueles para os quais o *Body* é verdadeiro (falso), e exemplos positivos (negativos) reais são aqueles para os quais o *Head* é verdadeiro (falso). Um exemplo é coberto por uma regra se o seu antecedente é verdade para esse exemplo.

Matriz de Contingência

Uma medida de avaliação de regra tem como objetivo dar uma indicação do poder de associação (hipotético) entre o *Body* e o *Head* daquela regra. Para mensurar essa associação, a cada uma das partes da regra, *Body* e *Head*, é possível associar respectivamente variáveis aleatórias e dicotômicas B e H . Essas variáveis assumem os valores verdadeiro, denotadas por B e H , e falso, denotado por \bar{B} e \bar{H} , se o *Body* e o *Head* da regra são avaliados como verdadeiro ou falso, respectivamente.

Dada uma regra arbitrária $B \rightarrow H$, um conjunto de exemplos contendo n exemplos e um procedimento com o qual verifica-se se cada possível *Body* ou *Head* é verdadeiro ou falso, é possível construir uma matriz de contingência para essa regra, mostrada na Tabela 4.2.

A matriz de contingência pode ser construída, para qualquer regra, a partir do conjunto de dados. Na Tabela 4.2, B denota o conjunto de exemplos para o qual o corpo (*Body*) da regra é verdadeiro, e \bar{B} denota o seu complemento (o conjunto de exemplos para o qual o corpo da regra é falso) e de forma similar H e \bar{H} . Assim, BH denota $B \cap H$, $B\bar{H}$ denota $B \cap \bar{H}$, e assim por diante. Utilizou-se o $n(X)$ para denotar a cardinalidade do conjunto

X, por exemplo, $n(BH)$ é o número de exemplos para o qual B é verdadeiro e H é falso (ou seja, o número de exemplos erroneamente cobertos pela regra).

Tabela 4.2: Matriz de contingência para a regra $B \rightarrow H$.

	B	\bar{B}	
H	$n(BH)$	$n(\bar{B}H)$	$n(H)$
\bar{H}	$n(\bar{B}\bar{H})$	$n(\bar{B}H)$	$n(\bar{H})$
	$n(B)$	$n(\bar{B})$	n

A matriz de contingência também pode ser representada usando frequências relativas, como mostrado na Tabela 4.3, em que os valores presentes na matriz de contingência foram divididos por n , ou seja, a frequência relativa $\frac{n(X)}{n}$ associada a X é denotada por $p(X)$. Esse valor da frequência relativa fornece uma estimativa não-viciada e não-tendenciosa da probabilidade associada ao evento X .

Tabela 4.3: Matriz de contingência expressa em frequências relativas para a regra $B \rightarrow H$.

	B	\bar{B}	
H	$p(BH)$	$p(\bar{B}H)$	$p(H)$
\bar{H}	$p(\bar{B}\bar{H})$	$p(\bar{B}H)$	$p(\bar{H})$
	$p(B)$	$p(\bar{B})$	1

Usando como base a matriz de contingência, é possível definir a maioria das medidas propostas na literatura para avaliação de regras, porém, o cálculo da matriz de contingência para regras de regressão não é realizado de forma direta como em classificação. Assim, no Capítulo 5 são apresentadas algumas abordagens para calcular a matriz de contingência para regras de regressão e dessa maneira incrementar o número de medidas para avaliação dessas regras.

4.1.2 O Ambiente Discover

Em nosso laboratório de pesquisa, LABIC, sempre foram realizadas implementações nas áreas de Aprendizado de Máquina e Mineração de Dados por parte de professores e alunos, conforme as necessidades de cada um. Porém, devido à falta de documentação dos programas já implementados, ou ainda porque seus autores se desligavam do laboratório, normalmente essas implementações não eram reaproveitadas. Dessa forma, um mesmo programa era implementado diversas vezes por diferentes pessoas.

Esses fatores impulsionaram então pesquisadores do LABIC a criarem um projeto que envolvesse membros do laboratório. Esse projeto, que recebeu o nome de DISCOVER, tem como objetivo fornecer um ambiente integrado para apoiar as etapas do processo

de Extração de Conhecimento de Dados e Textos (Baranauskas & Batista, 2000). Nesse ambiente são utilizados algoritmos de aprendizado implementados pela comunidade, bem como módulos com finalidades específicas desenvolvidos pelos pesquisadores do LABIC.

Assim, o objetivo principal desse ambiente é fornecer um campo de prova para os pesquisadores da área. A vantagem do DISCOVER como ferramenta de apoio à pesquisa a descoberta de conhecimento, em relação a outros sistemas, é a visão unificada que os formatos padrão utilizados oferecem para quem está desenvolvendo novos componentes, além de um conjunto de ferramentas para a sua manipulação.

Além disso, se espera que o DISCOVER ofereça uma série de vantagens em relação à composição das ferramentas desenvolvidas em nosso laboratório, reunindo-as em um ambiente integrado.

O Ambiente DISCOVER oferece funcionalidades relacionadas ao pré-processamento de dados, pré-processamento de textos e pós-processamento de conhecimento com relação à avaliação de modelos, regras e clusters e derivação de novos modelos. As principais funcionalidades desse ambiente são ilustradas na Figura 4.1.

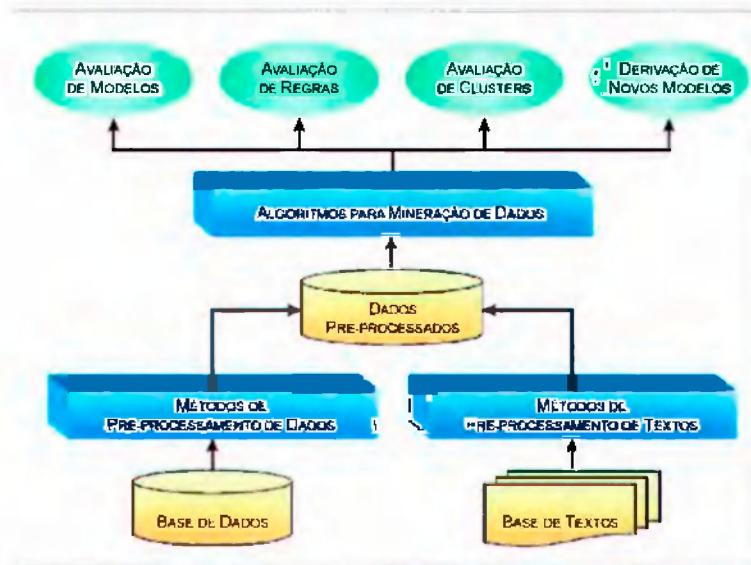


Figura 4.1: Funcionalidades do Ambiente DISCOVER.

O Ambiente DISCOVER é capaz de acessar os dados na forma atributo-valor ou no formato texto, devendo consequentemente estar disponível um conjunto de métodos para pré-processamento de dados e de textos. Caso a quantidade de dados exceda a capacidade dos algoritmos de extração de padrões, amostras representativas devem ser geradas para serem fornecidas aos algoritmos. Os resultados dos algoritmos de extração de padrões ou modelos podem ser avaliados através de diversas medidas, entre elas precisão, compreensibilidade e grau de interesse.

Inicialmente, o Ambiente DISCOVER consistia apenas de um repositório de *scripts*¹, que foram implementados na linguagem PERL (*Practical Extraction and Report Language*) (Till, 1996) e reutilizados de forma a facilitar a execução de experimentos. Algumas vantagens dessa linguagem são:

- permite desenvolver códigos de maneira bastante rápida;
- fornece um poderoso mecanismo para trabalhar com expressões regulares, muito utilizadas nessas implementações;
- permite manipular grande quantidade de dados de maneira eficiente e é bastante flexível com relação aos tipos de dados;
- facilita a escrita de protótipos e *scripts*, bem como funções genéricas e pacotes;
- além dessas vantagens, o CPAN² fornece um excelente repositório de código reutilizável dessa linguagem.

Portanto, *a priori*, o Ambiente DISCOVER consistia de um conjunto de *scripts* independentes que poderiam ser combinados conforme as necessidades dos usuários, e reunidos em um repositório compartilhado entre esses usuários. Assim, os usuários poderiam combinar esses *scripts* para executar experimentos mais complexos.

Posteriormente, surgiu a proposta de se criar um ambiente integrado, em que os *scripts* fossem substituídos por bibliotecas de classes empacotadas como componentes, com a sua composição sendo feita por meio de uma interface gráfica. Dessa forma, os programas desenvolvidos pelos diversos pesquisadores são componentes e o DISCOVER um ambiente computacional que possibilita a construção de experimentos por meio da composição desses componentes.

No entanto, para tornar o DISCOVER um ambiente integrado, sua implementação tornou-se bem mais complexa. Surgiu a necessidade de uma maior comunicação entre os participantes e de interação entre os diferentes componentes. Com relação à arquitetura do ambiente, em (Prati, 2003) foi proposto um *framework* para a integração dos componentes do Ambiente DISCOVER baseado em *software patterns*, no qual os componentes são integrados por meio de uma linguagem baseada em XML. Além disso, a interface gráfica responsável pela composição dos componentes está sendo implementada de acordo com a arquitetura proposta para o ambiente. Essa interface é apresentada em (Geromini, 2002).

Por meio de interfaces bem definidas entre os módulos e procedimentos para conversão de dados e de conhecimento para as sintaxes padrão, que é a principal forma de comunicação entre as funcionalidades, são utilizados no DISCOVER os algoritmos já desenvolvidos,

¹Scripts são pequenos programas que realizam tarefas atômicas.

²Endereço do site CPAN (*Comprehensive Perl Archive Network*) – <http://www.cpan.org/>.

testados e consolidados pela comunidade científica, evitando assim sua reimplementação e garantindo maior confiabilidade aos experimentos realizados.

O foco do Ambiente DISCOVER é o aprendizado simbólico cujas hipóteses são representadas usando árvores ou regras. Assim, foram especificadas diferentes sintaxes para representar as hipóteses obtidas pelos algoritmos de classificação, regressão e regras de associação. Além disso, como toda hipótese representada usando árvores de decisão pode ser transformada diretamente para regra de decisão optou-se por definir sintaxes padrão para representar regras de classificação, de regressão e de associação. O esquema de utilização das sintaxes padrão do Ambiente DISCOVER é ilustrado na Figura 4.2. De acordo com a figura, inicialmente, os conjuntos de dados são convertidos para a sintaxe padrão de dados do DISCOVER. Após o pré-processamento dos dados e definição do algoritmo a ser utilizado, os dados são convertidos para o formato de entrada do algoritmo selecionado. Deve-se ressaltar que, para cada tipo de conjunto de regras, diversos algoritmos podem ser utilizados. A conversão pode ser feita utilizando os métodos definidos na biblioteca para tratamento dos dados na sintaxe padrão (Batista, 2002, 2003).

Após a execução dos algoritmos, seus resultados são convertidos para a sintaxe padrão de regras do tipo de problema tratado (classificação, regressão ou associação). Isto permite também que os resultados de vários algoritmos de extração de padrões possam passar pelos mesmos métodos de pós-processamento que este ambiente oferece.

Além da sintaxe padrão de regras também foi definida para cada tipo de problema uma sintaxe padrão estendida que já engloba para cada regra alguns dados adicionais que são frequentemente utilizados no pós-processamento dessas regras.

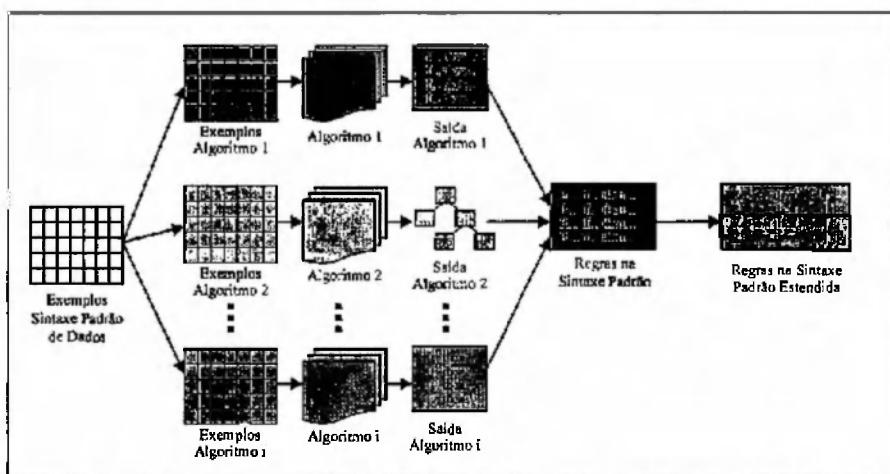


Figura 4.2: Metodologia para obtenção de regras nas sintaxes padrão.

Todas as sintaxes padrão do Ambiente DISCOVER estão descritas detalhadamente em (Rezende, Monard, Batista, Prati, Pugliesi, & Melanda, 2004). Por ser a regressão o foco desta tese é apresentada neste capítulo apenas a sintaxe padrão para representação de

dados e de regras de regressão. Ressalta-se que o *REPPE* é desenvolvido na filosofia do Ambiente DISCOVER e a sintaxe padrão para representar regras de regressão é definida para o DISCOVER. A sintaxe padrão para regras de regressão será apresentada na Seção 4.2 com mais detalhes por ser a sintaxe por nós definida e que é o ponto de partida para todo o pós-processamento que pode ser realizado utilizando o Ambiente *REPPE*.

Sintaxe Padrão para Representação de Dados

A sintaxe padrão para representação de dados, denominada DSX (*DISCOVER Dataset Syntax*), foi definida em (Batista, 2002) como sendo uma extensão do formato utilizado como entrada do algoritmo *C4.5*. Esta extensão foi realizada para adequar o formato dos arquivos de dados para as necessidades do Ambiente DISCOVER. Salienta-se que a principal forma de comunicação entre as diversas funcionalidades implementadas no DISCOVER é por meio da representação dos dados e conhecimento adquirido em formatos padrões definidos utilizando as sintaxes padrão. Assim, inicialmente a Base de Dados é colocada na sintaxe padrão de dados visando uma unificação entre todos os trabalhos realizados dentro do projeto do Ambiente DISCOVER.

Um conjunto de exemplos na sintaxe padrão para representação de dados deve apresentar dois arquivos. Os nomes dos atributos do conjunto de dados, assim como seus respectivos domínios, devem estar especificados em um arquivo *.names*. Os valores dos atributos para os exemplos devem estar contidos em um arquivo *.data*. Esses dois arquivos devem possuir o mesmo nome, se diferenciando somente pela extensão.

Na Figura 4.3 é apresentado um exemplo de arquivo *.names*. Nesse arquivo, a primeira linha apresenta o nome do atributo correspondente ao atributo classe, caso ele exista. Na Figura 4.4 é apresentado um exemplo de arquivo *.data* correspondente ao arquivo *.names* apresentado na Figura 4.3.

```
y.  
x1: real.  
x2: real.  
x3: nominal(brown, green, red).  
x4: real.  
x5: real.  
x6: nominal(no, yes).  
x7: nominal(large, normal).  
x8: real.  
x9: real.  
y: real.
```

Figura 4.3: Exemplo de arquivo *.names* na sintaxe padrão.

Uma biblioteca foi implementada para manipulação de conjunto de dados na sintaxe padrão. Esta biblioteca apresenta métodos para conversão de dados na sintaxe padrão

```

3.50591, -13.114, brown, -6.55701, -17.3705, no, normal, 153.043, 1026, -13.8646
0.254831, -10.0493, brown, -5.02465, -3.72734, no, normal, 191.097, 1126, -3.47251
-1.72699, -10.0929, red, -5.04645, -19.7693, no, large, 256.141, 1091, -0.863493
-4.94135, -13.7953, brown, -6.89764, -2.17294, yes, normal, 165.52, 1039, -7.11429
-2.12896, -11.5536, brown, -6.77679, -15.3664, no, normal, 311.389, 1152, -17.4924
0.30634, -10.7211, green, -5.36057, -15.85, no, normal, 419.422, 1197, -15.5437
3.94104, -10.9779, red, -5.48895, -4.88908, no, large, 101.375, 1068, 1.97052
1.1662, -13.5458, red, -6.77289, -32.7954, no, large, 374.022, 1099, 0.683101
1.92435, -13.9216, brown, -6.96078, -21.1189, no, normal, 280.062, 1081, -19.1946
0.387225, -13.2787, red, 0.193613, 0.590036, yes, normal, 496.513, 1123, 0.193613
0.892636, -11.973, green, -5.98649, -12.1062, no, normal, 194.862, 1017, -11.2136
3.73108, -10.352, brown, -6.17602, -6.09351, no, normal, 431.147, 1137, -2.36243

```

Figura 4.4: Exemplo de arquivo .data na sintaxe padrão.

para os formatos de entrada dos algoritmos *C4.5*, *C4.5 rules*, *C5.0*, Cubist, *CN2*, Newid, RT, RETIS, CART, Weka, SNNS e SVMTorch (Batista, 2002; Dosualdo & Rezende, 2003a).

Ferramentas e Sistemas Implementados no Discover

De maneira geral, o Ambiente DISCOVER pode ser entendido como um conjunto de métodos aplicados sobre dados e textos ou sobre o conhecimento extraído a partir dos mesmos. Desse modo, é muito importante que o DISCOVER ofereça uma base sólida para manipular dados e conhecimento (composta por sintaxes padrão), e bibliotecas que ofereçam um conjunto de funcionalidades básicas de manipulação de dados e conhecimento.

Entre as diversas ferramentas e sistemas implementados para o Ambiente DISCOVER podem ser citados:

- o sistema XRULER, no qual um conjunto de exemplos é dividido em amostras de treinamento e teste. As amostras de treinamento são submetidas a diferentes indutores e os classificadores obtidos são convertidos para o formato padrão de regras de classificação. Então, a partir de um subconjunto dessas regras que cobrem os exemplos obtém-se um classificador simbólico final (Baranauskas, 2001);
- a biblioteca de classes DOL (*DISCOVER Object Library*), que implementa a manipulação de dados na sintaxe padrão. Essa biblioteca apresenta métodos para conversão de dados na sintaxe padrão para os formatos de entrada dos sistemas de aprendizado: *C4.5*, *C4.5 rules*, *C5.0/See5*, Cubist, *CN2*, Ripper, Trepan, Newid, RT, *M5*, RETIS, CART, Weka, SNNS, SVMTorch, Apriori, MagnumOpus e MineSetTM. Além disso, a biblioteca fornece vários métodos para pré-processar os dados, entre os quais, podem ser citados: cálculo de estatísticas descritivas básicas, remoção, troca de posição e criação de novos atributos a partir de expressões, embaralhamento, amostragem aleatória e complementar, entre outros (Batista, 2003; Batista & Monard, 2003);

- uma biblioteca que transforma os classificadores simbólicos (induzidos pelos algoritmos ID3, C4.5, C4.5 rules, C5.0/See5, CN2, OC1, Ripper, T2 e MC4) para o formato padrão de regras de classificação do DISCOVER (Prati, Baranauskas, & Monard, 2001);
- uma biblioteca que transforma os regressores simbólicos (induzidos pelos algoritmos M5, RT, Cubist, CART e RETIS) para o formato padrão de regras de regressão do DISCOVER (Pugliesi, Dosualdo, & Rezende, 2003; Dosualdo, 2003);
- uma biblioteca que transforma as regras de associação (geradas pelo algoritmo *Apriori* e pelos softwares Weka, MagnumOpus e MineSetTM) para o formato padrão de regras de associação do DISCOVER (Melandra & Rezende, 2003);
- o Ambiente Computacional SNIFFER que gerencia avaliações e comparações experimentais de algoritmos de Aprendizado de Máquina. Entre suas principais funcionalidades podem ser citadas: gerenciamento de sintaxes dos sistemas de aprendizado, aplicação de métodos de *resampling* para estimar a taxa de erro verdadeira, recuperação das taxas de erro, cálculo e comparação de medidas de desempenho (Batista, 2003; Batista & Monard, 2003);
- a ferramenta PRETEXT, que realiza o pré-processamento de uma coleção de documentos utilizando a abordagem *bag-of-words*. Entre as principais funcionalidades da ferramenta, podem ser citadas: transformação de palavras escritas em português, espanhol ou inglês, em *stems*; criação da tabela atributo-valor no formato padrão do DISCOVER; redução da dimensionalidade do conjunto de atributos usando a lei de Zipf e os cortes de Luhn; utilização de indução construtiva na criação de novos atributos na tabela atributo-valor; entre outras (Matsubara, Martins, & Monard, 2003; Martins, 2003);
- o Ambiente Computacional DiPER que auxilia na avaliação de um modelo de regressão simbólico. Entre as principais funcionalidades do Ambiente DiPER podem ser citadas: conversão dos regressores gerados pelos algoritmos Cubist, RT, RETIS/RT, CART/RT, M5 Model e M5 Regression para o formato padrão de regras de regressão definido para o DISCOVER (Pugliesi, Dosualdo, & Rezende, 2003); aplicação de tratamento semântico ao conjunto de regras no formato padrão que também faz parte do DiPER; cálculo de medidas de precisão e comprehensibilidade para as regras de regressão; construção de uma tabela de contingência para regras de regressão utilizando pseudo-classes e cálculo de uma série de medidas derivadas dessa tabela; e por fim, realização de um teste de hipóteses para comparar os regressores induzidos por diferentes algoritmos de regressão sobre um mesmo conjunto de dados (Dosualdo, 2003; Dosualdo & Rezende, 2003b);

- o Ambiente RULEE, que viabiliza tanto a análise quanto a disponibilização de regras de classificação, regressão e associação. O desenvolvimento de um ambiente para exploração de regras que auxilie os especialistas do domínio e usuários finais na compreensão e identificação do conhecimento interessante é de grande relevância para o sucesso do processo de extração de conhecimento. Assim o desenvolvimento de um Ambiente para Exploração de Regras vem suprir essas deficiências por disponibilizar o conhecimento para facilitar o acesso aos usuários, além de agregar métodos para avaliar as regras quanto a precisão, comprehensibilidade ou grau de interesse, por exemplo (Paula, 2003);
- O Ambiente *ARInE*, que é a proposta de um sistema interativo de seleção de regras de associação, no qual o usuário define alguns parâmetros e o sistema retorna as regras selecionadas. Esse processo é repetido interativamente com o usuário, com variação dos parâmetros e aspectos de análise, até que seja identificado um conjunto de regras satisfatório para o usuário. Isso devido ao fato da técnica de regras de associação gerar um número elevado de regras, dificultando sobremaneira a interpretação do conjunto de regras pelo usuário (Melandra, 2002; Melanda & Rezende, 2003).
- *Framework* de integração de componentes e interface para o Ambiente DISCOVER. Dada as características de experimentação na pesquisa em descoberta de conhecimento, é extremamente útil ter um sistema que auxilie a configuração e a execução de experimentos. Utilizando a infra-estrutura dos componentes e a interface gráfica, que vem sendo desenvolvida por Geromini (2002), os pesquisadores poderão criar e testar efetivamente e eficientemente seus experimentos, o que pode resultar em uma significativa redução de esforço, além do reaproveitamento dos programas desenvolvidos. O *framework* também serve como uma base para o desenvolvimento tanto de novas ferramentas que explorem outra subáreas de pesquisa, quanto proporcionar uma arquitetura modular que permita sua própria expansão (Prati, 2003; Prati, Geromini, & Monard, 2003b,a,c).

Além dos trabalhos já implementados descritos aqui, cabe ressaltar que diversos outros trabalhos foram desenvolvidos ou estão em desenvolvimento pelos pesquisadores do LABIC, com funcionalidades para processamento de dados e textos, Aprendizado de Máquina e processamento de padrões, entre os quais podem ser citados: (Domingues, 2003; Pila, 2003; Bernardini, 2002; Gonçalves, 2002; Gomes, 2002; Imamura, 2001; Nagai, 2000; Lee, 2000; Caulkins, 2000; Horst, 1999; Rocha, 1999; Padilha, 1999; Félix, 1998).

4.1.3 Ferramentas de Regressão e Base de Dados

Nesta seção são apresentadas as ferramentas Cubist, Weka e RT, que foram utilizadas para a realização dos experimentos apresentados nos próximos capítulos. Também são descritas as bases de dados utilizadas. Vale ressaltar que tanto as ferramentas quanto as bases são para problemas de regressão.

Descrição das Ferramentas

A tarefa de qualquer algoritmo de regressão é estimar o valor do atributo meta de um novo exemplo baseado nos outros atributos. Os algoritmos de regressão descritos são o Cubist, RT e M5.

Cubist

O Cubist é uma ferramenta utilizada para a geração de modelos preditivos numéricos baseados em regras a partir de um conjunto de dados fornecido. Esses modelos são utilizados na predição de valores numéricos, ou seja, para problemas de regressão. Cada exemplo fornecido ao Cubist possui um atributo meta e os atributos preditores que são utilizados para predizer o valor desse atributo meta. O Cubist gera regras não ordenadas. Essas regras estão dispostas em ordem crescente do valor médio dos exemplos preditos.

A tarefa do Cubist é tentar estimar o valor do atributo-meta de um novo exemplo em função dos outros atributos. Para isso, ele constrói um modelo contendo uma ou mais regras, na qual cada regra é uma conjunção de condições associadas a uma expressão linear. Portanto, os modelos gerados pelo Cubist são modelos lineares *piecewise*. Porém, o Cubist pode também construir modelos múltiplos e pode combinar modelos baseados em regras com modelos baseados em exemplos.

Essa ferramenta está disponível para os sistemas operacionais Unix, Linux e Windows 98/Me/2000/XP. Outras características que favorecem o uso dessa ferramenta é a facilidade na sua utilização e na compreensão dos modelos gerados, além de suportar a análise de bases de dados com milhares de registros e dezenas ou centenas de atributos numéricos ou nominais (Rulequest-Research, 2001).

Weka

A ferramenta Weka³ (*Waikato Environment for Knowledge Analysis*) foi desenvolvida na Universidade de Waikato, na Nova Zelândia, implementada em Java e testada nos sistemas operacionais Linux, Windows e Macintosh (Witten & Frank, 1999). Ela pode ser

³<http://www.waikato.ac.nz/>

vista como uma coleção de algoritmos de Aprendizado de Máquina que podem ser utilizados para resolver problemas de Mineração de Dados do mundo real. Além de algoritmos para classificação, regressão, associação e *clustering*, a ferramenta Weka também fornece suporte às fases de pré-processamento e de visualização de dados.

No Weka, cada implementação de um algoritmo de aprendizado é representada por uma classe. A linguagem Java permite que as classes sejam organizadas em pacotes, que são simplesmente diretórios contendo uma coleção de classes relacionadas. Isso é bastante útil porque certos algoritmos compartilham muitas funcionalidades, e dessa forma, várias classes em um pacote podem ser utilizadas por mais de um algoritmo (Witten & Frank, 1999).

O pacote `weka.classifiers` contém implementações dos algoritmos de classificação e predição numérica do Weka. A classe `weka.classifiers.m5` é a classe que implementa o algoritmo *M5*. O *M5* é um algoritmo desenvolvido por Quinlan (Quinlan, 1992) para tratar atributos e classes contínuas. Estruturalmente, *M5* pertence à família TDIDT (*Top Down Induction Decision Trees*), mas com funções de regressão linear nos nós-folhas, ao invés de um valor categórico predizendo a classe. O algoritmo *M5* foi baseado no trabalho inicial de Breiman com o algoritmo CART. *M5* possui algumas vantagens frente a outras técnicas que manipulam atributos contínuos, por exemplo, a Regressão Linear clássica que impõe uma relação estritamente linear entre os dados. O algoritmo *M5* possui uma fase de particionamento, que divide o conjunto de dados; uma fase de poda, para reduzir o número de nós da árvore obtida; e uma fase adicional denominada *smoothing*, que tem como objetivo reduzir a grande diferença dos valores preditos entre os nós-folhas (Wang & Witten, 1997).

Da biblioteca Weka foi utilizado um algoritmo modificado do *M5*, o *M5 Prime* (Witten & Frank, 1999). Além de utilizar *M5 Prime* para gerar *Model Trees*, pode-se utilizá-lo para gerar *Regression Trees*, isto é, árvores de decisão que possuem nos nós folhas valores reais preditos a partir das funções lineares da *Model Tree* gerada.

RT

O RT (*Regression Trees*) foi desenvolvido por Luis Fernando Raínho Alves Torgo, do Departamento de Ciências de Computadores da Faculdade de Ciências da Universidade do Porto (Torgo, 2001). Esse algoritmo permite obter modelos de regressão em diferentes tipos de representação baseados em um conjunto de exemplos fornecido. Alguns tipos de modelos que podem ser obtidos são:

- modelo baseado em árvores de regressão, em que os nós-folhas são rotulados com a média dos valores mapeados;

- modelo paramétrico global;
- modelo local, por meio do algoritmo *K-Nearest Neighbour*.

O RT permite ainda, emular os algoritmos de regressão RETIS e CART, descritos a seguir.

O RETIS (*Regression Tree Induction System*) é um sistema utilizado para induzir árvores de regressão desenvolvido por Aram Karalic. As árvores de regressão são utilizadas para modelar uma relação linear *piecewise* entre atributos nominais ou contínuos e um atributo-meta contínuo (Karalic, 1995). Esse algoritmo gera uma *Model Tree*.

O algoritmo CART (*Classification And Regression Trees*) foi desenvolvido por Breiman, Friedman, Olshen e Stone (Breiman, Friedman, Stone, & Olshen, 1984).

O algoritmo CART permite a construção de árvores de decisão e árvores de regressão (*Regression Trees*) realizando um particionamento recursivo binário do conjunto de dados e associando a cada nó-folha da árvore uma classe, no caso das árvores de decisão, ou um valor contínuo, no caso das árvores de regressão.

Descrição das Bases de Dados

Nesta seção são descritas todas as bases de dados utilizadas nos experimentos, sendo a maioria delas obtidas no Repositório de Dados da UCI (Blake & Merz, 1998) e do DELVE (Delve, 2003).

Abalone

O objetivo dos experimentos com a base de dados Abalone é predizer a idade de um abalone (tipo de molusco da califórnia) a partir de medidas físicas. A idade do molusco é determinada pelo número de anéis existentes no interior do mesmo e, para se contar tais anéis, é preciso cortar o molusco através do cone, colorindo-o e contar o número de anéis por meio de um microscópio, o que é uma tarefa lenta e cansativa. Com essa base de dados espera-se predizer o número de anéis, que determina a idade, a partir de outras medidas, mais fáceis de obter, como peso e diâmetro.

Como pode ser visto na Tabela 4.4, a base de dados Abalone possui 4.177 exemplos, com 7 atributos contínuos e 1 nominal, além da classe (número de anéis), que é um valor numérico. Por ter sido retirada do Repositório da UCI, não foi necessário realizar limpeza nos dados antes de iniciar o processo de extração de padrões.

Tabela 4.4: Descrição da base de dados Abalone.

Nº de Exemplos	4.177
Nº de Atributos Contínuos	7
Nº de Atributos Nominais	1

Auto-MPG

A base de dados Auto-MPG tem como objetivo prever o consumo de combustível de um carro na cidade em milhas por galão (mpg), com base em características como atributos cilindros, potência do motor, peso, aceleração e ano de fabricação.

Quando retirada do Repositório da UCI, essa base de dados possuía um atributo, (*car model*), cujo valor era único para todos os exemplos. Assim, visando melhorar a extração de padrões, esse atributo foi excluído. A descrição dos dados utilizados nesses experimentos é apresentada na Tabela 4.5.

Tabela 4.5: Descrição da base de dados Auto-MGP.

Nº de Exemplos	398
Nº de Atributos Contínuos	4
Nº de Atributos Nominais	3

Portanto, nos experimentos apresentados nesta tese, a base Auto-MPG possui 7 atributos, sendo 4 contínuos e 3 nominais, além da classe, que representa o consumo em milhas por galão.

Housing

O valor de moradias em subúrbios de Boston pode ser predito com a base de dados Housing. Cada exemplo dessa base descreve uma área com características como taxa de crime, concentração de óxido nítrico, acesso a rodovias, número médio de quartos por moradia, etc. Na Tabela 4.6 é apresentada a descrição dos dados dessa base.

Tabela 4.6: Descrição da base de dados Housing.

Nº de Exemplos	506
Nº de Atributos Contínuos	12
Nº de Atributos Nominais	1

Nesses experimentos foi utilizada a base de dados Housing sem modificações, possuindo 506 exemplos, 12 atributos contínuos, 1 nominal e a classe numérica, representando o valor das moradias.

CPU Performance

Os experimentos com a base de dados CPU Performance têm como objetivo prever o poder de processamento de um computador, com base em características como quantidade mínima e máxima de memória principal. Assim, cada exemplo representa uma configuração diferente de computador. A descrição dos dados utilizados nesses experimentos é apresentada na Tabela 4.7.

Tabela 4.7: Descrição da base de dados CPU Performance.

Nº de Exemplos	209
Nº de Atributos Contínuos	6
Nº de Atributos Nominais	0

Assim, a base CPU Performance possui 6 atributos, sendo todos contínuos, além da classe, que representa o desempenho da CPU (*Central Processing Unit*).

Pole

A base de dados Pole representa uma aplicação comercial descrita em (Weiss & Indurkhyá, 1995). Os dados descrevem um problema de telecomunicação. Porém, não há muita informação disponível sobre essa base. Na Tabela 4.8 é apresentada a descrição dos dados dessa base.

Tabela 4.8: Descrição da base de dados Pole.

Nº de Exemplos	15.000
Nº de Atributos Contínuos	48
Nº de Atributos Nominais	0

Assim, a base de dados Pole possui 48 atributos, sendo todos contínuos, além da classe também contínua.

Servo

Esta base de dados se refere a problemas de controle de robô. Os dados são uma simulação de um sistema de “servo mecanismo” que envolve um amplificador, um motor, uma frente porca/parafuso, e um trilho. Em todos os exemplos, o valor de saída é certamente quase um tempo de elevação, ou o tempo necessário para o sistema responder a uma mudança de passo em uma posição fixa. Esta é uma coleção interessante de dados. Os dados cobrem um fenômeno não-linear – predizendo o tempo de elevação de um mecanismo do “servo” em termos de quatro variáveis. Esta base de dados está contida no repositório

da UCI, e possui 167 exemplos e 4 atributos nominais. Na Tabela 4.9 é apresentada a descrição dos dados dessa base.

Tabela 4.9: Descrição da base de dados Servo.

Nº de Exemplos	167
Nº de Atributos Contínuos	0
Nº de Atributos Nominais	4

Computer Activity

A base de dados “Computer Activity” é uma coleção de medidas das atividade de um sistema computacional. Os dados foram coletados de uma Sun Sparcstation 20/712 com 128 Mbytes de memória rodando em um departamento universitário multi-usuário. Os usuários faziam uma grande variedade de tarefas que variavam desde acesso a internet, edição de arquivos ou execução de programas com grande utilização de CPU. Os dados foram coletados continuamente em duas ocasiões separadas. Em ambas ocasiões, os dados das atividades do sistema foram coletados a cada 5 segundos. O conjunto de dados final é obtido de ambas as ocasiões com números iguais de observações provenientes de cada período coletado. Esta base de dados está contida no repositório de dados DELVE, e possui 8192 exemplos e 22 atributos contínuos. Na Tabela 4.10 é apresentada a descrição dos dados dessa base.

Tabela 4.10: Descrição da base de dados Computer Activity.

Nº de Exemplos	8.192
Nº de Atributos Contínuos	22
Nº de Atributos Nominais	0

Kinematics

Esta base de dados está relacionada a cinemática precoce de um braço de robô com 8 graus de liberdade. Dentre as variações existentes dessa base, utilizou-se a 8nm que é altamente não-linear e com média quantidade de ruído. Esta base de dados está contida no repositório de dados DELVE, e possui 8192 exemplos e 8 atributos contínuos. Na Tabela 4.11 é apresentada a descrição dos dados dessa base.

Tabela 4.11: Descrição da base de dados Kinematics.

Nº de Exemplos	8.192
Nº de Atributos Contínuos	8
Nº de Atributos Nominais	0

Ailerons

Esta base de dados endereça um problema de controle, ou seja, pilotando uma aeronave F16. Os atributos descrevem o status do avião, enquanto o objetivo é predizer a ação de controle nas *ailerons* das aeronaves. Esta base foi obtida dos experimentos de Rui Camacho (rcamacho@garfield.fe.up.pt), e possui 13750 exemplos e 40 atributos contínuos. Na Tabela 4.12 é apresentada a descrição dos dados dessa base.

Tabela 4.12: Descrição da base de dados Ailerons.

Nº de Exemplos	13.750
Nº de Atributos Contínuos	40
Nº de Atributos Nominais	0

Delta Ailerons

Esta base de dados também é obtida para a tarefa de controle de *ailerons* de uma aeronave F16, entretanto o atributo meta e os atributos de entrada são diferentes da base Ailerons. Neste caso, o atributo meta é uma variação ao invés de um valor absoluto, e há uma pré-seleção de atributos. Esta base também foi obtida dos experimentos de Rui Camacho (rcamacho@garfield.fe.up.pt), e possui 7129 exemplos e 6 atributos contínuos. Na Tabela 4.13 é apresentada a descrição dos dados dessa base.

Tabela 4.13: Descrição da base de dados Delta Ailerons.

Nº de Exemplos	7.129
Nº de Atributos Contínuos	6
Nº de Atributos Nominais	0

Elevators

Este conjunto de dados foi também obtido a partir da tarefa de controlar uma aeronave F16, embora a variável meta e os atributos sejam diferentes do domínio da base Ailerons. Neste caso a variável meta é relacionada a uma ação obtida dos elevadores da aeronave. Esta base foi obtida dos experimentos de Rui Camacho (rcamacho@garfield.fe.up.pt), e possui 16559 exemplos e 18 atributos contínuos. Na Tabela 4.14 é apresentada a descrição dos dados dessa base.

Census

Esta base de dados foi projetada com base nos dados fornecidos por *US Census Bureau* (<http://www.census.gov>). Os dados foram colecionados como parte do censo norte

Tabela 4.14: Descrição da base de dados Elevators.

Nº de Exemplos	16.559
Nº de Atributos Contínuos	18
Nº de Atributos Nominais	0

americano de 1990. Foram obtidos os dados de todos os estados. A maioria dos cálculos foram alterados para proporções apropriadas. Estes dados visam predizer o preço médio das casas na região baseado na composição demográfica e um estado de mercado de casas na região. Esta base de dados está contida no repositório de dados DELVE, e possui 22.784 exemplos e 8 atributos contínuos. Na Tabela 4.15 é apresentada a descrição dos dados dessa base.

Tabela 4.15: Descrição da base de dados Census.

Nº de Exemplos	22.784
Nº de Atributos Contínuos	8
Nº de Atributos Nominais	0

Wisconsin Breast Cancer

A base de dados Wisconsin Breast Cancer visa predizer o tempo de recorrência do câncer de mama. Cada exemplo representa os dados de *follow-up* para um caso de câncer de mama. Esses são pacientes consecutivos vistos pelo médico Dr. Wolberg desde 1984, e inclui apenas os casos exibindo câncer de mama invasivo e nenhuma evidência de metastase distante no momento do diagnóstico. As primeiras 30 características são calculadas de uma imagem digitalizada de uma fina agulha de aspiração (*fine needle aspirate (FNA)*) de uma parte da mama. Elas descrevem características dos núcleos celulares presentes na imagem.

Esta base de dados está contida no repositório da UCI, e possui 194 exemplos e 32 atributos contínuos. Na Tabela 4.16 é apresentada a descrição dos dados dessa base.

Tabela 4.16: Descrição da base de dados Wisconsin Breast Cancer.

Nº de Exemplos	194
Nº de Atributos Contínuos	32
Nº de Atributos Nominais	0

4.2 Sintaxe Padrão para Representar Regras de Regressão

O foco desta seção é a sintaxe padrão para representação de regras de regressão adequada para englobar o tipo de representação da hipótese induzida por vários algoritmos de regressão (Puggiesi, Dosualdo, & Rezende, 2003; Dosualdo & Rezende, 2003a). Além da definição da sintaxe para regras de regressão, foram desenvolvidos *scripts* para conversão da saída dos algoritmos M5, RT, Cubist, CART e RETIS para a sintaxe padrão de regressão.

O processo de padronização dos regressores funciona da seguinte forma: primeiramente, o conjunto de dados é submetido a algum dos possíveis algoritmos de regressão, e a saída por eles gerada é salva em um arquivo. Em seguida, é realizada a conversão dessa saída do algoritmo para o formato padrão de regras de regressão, para gerar então um arquivo que contenha o regressor no formato padrão de regras de regressão definido para o Ambiente DISCOVER.

Independentemente do algoritmo de regressão utilizado para induzir a hipótese, o conjunto de regras padronizado obtido a partir dessa hipótese possui o formato apresentado na Figura 4.5.

```
[R0001]      IF <condition_11>
              AND <condition_12>
          THEN <conclusion_1>

[R0002]      IF <condition_21>
              AND <condition_22>
              AND <condition_23>
          THEN <conclusion_2>

[R0003]      IF <condition_3>
          THEN <conclusion_3>

[R0004]      IF <condition_41>
              AND <condition_42>
              AND <condition_43>
              AND <condition_44>
          THEN <conclusion_4>
```

Figura 4.5: Exemplo de um conjunto de regras de regressão padronizadas.

Ao serem executados, os algoritmos possuem entrada de dados diferentes e também produzem saídas em formatos distintos. Assim, foi necessário desenvolver *scripts* de conversão das hipóteses de cada algoritmo de regressão para a sintaxe padrão de regras de regressão.

4.2.1 Gramática da Sintaxe Padrão

Os métodos de regressão simbólicos geralmente induzem hipóteses no formato de regras e árvores. As regras são diferentes das árvores, e mesmo as árvores geradas por diferentes algoritmos são distintas uma das outras: elas diferem no modo como é particionado o conjunto de dados, na escolha dos atributos e valores utilizados nos nós internos da árvore, no modo como predizem o valor do atributo-meta em seus nós-folha (por exemplo, uma *Regression Tree* calcula a média dos exemplos e uma *Model Tree* possui uma equação linear), nos mecanismos de poda e no formato em que a árvore pode ser visualizada.

Devido aos diferentes formatos de representação das hipóteses geradas por esses algoritmos, qualquer medida ou estatística que precise ser calculada sobre as hipóteses tem de ser implementada várias vezes, uma para cada formato de hipótese. Dessa maneira, é definida uma sintaxe padrão para representação de regras de regressão para o Ambiente DISCOVER. Baseado nessa sintaxe, padroniza-se os regressores convertendo-os para um formato único em que as regras são não ordenadas. Além de facilitar a implementação de medidas e cálculos a serem efetuados sobre as hipóteses, isso permite comparar os regressores gerados pelos diferentes algoritmos.

Para possibilitar a integração e avaliação de conhecimentos extraídos por diferentes algoritmos foi especificada a gramática a ser utilizada na conversão das regras geradas pelos algoritmos de regressão para a sintaxe padrão de regras de regressão. A definição de tal gramática levou em consideração a padronização definida pelo grupo que trabalha com o Ambiente DISCOVER.

A gramática, que define a sintaxe padrão para regras de regressão, é definida como uma quádrupla $G_1 = (V_{n1}, V_{t1}, P_1, S_1)$:

S_1 . <regra regressão>

V_{n1} : { <regra regressão>, <nro regra>, <regra_r>, <condições>, <atrib_meta>, <regressão>, <fator>, <termo>, <comparação>, <expressão>, <atributo>, <operador relacional>, <valor numérico>, <valor categórico>, <conjunto>, <identificador>, <soma>, <constante>, <sinal>, <equação linear>, <dígitos>, <dígito>, <letra> }

V_{t1} : { IF, THEN, OR, AND, IN, {, }, <=,>=, =, <,>, <>, +/-, [,], +, -, ., ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, - }

P_1 . é representado na Figura 4.6 e descrito usando a notação EBNF em que:

$$[\alpha] = \alpha | \lambda$$

$\{\alpha\}$ = repetição da cadeia α zero ou mais vezes

$\alpha | \beta$ = α ou β devem ser escolhidos

Não terminais aparecem entre < e >, e terminais aparecem em negrito.

Por exemplo, a regra apresentada na Figura 4.7 foi gerada pelo algoritmo de regressão Cubist e armazenada, juntamente com as demais, em um arquivo-texto.

As regras geradas são convertidas para a sintaxe padrão de regras de regressão utilizando a gramática de conversão apresentada anteriormente. Um exemplo da regra apresentada na Figura 4.7 convertida para essa sintaxe pode ser visto na Figura 4.8. Todas as regras convertidas também são armazenadas em um outro arquivo-texto.

Observe que as informações fornecidas pela regra original (por exemplo: [898 cases, mean 7.8, range 1 to 21, est err 1.2]) não são armazenadas na sintaxe padrão, uma vez que essas medidas normalmente são distintas para cada algoritmo de regressão. As informações necessárias à avaliação de regras são calculadas de forma padronizada e independem do algoritmo que as geraram.

Além da sintaxe padrão apresentada, também foi definida uma gramática para a sintaxe padrão estendida para regras de regressão, a qual armazena também a frequência relativa dos valores da matriz de contingência e o número total de exemplos, as medidas de precisão (MAD e MSE, apresentadas na Seção 5.2) e outros valores importantes para o cálculo das medidas.

Na Figura 4.9 é apresentado outro exemplo contendo regras de regressão escritas na sintaxe padrão, e na Figura 4.10 é apresentado esse exemplo na sintaxe padrão estendida a qual é a sintaxe padrão, com os valores da tabela de contingência e outras medidas que são imprescindíveis para cada tipo de problema tratado. Em regressão, a sintaxe estendida engloba a matriz de contingência e duas medidas de erro, a MAD e MSE. Isso devido ao fato da importância dessas medidas estarem disponíveis para consultas diretas.

Para cada regra na sintaxe padrão estendida para regras de regressão, os elementos entre os primeiros colchetes referem-se, respectivamente, a $p(BH)$, $p(BH)$, $p(\overline{BH})$, $p(\overline{BH})$, n . Entre os segundos colchetes estão os valores da medidas de erro MAD e MSE, respectivamente. Observe que a primeira produção da gramática da sintaxe padrão estendida é semelhante da gramática G_1 com a inclusão do símbolo não terminal <valores> no final da mesma.

```

<regra regressão> ::= <nro regra> <regra_r>
<regra_r> ::= [ IF <condições> THEN ] <atributo> = <regressão>
<condições> ::= <fator> | <fator> OR <condições>
<fator> ::= <termo> | <termo> AND <fator>
<termo> ::= <comparação> | <expressão>
<comparação> ::= <atributo> <operador relacional> <valor numérico>
                | <atributo> <operador relacional> <valor categórico>
                | <atributo> IN { <conjunto> }
<atributo> ::= <identificador>
<expressão> ::= <soma> <operador relacional> <valor numérico>
<soma> ::= <constante> <atributo>
            | <constante> <atributo> <sinal> <soma>
<operador relacional> ::= <= | >= | = | < | > | <>
<regressão> ::= <equação linear> | <valor numérico>
                | <valor numérico> +/- <valor numérico>
<equação linear> ::= <constante> <sinal> <soma>
<conjunto> ::= <valor categórico> | <valor categórico> , <conjunto>
<nro regra> ::= [ R <dígito> <dígito> <dígito> <dígito> ]
<sinal> ::= + | -
<constante> ::= <valor numérico>
<valor numérico> ::= [ <sinal> ] <dígitos> [ . <dígitos> ]
                    | [ <sinal> ] <dígitos> [ . <dígitos> ] e <sinal> <dígitos>
<dígitos> ::= <dígito> { <dígito> }
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<valor categórico> ::= <identificador>
<identificador> ::= <letra> { <letra> | <dígito> }
                    | <dígito> { <letra> | <dígito> }
<letra> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o
            | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D
            | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S
            | T | U | V | W | X | Y | Z | _

```

Figura 4.6: Produções da gramática da sintaxe padrão de regras de regressão.

```

Rule 1: [898 cases, mean 7.8, range 1 to 21, est err 1.2]
if
  sex = I
then
  rings = 2.7 + 15.3 shell_weight - 7.1 shucked_weight + 8.8 diameter
    + 15 height + 0.2 whole_weight

```

Figura 4.7: Regra de regressão gerada pelo algoritmo Cubist.

```

[R0001] IF sex IN {I}
      THEN rings = 2.7 +15.3 shell_weight -7.1 shucked_weight
           +8.8 diameter +15 height +0.2 whole_weight

```

Figura 4.8: Regra de regressão da Figura 4.7 na sintaxe padrão.

```

Standard Regression Rules Converter      Copyright (c) Daniel Gomes Dosualdo
Inducer: RT                            Input File: mv_RT.out
Date: Fri Dec 19 21:38:22 2003          Output File: mv_RT.rules

[R0001] IF x6 <= -32.274651
      AND x8 IN { normal }
      AND x1 <= -3.674200
      THEN y = -38.512024 +/- 0.336

[R0002] IF x6 <= -32.274651
      AND x8 IN { normal }
      AND x1 <= -0.598827
      AND x1 > -3.674200
      THEN y = -36.037579 +/- 0.230

:
:
```

Figura 4.9: Exemplo de arquivo de regras de regressão na sintaxe padrão.

A gramática, que define a sintaxe padrão estendida para regras de regressão, é definida como uma quádrupla $G_2 = (V_{n2}, V_{t2}, P_2, S_2)$:

$S_2: \langle \text{regra regressão} \rangle$

$V_{n2}: V_{n1} \cup \{ \langle \text{valores} \rangle, \langle \text{matriz contingência} \rangle, \langle \text{precisão} \rangle \}$

$V_{t2} : V_{t1}$

$P_2 : (P_1 - \{\text{<regra regressão>}\}) \cup P$. P são as produções representadas na Figura 4.11.

```
Measures of Regression Rules          Copyright (c) Jaqueline Brigladori Puggiesi
Inducer: RT                         Input File: mv_RT.rules
Date: Sun Dec 21 00:31:35 2003        Output File: mv_RT.saida
```

```
[R0001] IF x6 <= -32.274651
      AND x8 IN { normal }
      AND x1 <= -3.674200
      THEN y = -38.512024 +/- 0.336
[ 0, 0.00232177894048398, 0.997678221059516, 0, 30580 ] [1.122, 1.83112297461408]

[R0002] IF x6 <= -32.274651
      AND x8 IN { normal }
      AND x1 <= -0.598827
      AND x1 > -3.674200
      THEN y = -36.037579 +/- 0.230
[ 0, 0.00516677567037279, 0.994833224329627, 0, 30580 ] [1.150, 2.0087059207443]

:
:
```

Figura 4.10: Exemplo de arquivo de regras de regressão na sintaxe padrão estendida.

```
<regra regressão> ::= <nro regra> <regra_r> <valores>
<valores> ::= <matriz contingência> <precisão>
<matriz contingência> ::= [ <valor numérico> ,
                           <valor numérico> , <valor numérico> ,
                           <valor numérico> , <valor numérico> ]
<precisão> ::= [ <valor numérico> , <valor numérico> ]
```

Figura 4.11: Produções que completam a gramática da sintaxe padrão de regras de regressão estendida.

4.2.2 Conversão dos Regressores para o Formato Padrão

Para converter as regras e árvores de regressão para a sintaxe padrão de regras de regressão foram implementados conversores na linguagem PERL, sendo que tanto as regras de regressão quanto árvores de regressão, independente do formato original, são convertidas para a mesma sintaxe de regras de regressão. No momento, estão implementados os *scripts* para conversão da saída dos seguintes algoritmos de regressão: M5, RT, Cubist, CART e RETIS.

Os algoritmos de regressão simbólicos utilizados no DISCOVER, juntamente com a forma como representam suas hipóteses, são apresentados na Tabela 4.17.

Tabela 4.17: Algoritmos de regressão utilizados.

Algoritmo	Forma de Representação
Cubist	Conjunto de regras de regressão
M5/ Weka	<i>Model Tree</i> ou <i>Regression Tree</i>
RT	<i>Regression Tree</i> com desvios-padrões
RETIS / RT	<i>Model Tree</i>
CART / RT	<i>Regression Tree</i> com desvios-padrões

Um esquema sobre os conversores de dados de algoritmos de regressão pode ser visualizado na Figura 4.12.

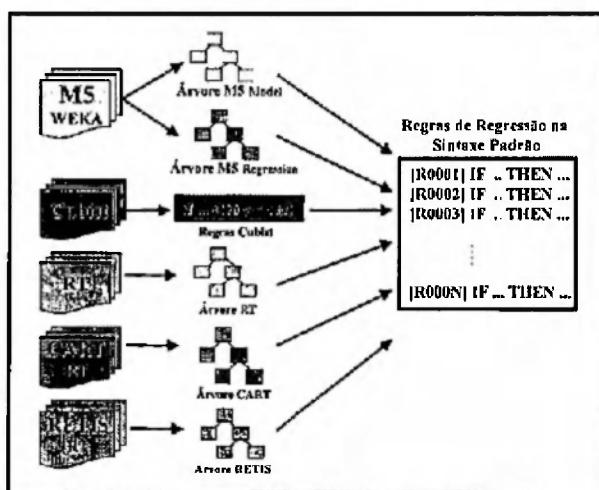


Figura 4.12: Conversão das saídas dos algoritmos de regressão utilizados no DISCOVER e no REPPE.

Na Seção 4.1.3 foi apresentada uma descrição dos sistemas e algoritmos de regressão utilizados, para os quais foram implementados conversores. Os sistemas RT e Weka são *freeware*, e com relação ao Cubist, o LABIC possui uma versão do mesmo.

Depois de executados os algoritmos de regressão, converte-se os diferentes modelos gerados para a sintaxe padrão de regras de regressão. Na Figura 4.13 são apresentadas algumas regras geradas pelo algoritmo Cubist utilizando a base de dados Abalone e na Figura 4.14 pode-se visualizar essas mesmas regras convertidas para o formato padrão de regras de regressão.

Na Figura 4.15 é apresentada a árvore de regressão gerada pelo algoritmo RT utilizando a base de dados Auto-MPG, e as regras geradas na sintaxe padrão de regras de regressão a partir dessa árvore podem ser visualizadas na Figura 4.16.

```

Cubist [Release 1.10] Mon Mar 31 16:29:56 2003
-----
Options:
  Application 'abalone'

Target attribute 'rings'

Read 3762 cases (9 attributes) from abalone.data

Model:

Rule 1: [42 cases, mean 5.7, range 3 to 10, est err 1.0]

if
  sex in {M, F}
  diameter <= 0.23
then
  rings = 3.8 + 67.5 shell_weight + 34.2 diameter - 25.8 length
    - 13.2 shucked_weight + 5.1 whole_weight - 4.6 viscera_weight
    + 4 height

Rule 2: [809 cases, mean 6.9, range 1 to 16, est err 1.0]

if
  sex = I
  shell_weight <= 0.1535
then
  rings = 2.7 + 15.3 shell_weight - 6.5 shucked_weight + 8.8 diameter
    + 14 height

```

Figura 4.13: Algumas regras geradas pelo Cubist utilizando a base de dados Abalone.

```

Standard Regression Rules Conversor Copyright (c) Daniel Gomes Dosualdo
Inducer: cubist           Input File: abalone.tmp.rules
Date: Mon Mar 31 16:29:59 2003      Output File: abalone.tmp.Rules

[R0001] IF diameter <= 0.23
  AND sex IN {M, F}
  THEN rings = 3.8 +67.5 shell_weight +34.2 diameter -25.8 length
    -13.2 shucked_weight +5.1 whole_weight
    -4.6 viscera_weight +4 height

[R0002] IF shell_weight <= 0.1535
  AND sex IN {I}
  THEN rings = 2.7 +15.3 shell_weight -6.5 shucked_weight
    +8.8 diameter +14 height

```

Figura 4.14: Regras da Figura 4.13 na sintaxe padrão de regras de regressão.

```
*****
RT version 4.1.0 - RESEARCH VERSION
*****
(c) Copyright Luis Torgo, All Rights Reserved.          05-Jul-01
Problem with 8 attributes.
360 exemplars were loaded.

SAVING THE UNPRUNED TREE ON FILE mpg.data.tree

PRUNING THE TREE...DONE.
SAVING THE PRUNED TREE ON FILE mpg.data.pruned

RESULTING TREE :
cylinders In { 8  6  3 }      (mpg = 22.80)
True   (181 of 360)
    displacement <= 284.50  (mpg = 16.85)
    True   (84 of 181)
        LEAF :: mpg = 19.47 +/- 0.634 [15.00..32.70]
    False  (97 of 181)
        LEAF :: mpg = 14.58 +/- 0.472 [9.00..23.00]
False  (179 of 360)
    horsepower <= 70.50      (mpg = 28.82)
    True   (66 of 179)
        LEAF :: mpg = 33.13 +/- 1.304 [23.00..46.60]
    False  (113 of 179)
        model In { 70 71 72 73 74 75 76 77 78 } (mpg = 26.30)
        True   (83 of 113)
            LEAF :: mpg = 24.81 +/- 0.705 [18.00..33.50]
        False  (30 of 113)
            LEAF :: mpg = 30.44 +/- 1.533 [22.30..41.60]

NUMBER OF NODES 9 [5 are leaves]
```

Figura 4.15: Conjunto de regras geradas pelo RT utilizando a base de dados Auto-MPG.

Standard Regression Rules ConverSOR Inducer: rt Date: Mon Mar 31 22:31:12 2003	Copyright (c) Daniel Gomes Dosualdo Input File: mpg.out Output File: mpg.rules
--	--

```
[R0001] IF cylinders IN {8, 6, 3}
      AND displacement <= 284.50
      THEN mpg = 19.47 +/- 0.634

[R0002] IF cylinders IN {8, 6, 3}
      AND displacement > 284.50
      THEN mpg = 14.58 +/- 0.472

[R0003] IF cylinders IN {4, 5}
      AND horsepower <= 70.50
      THEN mpg = 33.13 +/- 1.304

[R0004] IF cylinders IN {4, 5}
      AND horsepower > 70.50
      AND model IN {70, 71, 72, 73, 74, 75, 76, 77, 78}
      THEN mpg = 24.81 +/- 0.705

[R0005] IF cylinders IN {4, 5}
      AND horsepower > 70.50
      AND model IN {79, 80, 81, 82}
      THEN mpg = 30.44 +/- 1.533
```

Figura 4.16: Conjunto de regras da Figura 4.15 na sintaxe padrão de regras de regressão.

Na Figura 4.17 é apresentada a *Model Tree* gerada pelo algoritmo $M5$ do Weka utilizando a base de dados Housing. Na Figura 4.18 pode-se visualizar algumas regras que foram convertidas a partir dessa árvore para o formato padrão de regras de regressão.

```
Pruned training model tree:

RM <= 6.68 :
| DIS <= 1.98 :
| | LSTAT <= 14.6 : LM1 (15/81.1%)
| | LSTAT > 14.6 : LM2 (57/33.6%)
| DIS > 1.98 : LM3 (275/34.8%)
RM > 6.68 :
| RM <= 7.44 :
| | INDUS <= 7.82 :
| | | RM <= 7.12 :
| | | | LSTAT <= 5.2 : LM4 (20/32.6%)
| | | | LSTAT > 5.2 : LM5 (20/45.9%)
| | | RM > 7.12 : LM6 (22/25.9%)
| | INDUS > 7.82 :
| | | NOX <= 0.589 : LM7 (8/18%)
| | | NOX > 0.589 :
| | | | NOX <= 0.651 : LM8 (4/14.7%)
| | | | NOX > 0.651 : LM9 (4/34.3%)
| RM > 7.44 : LM10 (30/49.7%)

Models at the leaves:
Smoothed (complex):

LM1: MedHouseVal = 75.7 - 0.129CRIM + 0.00632ZN + 0.441CHAS - 2.99NOX
      + 0.184RM - 22.4DIS + 0.0711RAD - 0.00231TAX
      - 0.118PTRATIO - 0.997LSTAT
LM2: MedHouseVal = 36.4 - 0.148CRIM + 0.00632ZN + 0.441CHAS - 2.99NOX
      + 0.184RM - 3.96DIS + 0.0711RAD - 0.00926TAX
      - 0.118PTRATIO - 0.354LSTAT
LM3: MedHouseVal = 20.2 - 0.0182CRIM + 0.0032ZN + 0.198CHAS - 1.32NOX
      + 3.81RM - 0.0448AGE - 0.707DIS + 0.225RAD - 0.0124TAX
      - 0.654PTRATIO - 0.205LSTAT
LM4: MedHouseVal = 0.633 - 0.0204CRIM + 0.00546ZN - 0.285INDUS + 0.278CHAS
      + 5.39NOX + 6.16RM - 0.00522AGE - 0.552DIS + 0.0432RAD
      - 0.00146TAX - 0.355PTRATIO - 0.61LSTAT
LM5: MedHouseVal = 22.7 - 0.0204CRIM + 0.00546ZN - 0.285INDUS + 0.278CHAS
      + 5.39NOX + 2.89RM - 0.0289AGE - 0.552DIS + 0.0432RAD
      - 0.00146TAX - 0.355PTRATIO - 0.61LSTAT
LM6: MedHouseVal = 46 - 0.0204CRIM + 0.00546ZN - 0.146INDUS + 0.278CHAS
      + 6.98NOX - 0.322RM - 0.0162AGE - 0.41DIS + 0.0432RAD
      - 0.00146TAX - 0.355PTRATIO - 0.662LSTAT
LM7: MedHouseVal = 37.9 - 0.0204CRIM + 0.00546ZN + 0.278CHAS + 1.79NOX
      + 1.5RM - 0.271DIS + 0.0432RAD - 0.00146TAX
      - 0.514PTRATIO - 1.28LSTAT
LM8: MedHouseVal = 39 - 0.0204CRIM + 0.00546ZN + 0.278CHAS + 7.87NOX
      + 1.5RM - 0.711DIS + 0.0432RAD - 0.00146TAX
      - 0.514PTRATIO - 1.52LSTAT
LM9: MedHouseVal = 37.4 - 0.0204CRIM + 0.00546ZN + 0.278CHAS + 7.87NOX
      + 1.5RM - 0.271DIS + 0.0432RAD - 0.00146TAX
      - 0.514PTRATIO - 1.52LSTAT
LM10: MedHouseVal = 24.8 - 6.19CRIM + 0.00546ZN + 0.649INDUS + 0.278CHAS
      - 1.76NOX + 5.82RM - 0.379DIS + 0.0432RAD - 0.00146TAX
      - 1.5PTRATIO - 0.492LSTAT
```

Figura 4.17: *Model Tree* gerada pelo $M5$ utilizando a base de dados Housing.

Standard Regression Rules Conversor Inducer: m5 Date: Mon Mar 31 17:13:32 2003	Copyright (c) Daniel Gomes Dosualdo Input File: housing.out Output File: housing.rules
--	--

```

[R0001] IF RM <= 6.68
  AND DIS <= 1.98
  AND LSTAT <= 14.6
  THEN MedHouseVal = 75.7 - 0.129CRIM + 0.00632ZN + 0.441CHAS - 2.99NOX
    + 0.184RM - 22.4DIS + 0.0711RAD - 0.00231TAX - 0.118PTRATIO - 0.997LSTAT

[R0002] IF RM <= 6.68
  AND DIS <= 1.98
  AND LSTAT > 14.6
  THEN MedHouseVal = 36.4 - 0.148CRIM + 0.00632ZN + 0.441CHAS - 2.99NOX
    + 0.184RM - 3.96DIS + 0.0711RAD - 0.00926TAX - 0.118PTRATIO - 0.354LSTAT
  .

[R0005] IF RM > 6.68
  AND RM <= 7.44
  AND INDUS <= 7.82
  AND RM <= 7.12
  AND LSTAT > 5.2
  THEN MedHouseVal = 22.7 - 0.0204CRIM + 0.00546ZN - 0.285INDUS + 0.278CHAS
    + 5.39NOX + 2.89RM - 0.0289AGE - 0.552DIS + 0.0432RAD
    - 0.00146TAX - 0.355PTRATIO - 0.61LSTAT

[R0006] IF RM > 6.68
  AND RM <= 7.44
  AND INDUS <= 7.82
  AND RM > 7.12
  THEN MedHouseVal = 46 - 0.0204CRIM + 0.00546ZN - 0.146INDUS + 0.278CHAS
    + 6.98NOX - 0.322RM - 0.0162AGE - 0.41DIS + 0.0432RAD
    - 0.00146TAX - 0.355PTRATIO - 0.662LSTAT
  .

[R0009] IF RM > 6.68
  AND RM <= 7.44
  AND INDUS > 7.82
  AND NOX > 0.589
  AND NOX > 0.651
  THEN MedHouseVal = 37.4 - 0.0204CRIM + 0.00546ZN + 0.278CHAS + 7.87NOX
    + 1.5RM - 0.271DIS + 0.0432RAD - 0.00146TAX - 0.514PTRATIO - 1.52LSTAT

[R0010] IF RM > 6.68
  AND RM > 7.44
  THEN MedHouseVal = 24.8 - 6.19CRIM + 0.00546ZN + 0.649INDUS + 0.278CHAS
    - 1.76NOX + 5.82RM - 0.379DIS + 0.0432RAD
    - 0.00146TAX - 1.5PTRATIO - 0.492LSTAT
  .

```

Figura 4.18: Algumas regras da Figura 4.17 na sintaxe padrão de regras de regressão.

4.3 Funcionalidades do Ambiente REPPE

As funcionalidades do Ambiente REPPE que foi proposto e implementado são apresentadas na Figura 4.19.

Como já citado, os métodos de regressão simbólicos geralmente induzem hipóteses no formato de regras e árvores. As regras são diferente das árvores, e mesmo as árvores

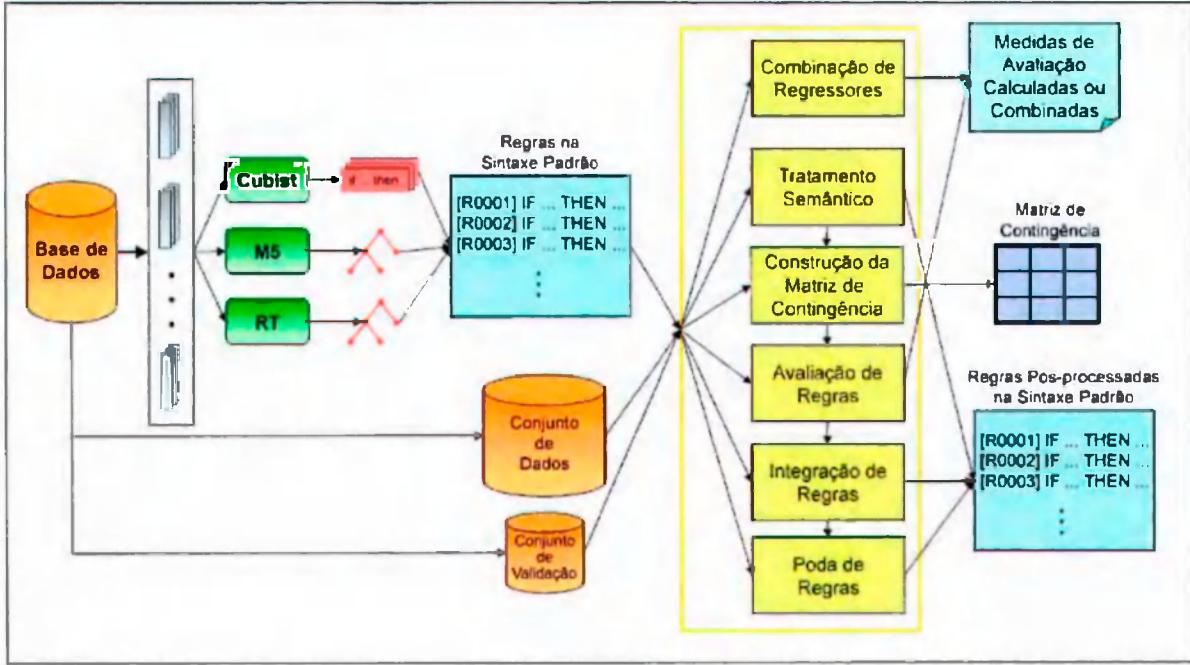


Figura 4.19: Funcionalidades do Ambiente *REPPE*.

geradas por diferentes algoritmos são distintas umas das outras. Devido a esses diferentes formatos de representação de hipóteses, qualquer medida ou estatística que precise ser calculada sobre as hipóteses tem de ser implementada várias vezes, uma para cada formato de hipótese. Dessa forma, foi definida e apresentada a sintaxe padrão para representação de regras de regressão para o Ambiente DISCOVER. Baseado nessa sintaxe, padroniza-se os regressores convertendo-os para um formato único em que as regras são não ordenadas. Além de facilitar a implementação de medidas e cálculos a serem efetuados sobre as hipóteses, isso permite comparar os regressores gerados pelos diferentes algoritmos.

Essa padronização também é importante para a integração e poda de regras extraídas por diferentes algoritmos de aprendizado.

A seguir são descritas as funcionalidades juntamente com os módulos que constituem o Ambiente *REPPE*.

4.3.1 Avaliação de Regras de Regressão no *REPPE*

A avaliação de conhecimento leva em consideração algumas particularidades dos diferentes tipos de problema. Assim, em regras de regressão o cálculo da cobertura da regra, ou seja, a verificação da classe estar correta ou não em problemas de regressão se torna mais complicada do que em classificação, ou seja, o cálculo dos valores relacionados à H e \bar{H} não são triviais. Essa verificação é imprescindível para o cálculo dos valores da matriz de contingência das regras (apresentada na Seção 4.1.1). Várias medidas, tanto de precisão quanto de qualidade do conhecimento obtido, são calculadas a partir dos valores da

matriz de contingência, como o *framework* descrito em (Lavrač, Flach, & Zupan, 1999). Com o cálculo da matriz de contingência para regras de regressão incrementa-se o número de medidas disponíveis para análise das regras. O cálculo da matriz de contingência para regras de regressão é realizado no Módulo Construção da Matriz de Contingência, descrito no Capítulo 5.

A fim de verificar se o conhecimento possui as características desejadas, após adquirido, um pós-processamento deve ser realizado para avaliar o desempenho e a qualidade do mesmo. Essa avaliação procura investigar a precisão e a representação do modelo, a complexidade e a dificuldade de entendimento do conhecimento extraído. Além disso, o pós-processamento também tem como objetivo a filtragem do que foi aprendido, eliminando o conhecimento espúrio, sem valor ou de situações óbvias guiadas pelo senso comum. O cálculo das medidas de avaliação são realizados no Módulo Avaliação de Regras (descrito no Capítulo 5) que tem implementado tanto as medidas derivadas da matriz de contingência, quanto as não derivadas da matriz de contingência, como MAD (*Mean Absolute Deviation*), MSE (*Mean Squared Error*) e variância.

Muitas vezes, depois que os regressores são convertidos para o formato padrão de regras de regressão, as regras que formam esse regressor podem apresentar condições desnecessárias, principalmente quando o conjunto de regras é convertido a partir de uma árvore. Isso acontece porque em muitos casos um mesmo atributo (numérico ou nominal) é utilizado para partitionar o conjunto de dados em diferentes níveis de uma árvore assumindo diferentes valores, o que faz com que uma condição sobreponha a outra, tanto no caso de atributos numéricos quanto nominais. Nesse sentido, foi desenvolvido o Módulo Tratamento Semântico com o objetivo de excluir as condições desnecessárias de cada regra.

Caso as regras tenham sido geradas a partir de uma árvore, geralmente, é necessário realizar um tratamento semântico a fim de excluir as condições desnecessárias de cada regra, pois em vários casos um mesmo atributo (numérico ou nominal) é utilizado em diferentes níveis de uma árvore assumindo diferentes valores, o que faz com que uma condição sobreponha a outra, tanto no caso de atributos numéricos quanto nominais.

O problema com os atributos numéricos acontece quando existem várias condições em uma regra sobre um mesmo atributo numérico com diferentes valores sobre um mesmo operador ($<$, \leq , $>$, \geq), fazendo com que tais condições se sobreponham. A solução nesse caso consiste em encontrar o intervalo correto para o atributo numérico na regra em que ele aparece em várias condições desse tipo.

Por exemplo, seja um conjunto de dados com x_1 , x_2 e x_3 atributos numéricos, e uma regra de regressão como apresentada a seguir:

```

IF  $x_1 \leq 0,17$ 
  AND  $x_1 \leq 0,10$ 
  AND  $x_1 \leq 0,08$ 
  AND  $x_1 > 0,03$ 
  AND  $x_2 > 3,84$ 
THEN  $y = 5,02 + 0,4x_1 + 1,19x_3$ 

```

Pode ser observado que as duas primeiras condições da regra não são necessárias, uma vez que a terceira condição ($x_1 \leq 0,08$) sobrepõe as duas primeiras. Dessa forma, após a aplicação do tratamento semântico, obtém-se a seguinte regra de regressão:

```

IF  $x_1 \leq 0,08$ 
  AND  $x_1 > 0,03$ 
  AND  $x_2 > 3,84$ 
THEN  $y = 5,02 + 0,4x_1 + 1,19x_3$ 

```

Com os atributos nominais, o problema ocorre quando existe mais de uma condição sobre um mesmo atributo nominal em uma determinada regra. A solução aqui consiste em encontrar a intersecção dos valores nominais entre todas as condições que aparecem naquela regra para o atributo nominal em questão.

Seja um conjunto de dados sobre automóveis com x_1 e x_2 atributos numéricos, e x_3 um atributo nominal assumindo os valores *gm*, *ford*, *fiat*, *wolksvagem*, *toyota*, *peugeot*, *audi* e *renaux*. Agora considere a regra de regressão apresentada a seguir:

```

IF  $x_1 < 8,75$ 
  AND  $x_2 \text{ IN } \{gm, ford, fiat, wolksvagem, toyota, peugeot, renaux\}$ 
  AND  $x_3 \text{ IN } \{toyota, peugeot, renaux\}$ 
  AND  $x_3 \text{ IN } \{peugeot, renaux\}$ 
THEN  $y = 11,2 - 1,57x_1 + 0,44x_2$ 

```

Analizando a regra, nota-se que para um exemplo de teste ser coberto por essa regra, o atributo nominal x_3 deve assumir os valores *peugeot* ou *renaux*. Assim sendo, aplicando o tratamento semântico, obtém-se a regra abaixo. Nota-se claramente a maior comprehensibilidade da regra apresentada a seguir comparando-se com a regra antes da aplicação do tratamento semântico.

```

IF  $x_1 < 8,75$ 
  AND  $x_3 \text{ IN } \{peugeot, renaux\}$ 
THEN  $y = 11,2 - 1,57x_1 + 0,44x_2$ 

```

Na Figura 4.20 são mostradas as regras de regressão com tratamento semântico geradas pelo algoritmo *M5* e mostradas nas Figuras 4.17 e 4.18. Observe que as regras [R0005], [R0006], [R0009] e [R0010] dessa figura são mais compreensíveis que as correspondentes na Figura 4.18 em função do tratamento semântico realizado.

```

Semantic Treatment of Regression Rules Copyright (c) Daniel Gomes Desualdo
Inducer: m5 Input File: housing.rules
Date: Mon Mar 31 17:13:32 2003 Output File: housing.Rules

[R0001] IF DIS <= 1.98
AND RM <= 6.68
AND LSTAT <= 14.6
THEN MedHouseVal = 75.7 - 0.129CRIM + 0.00632ZN + 0.441CHAS - 2.99NOX
+ 0.184RM - 22.4DIS + 0.0711RAD - 0.00231TAX - 0.118PTRATIO - 0.997LSTAT

[R0002] IF DIS <= 1.98
AND RM <= 6.68
AND LSTAT > 14.6
THEN MedHouseVal = 36.4 - 0.148CRIM + 0.00632ZN + 0.441CHAS - 2.99NOX
+ 0.184RM - 3.96DIS + 0.0711RAD - 0.00926TAX - 0.118PTRATIO - 0.354LSTAT
.

.

[R0005] IF INDUS <= 7.82
AND RM <= 7.12
AND RM > 6.68
AND LSTAT > 5.2
THEN MedHouseVal = 22.7 - 0.0204CRIM + 0.00546ZN - 0.286INDUS + 0.278CHAS
+ 5.39NOX + 2.89RM - 0.0289AGE - 0.552DIS + 0.0432RAD
- 0.00146TAX - 0.355PTRATIO - 0.61LSTAT

[R0006] IF INDUS <= 7.82
AND RM <= 7.44
AND RM > 7.12
THEN MedHouseVal = 46 - 0.0204CRIM + 0.00546ZN - 0.146INDUS + 0.278CHAS
+ 6.98NOX - 0.322RM - 0.0162AGE - 0.41DIS + 0.0432RAD
- 0.00146TAX - 0.355PTRATIO - 0.662LSTAT
.

.

[R0009] IF INDUS > 7.82
AND RM <= 7.44
AND RM > 6.68
AND NOX > 0.651
THEN MedHouseVal = 37.4 - 0.0204CRIM + 0.00546ZN + 0.278CHAS + 7.87NOX
+ 1.5RM - 0.271DIS + 0.0432RAD - 0.00146TAX - 0.514PTRATIO - 1.52LSTAT

[R0010] IF RM > 7.44
THEN MedHouseVal = 24.8 - 6.19CRIM + 0.00546ZN + 0.649INDUS + 0.278CHAS
- 1.76NOX + 5.82RM - 0.379DIS + 0.0432RAD
- 0.00146TAX - 1.5PTRATIO - 0.492LSTAT

```

Figura 4.20: Regras da Figura 4.18 na sintaxe padrão de regras de regressão com tratamento semântico.

O tratamento semântico é opcional e quando realizado é interessante que o seja antes da construção da matriz de contingência, de modo a tornar mais eficiente a construção da mesma. Depois de construída a matriz, é realizado o cálculo das medidas para avaliação de regras de regressão, sendo essas medidas utilizadas para a realização da integração e poda de regras. O módulo Combinação de Regressores, descrito a seguir, é executado independentemente dos demais.

4.3.2 Combinação de Regressores

A combinação de preditores é uma maneira de realizar o pós-processamento de conhecimento que pode ser realizado quando se tem como objetivo principal a melhora da precisão do preditor. Assim, o Ambiente *RÉPPE* possui o Módulo Combinação de Regressores que visa a combinação de regressores homogêneos (*bagging* e *boosting*) e heterogêneos (*stacking*) para problemas de regressão, a fim de melhorar a precisão.

A principal dificuldade, para problemas de regressão, refere-se ao *boosting*, uma vez que a cada nova iteração deve-se verificar quais exemplos foram cobertos corretamente pelo algoritmo, e regras de regressão não predizem um valor categórico definido e sim, um valor numérico, envolvendo um grau de incerteza quanto à classe predita.

Por ser o atributo meta contínuo, o cálculo da cobertura da regra, ou seja, a verificação da classe estar correta ou não em problemas de regressão é mais complicado do que em classificação. Para resolver esse problema foi utilizada uma estratégia baseada em um limiar, utilizando desvio-padrão, para verificar quais exemplos são cobertos corretamente pelo regressor (Pugliesi & Rezende, 2003). No Capítulo 6 é apresentada a combinação de regressores disponível no Ambiente *RÉPPE*. Também são apresentados os experimentos realizados para a validação desse módulo.

4.3.3 Integração de Regras de Regressão

Uma outra maneira de realizar o pós-processamento do conhecimento é por meio da integração do conhecimento. Integração do conhecimento refere-se ao processo de incorporar novo conhecimento em uma base de conhecimento já existente. Isso envolve determinar como diferentes conhecimentos interagem, e como o conhecimento de diferentes bases de conhecimento devem ser modificados para acomodar um ao outro. Integração de conhecimento envolve a construção de uma base de conhecimento a partir de vários sistemas de aprendizado e depois a integração desse conhecimento. O Módulo Integração de Regras apresentado na Figura 4.19 realiza a integração de regras no Ambiente *RÉPPE* tanto de regras geradas pelo mesmo algoritmo de aprendizado utilizando diferentes amostras, quanto de regras geradas por algoritmos distintos.

Após a integração de regras provenientes de diferentes amostras ou algoritmos, o volume de conhecimento produzido pelos algoritmos de aprendizado normalmente cresce, podendo inclusive sobrecarregar os usuários. Fornecer ao usuário uma grande quantidade de padrões descobertos a partir dos dados não é produtivo. Normalmente, ele procura por uma pequena lista de padrões interessantes (em sua opinião) e é essa lista que é importante ser gerada pelo processo de Mineração de Dados.

Encontrar quais padrões interessam a um usuário, visando tanto o desempenho quanto

a qualidade, é um problema difícil. Uma abordagem utilizada é eliminar um conjunto de regras que não são interessantes, enquanto limita a interação com o usuário a poucas e simples questões de predição. Essa eliminação de regras é realizada no Módulo Poda de Regras, no qual, de forma gradativa, são eliminadas 5%, 10%, 25% e 50% das regras originais.

Para realizar a integração de regras de regressão, descrita no Capítulo 7, utiliza-se o módulo de avaliação de regras.

4.4 Considerações Finais

A regressão é uma tarefa pouco explorada dentro da área de Aprendizado de Máquina e Mineração de Dados, visto que a maioria dos trabalhos trata de problemas de classificação. Da mesma maneira, no DISCOVER, o panorama não é muito diferente.

Nesse sentido, foi definida uma sintaxe padrão para representar regras de regressão no DISCOVER, e a implementação de uma classe para converter as saídas dos algoritmos de regressão Cubist, RT e Weka para essa sintaxe. O uso dessa sintaxe padrão é importante pois permite implementar diversas medidas de pós-processamento de uma única vez e utilizá-las para os tipos de saída gerada pelos diferentes algoritmos.

Para tanto, foi projetado e implementado o Ambiente *R&EPPE* com o objetivo de dar suporte ao pós-processamento de problemas de regressão. Além da sintaxe padrão desenvolvida para representar as regras de regressão no DISCOVER, no *R&EPPE* foram implementados seis módulos para trabalhar com problemas de regressão. São eles: Tratamento Semântico, Combinação de Regressores, Integração de Regras, Poda de Regras, Avaliação de Regras e Construção da Matriz de Contingência. Esses módulos estão detalhados nos Capítulos 5, 6 e 7.

Esses módulos objetivam incrementar a potencialidade do Ambiente DISCOVER para tratar problemas de regressão, uma vez que nesse ambiente a maioria das implementações estão voltadas para problemas de classificação.

Avaliação de Regras de Regressão

O processo de Mineração de Dados inicia-se com a escolha das fontes de dados a serem utilizadas e a definição dos objetivos. Uma parcela desses dados é selecionada, pré-processada e submetida aos métodos e ferramentas adequados com o objetivo de encontrar padrões e/ou modelos que representem o conhecimento obtido. Depois de extraídos, os padrões são pós-processados e o conhecimento adquirido é avaliado quanto à sua qualidade e/ou utilidade para determinar a viabilidade de sua utilização no apoio a algum processo de tomada de decisão.

Quando o conhecimento é obtido de uma base de dados, o usuário pode querer saber: se o conhecimento representa o que ele sabe; se não representa, qual parte do seu conhecimento anterior que está ou não correta ou é interessante; ou ainda, de que maneira esse novo conhecimento difere de seu conhecimento anterior (Liu & Hsu, 1996; Liu, Hsu, & Chen, 1997). Anteriormente, pesquisas assumiam que era de responsabilidade do usuário analisar o conhecimento. Entretanto, quando o número de regras é grande, é muito difícil para o usuário analisá-las manualmente.

A fim de verificar se o conhecimento possui as características desejadas, após adquirido, um pós-processamento deve ser realizado para avaliar o desempenho e a qualidade do mesmo. Essa avaliação procura investigar a precisão e a representação do modelo, a complexidade e a dificuldade de entendimento do conhecimento extraído. Além disso, o pós-processamento também tem como objetivo a filtragem do que foi aprendido, eliminando o conhecimento espúrio, sem valor ou de situações óbvias guiadas pelo senso comum.

Para as medidas de avaliação aqui relatadas, será adotada a representação mais genérica ($B \rightarrow H$), da regra *if < condição > then < conclusão >* apresentada no Capítulo 3, a fim de que se possa abranger uma maior variedade de regras.

Algumas medidas de avaliação podem ser calculadas utilizando os valores da matriz de contingência, que consiste de uma matriz 2×2 na qual são identificados o número de elementos preditos correta e incorretamente, que foi mostrada na Seção 4.1.1.

A avaliação de conhecimento leva em consideração algumas particularidades dos diferentes tipos de problema. Regras de regressão, apesar de serem semelhantes às regras de classificação, não predizem um valor categórico definido e sim, um valor numérico, envolvendo um grau de incerteza quanto à classe predita (Cheng, 1998). Assim, por ser o atributo meta contínuo, o cálculo da cobertura da regra, ou seja, a verificação da classe estar correta ou não em problemas de regressão se torna mais complicada do que em classificação (o cálculo dos valores relacionados à H e \bar{H} não são triviais). Essa verificação é imprescindível para o cálculo dos valores da matriz de contingência das regras. Com o cálculo da matriz de contingência para regras de regressão incrementa-se o número de medidas disponíveis para análise das regras, uma vez que várias medidas, tanto de precisão quanto de qualidade do conhecimento obtido, são calculadas a partir dos valores da matriz de contingência, como o *framework* descrito em (Lavrac, Flach, & Zupan, 1999).

As medidas de avaliação de regras de regressão que levam em consideração os valores da matriz de contingência para o seu cálculo são relatadas na Seção 5.1. Já na Seção 5.2 são apresentadas as medidas de avaliação que não utilizam os valores da matriz de contingência para o seu cálculo. Na Seção 5.3 são apresentadas as três estratégias propostas para o cálculo dos valores da matriz de contingência para problemas de regressão, sendo na Seção 5.4 mostrados os resultados dos experimentos realizados para a validação das três estratégias. Finalmente, a Seção 5.5 contém as considerações finais deste capítulo.

5.1 *Medidas Derivadas da Matriz de Contingência*

A partir do cálculo dos valores da matriz de contingência, várias medidas podem ser definidas, tais como as medidas de qualidade de regras apresentadas em (Lavrac, Flach, & Zupan, 1999; Horst, 1999; Piatetsky-Shapiro, 1991).

Precisão (*Acc*)

De forma geral, a precisão pode ser medida pelo total de exemplos preditos corretamente dividido pelo total de exemplos cobertos. Ou seja, a precisão de uma regra R é definida como a probabilidade condicional de H ser verdadeiro dado que B é

verdadeiro, como mostrada na seguinte equação:

$$Acc(R) = p(H|B) = \frac{p(BH)}{p(B)} = \frac{n(BH)}{n(B)} \quad (5.1)$$

Essa definição de precisão visa a avaliação de regras isoladas, sendo diferente da precisão de um conjunto de regras ou preditor, que é definido como:

$$Acc = p(HB) + p(\overline{BH}) = \frac{n(BH) + n(\overline{BH})}{n} \quad (5.2)$$

Erro Amostral ou Aparente (*Err*)

O erro amostral refere-se à taxa de erro da hipótese sobre a amostra de dados disponível. Como pode ser visualizado na equação dada a seguir, quanto maior o erro, menor a precisão da regra.

$$Err = 1 - Acc \quad (5.3)$$

Confiança Negativa (*NegRel*)

Essa medida é definida como sendo a probabilidade condicional de H ser falso dado que B é falso. A confiança negativa representa a confiança de que, se o exemplo não é coberto pela regra, então a conclusão é falsa. É dada por:

$$NegRel(R) = p(\overline{H}|\overline{B}) = \frac{p(\overline{BH})}{p(\overline{B})} = \frac{n(\overline{BH})}{n(\overline{B})} \quad (5.4)$$

Sensitividade (*Sens*)

A sensitividade é definida como a probabilidade condicional de B ser verdadeiro dado que H é verdadeiro. É representada por:

$$Sens(R) = p(B|H) = \frac{p(BH)}{p(H)} = \frac{n(BH)}{n(H)} \quad (5.5)$$

Especificidade (*Spec*)

Essa medida é definida como sendo a probabilidade condicional de B ser falso dado que H também é falso. É definida por:

$$Spec(R) = p(\overline{B}|\overline{H}) = \frac{p(\overline{BH})}{p(\overline{H})} = \frac{n(\overline{BH})}{n(\overline{H})} \quad (5.6)$$

Cobertura (*Cov*)

A cobertura de uma regra representa a fração de todos os exemplos que são cobertos pela regra, podendo ser considerada como uma medida de generalidade da regra. Quanto maior o valor dessa medida, maior o número de exemplos cobertos pela regra. É dada por:

$$Cov(R) = p(B) = \frac{n(B)}{n} \quad (5.7)$$

Suporte (*Sup*)

O suporte de uma regra indica a fração de todos os exemplos que são cobertos corretamente pela regra. É representada por:

$$Sup(R) = p(BH) = \frac{n(BH)}{n} \quad (5.8)$$

Novidade (*Nov*)

Essa medida tem como objetivo identificar o quanto uma regra é interessante, inovadora ou não usual. Ela é definida pela comparação entre o valor esperado da medida suporte ($p(HB)$) e os valores de $p(B)$ e $p(H)$. Quanto maior a diferença entre o valor esperado e o valor observado, maior a probabilidade de existir uma correlação inesperada e verdadeira entre B e H . É dada por:

$$Nov(R) = p(BH) - p(H) \times p(B) = \frac{n(BH)}{n} - \left(\frac{n(H)}{n} \times \frac{n(B)}{n} \right) \quad (5.9)$$

Satisfação (*Sat*)

A satisfação da regra representa a queda relativa na precisão entre a regra $B \rightarrow \text{verdadeiro}$ e a regra $B \rightarrow H$. Pode ser observado que a satisfação de uma regra está relacionada com a precisão, pois $Sat(B \rightarrow H) = 1$ se e somente se $Acc(B \rightarrow H) = 1$. Essa medida está mais relacionada com a descoberta de conhecimento novo. Ela é definida por:

$$Sat(R) = \frac{p(\bar{H}) - p(\bar{H}|B)}{p(\bar{H})} - \frac{\frac{n(\bar{H})}{n} - \frac{n(B\bar{H})}{n}}{\frac{n(\bar{H})}{n}} = 1 - \frac{n(B\bar{H})}{n(B)} + \frac{n}{n(H)} \quad (5.10)$$

Consistência

Consistência é a medida de quanto um conhecimento é específico para o problema. Quanto mais alta a consistência, mais precisamente é coberta a classe em questão. A consistência é máxima quando a regra cobre somente os exemplos da classe e

nenhum exemplo fora desta classe, isto é, quando não existem falsos positivos. A consistência de uma regra R é definida por:

$$Consist\acute{e}ncia(R) = \frac{n(BH)}{n(BH) + n(\overline{BH})} \quad (5.11)$$

Deve ser observado que esta medida não reflete o número de exemplos cobertos pela regra.

Completude

Completude é a medida que calcula o quanto do domínio do problema é coberto pela regra. Quanto mais alta a completude, mais exemplos são cobertos pela regra. A completude de uma regra R é definida por:

$$Completude(R) = \frac{n(BH)}{n(BH) + n(\overline{BH})} \quad (5.12)$$

Estatística de Cohen

A estatística de Cohen busca encontrar um nível de associação na diagonal principal da matriz de contingência, para servir como medida de qualidade de uma regra. O cálculo é feito por meio da combinação dos valores de consistência e completude de uma regra com o número total de exemplos e o número de exemplos de uma classe cobertos pela regra R , por meio da seguinte equação:

$$\psi_{Cohen}(R) = \frac{n \cdot Consist\acute{e}ncia(R) \cdot Completude(R) - n(BH)}{n \cdot \frac{Consist\acute{e}ncia(R) + Completude(R)}{2} - n(BH)} \quad (5.13)$$

O valor retornado por essa equação está entre -1 e 1. Valores negativos representam um relacionamento inverso entre uma regra e a classe por ela predita.

Estatística IMAFO

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

$$Q_{IMAFO}(R) = (Acc \cdot E_R) \cdot 10 \quad (5.14)$$

sendo Acc a precisão da regra R , e E_R a estimativa de cobertura da regra R calculada pela seguinte equação:

$$E_R = \exp \left(\frac{n(\overline{BH})}{n(H)} - 1 \right) \quad (5.15)$$

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

Estatística de Coleman

A estatística de Coleman é uma medida da “combinação” entre a primeira coluna e cada linha da tabela de contingência. No caso de uma tabela de contingência, como a Tabela 4.2, a relação correta é entre a primeira coluna e a primeira linha. Esta medida é calculada de forma similar a estatística de Cohen por meio da seguinte equação:

$$Q_{Coleman}(R) = \frac{n \cdot Consist\acute{e}ncia(R) \cdot Completude(R) - n(BH)}{n \cdot Completude(R) - n(BH)} \quad (5.16)$$

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

SKIB1 e SKIB2

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

$$Q_{SKIB1}(R) = Q_{Coleman}(R) \cdot \frac{2 + Q_{Cohen}(R)}{3} \quad (5.17)$$

$$Q_{SKIB2}(R) = Q_{Coleman}(R) \cdot \frac{1 + Completude(R)}{2} \quad (5.18)$$

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

Uso da Informação

A teoria da informação é também uma área que está estritamente relacionada a estatística, e pode oferecer alguma ajuda para medições de qualidade. O cálculo do

ganho de informação resultante de uma regra particular pode ser dado pela seguinte equação:

$$Q_{IKIB}(R) = -\log \frac{n(H)}{n} + \log \text{Consistência}(R) \quad (5.19)$$

sendo todos os *logs* na base 2. Porém, somente é válida se $\text{Consistência}(R) \geq \frac{n(H)}{n}$.

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

5.2 Medidas Não Derivadas da Matriz de Contingência

Nesta seção são apresentadas algumas medidas para problemas de regressão que não utilizam os valores da matriz de contingência para seu cálculo.

Normalmente, em regressão utilizam-se três medidas para determinar a precisão dos modelos, que estão ligadas à magnitude e a forma de verificação de erros (Torgo, 1995; Merz, 1998; Torgo, 1999):

MAD (*Mean Absolute Deviation* ou Média da Diferença Absoluta)

MAD é uma medida de erro que quantifica o erro do modelo pela média dos desvios absolutos de suas previsões, isto é, MAD consiste na média da diferença (em módulo) entre os valores reais e preditos para um atributo meta, como mostra a equação a seguir:

$$\text{MAD}(R) = \frac{1}{n} \sum_{i=1}^n |y_i - h(x_i)| \quad (5.20)$$

MSE (*Mean Squared Error* ou Média dos Erros ao Quadrado)

A MSE (Equação 5.21) consiste na média do quadrado da diferença entre os valores reais e preditos para um atributo meta. Essa medida, muitas vezes, é utilizada para minimizar o erro quanto à previsão dos valores.

$$\text{MSE}(R) = \frac{1}{n} \sum_{i=1}^n (y_i - h(x_i))^2 \quad (5.21)$$

RMSE (*Relative Mean Squared Error* ou MSE Relativa)

A medida RMSE fornece um valor relativo para o erro. Um valor entre 0 e 1 indica que a regra R está se saindo melhor que apenas predizer o valor médio do atributo

meta Y .

$$\text{RMSE}(R) = \left(\frac{1}{n} \sum_{i=1}^n (y_i - h(x_i))^2 \right) / \left(\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 \right) = \frac{\text{MSE}(R)}{\text{MSE}(\bar{y})} \quad (5.22)$$

Além das medidas para determinar a precisão dos modelos é importante calcular a variância e o desvio padrão desses erros.

Os resultados, tanto do erro quanto da variância, também dependem do fato de como o experimento foi realizado, ou seja, qual paradigma para estimativa do erro foi utilizada, como, *bootstrap*, *holdout*, *k-fold cross-validation*, *k-fold stratified cross-validation*, e *leave-one-out*. (Mitchell, 1997).

Variância

A variância mede quanto as previsões do algoritmo de aprendizado variam com respeito a outras, isto é, como é a flutuação para conjuntos diferentes de treinamento. Vale ressaltar que o desvio padrão é a raiz quadrada da variância.

$$\text{var}(R) = \frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n - 1} \quad (5.23)$$

Algumas medidas de precisão e cobertura também podem ser utilizadas como parâmetros para a avaliação da qualidade do conhecimento extraído. Uma medida de qualidade permite avaliar e, consequentemente, aprovar, aceitar ou recusar o conhecimento extraído.

Essas abordagens consideraram alguns fatores como (Liu & Hsu, 1996): (i) a possível existência de um grande número de regras dificultando a análise por parte do usuário; (ii) a possível complexidade ao avaliar a regra, por exemplo, o número de condições ou a estrutura de representação.

O primeiro fator deve-se ao tamanho do conjunto de dados analisado, pois dependendo das configurações dos algoritmos podem ser retiradas regras muito precisas que cobrem poucos exemplos, ou regras imprecisas que cobrem muitos exemplos. Já o segundo fator deve-se a características da compreensão dos usuários.

Também deve ser considerado que se é um ser humano quem deve aplicar o procedimento de predição, este deve ser facilmente compreensível ou senão enganos podem ocorrer na utilização do conhecimento. Um exemplo é o caso Three-Mile Island, em que os dispositivos automáticos corretamente recomendaram o desligamento, mas esta recomendação não foi executada pelos operadores humanos que a consideraram infundada. Uma história similar se aplica ao desastre de Chernobyl.

O critério de comprehensibilidade também está ligado a conceitos subjetivos relacionados, principalmente, a facilidade de compreensão da regra e a capacidade exploratória dos padrões extraídos no processo. A comprehensibilidade tratada como a complexidade das regras leva em consideração tanto o número de atributos na parte condicional da regra quanto na parte conclusiva.

Medir o grau de comprehensibilidade dos padrões encontrados é uma tarefa dependente da capacidade de abstração do usuário. Dependendo do modelo obtido é necessário um critério diferenciado para avaliar a comprehensibilidade.

Para árvores de regressão a comprehensibilidade pode ser medida pelo número de nós da árvore. Árvores com muitos nós aumentam a complexidade de abstração (Ram, 1990). Além de verificar o número de nós outro fato que deve ser levado em consideração é a profundidade e a largura da árvore, sendo que quanto mais profunda for a árvore menor será sua capacidade de explicação (Weiss & Indurkha, 1998). Já quanto a largura de uma árvore, geralmente, quanto mais larga, mais difícil é de ser compreendida devido ao maior número de possíveis soluções.

Já para regras de regressão um fator utilizado para verificar a comprehensibilidade é o número de condições da regra, visto que um grande número de condições pode dificultar o entendimento da regra.

Um outro fator que influencia em ambas as técnicas é o número de termos existentes na função (no caso das técnicas que predizem o atributo-meta utilizando-se uma função matemática), uma vez que a existência de muitos termos pode dificultar a compreensão dessa função.

Vale ressaltar que medidas de comprehensibilidade baseadas em número de nós, condições e termos em uma árvore, regra ou fórmula levam em conta apenas critérios sintáticos. Intuitivamente, uma medida de comprehensibilidade deveria levar em conta também critérios semânticos ou aspectos da ciência cognitiva (Pazzani, 2000b).

Às vezes, quando árvores de regressão são convertidas para regras de regressão, as regras que constituem podem possuir condições desnecessárias. Isso acontece porque em muitos casos um mesmo atributo é utilizado para particionar o conjunto de dados em diferentes níveis de uma árvore assumindo diferentes valores, o que faz com que uma condição sobreponha a outra, tanto no caso de atributos numéricos quanto nominais. Nesse sentido, pode ser aplicado um tratamento semântico às regras com o objetivo de excluir as condições desnecessárias de cada regra. Esse tratamento, que no Ambiente *REPPE* é implementado no módulo de tratamento semântico mostrado na Figura 4.19, apresenta duas vantagens:

1. melhora a comprehensibilidade da regra, visto que quanto maior o número de condições, mais difícil o entendimento da regra;

2. aumenta a eficiência dos programas que utilizam essas regras como entrada, já que muitos testes desnecessários são evitados. Isso se apresenta extremamente importante quando o arquivo de regras é extenso e/ou o número de exemplos de teste é grande.

Além da precisão, cobertura e comprehensibilidade, um outro problema no campo de descoberta de conhecimento é o desenvolvimento de medidas de grau de interesse dos padrões encontrados. O grau de interesse é uma maneira de avaliar qualidade tentando capturar o quanto de conhecimento interessante (ou inesperado) elas apresentam. As medidas de grau de interesse estão baseadas em vários aspectos, principalmente na utilidade que as regras representam para o usuário (Dong & Li, 1998).

Considerando as abordagens e os fatores de interesse, é comum encontrar na literatura diversas referências a medidas de grau de interesse para problemas de classificação, algumas delas em (Freitas, 1998b,a; Horst, 1999; Sahar, 1999; Sahar & Mansour, 1999). Entretanto, medidas de grau de interesse para problemas de regressão ainda estão em fase inicial de estudos, devido principalmente à incerteza de predizer o atributo-meta que se trata de um valor numérico. Desse modo, interessa ao usuário identificar os padrões que tenham erros mínimos e que consigam generalizar um grande número de exemplos. Piatetsky-Shapiro foi um dos primeiros pesquisadores a discutir a problemática do grau de interesse do conhecimento obtido em (Piatetsky-Shapiro, 1991). Desde então, surgiram diversas propostas de medidas de grau de interesse.

Tais medidas de grau de interesse estão divididas em medidas objetivas, que dependem apenas da estrutura dos padrões e dos dados utilizados no processo de descoberta, e as medidas subjetivas, que também dependem da classe de usuários que examinam os padrões.

Em (Torgo, 1995) são apresentadas medidas para avaliar a qualidade de regras especializadas. Uma regra especializada R' é construída variando-se operadores ou atributos de uma regra original R com a intenção de se obter uma melhor precisão. A seguir são descritas algumas medidas de grau de interesse objetiva. As medidas de avaliação de qualidade, GanhoMAD, LC e Q, quantificam algumas características de regras de regressão, como precisão e representação da classe no formato de função matemática.

GanhoMAD

A medida GanhoMAD quantifica o ganho entre duas regras de regressão diferenciadas pela medida MAD. A medida GanhoMAD é definida pela seguinte equação:

$$\text{GanhoMAD}(R) = \begin{cases} 1 - \frac{\text{MAD}(R)}{\text{MAD}(R) - \text{MAD}(R')} & \text{se } R \text{ for a regra original} \\ \frac{\text{MAD}(R) - \text{MAD}(R')}{\text{MAD}(R)} & \text{se } R \text{ for uma regra especializada} \end{cases} \quad (5.24)$$

na qual, $MAD(R)$ é o valor MAD da regra original R , enquanto que $MAD(R')$ é o valor da MAD da regra especializada R' .

Para a utilização dessa medida é necessário que a MAD esteja normalizada. Essa normalização é necessária para evitar a ocorrência de um ganho negativo, isto é, $MAD_n = MAD/MAD_{maior}$, na qual MAD_{maior} é a maior diferença entre um valor predito e real para a regra R .

LC (*Lost Coverage*)

A medida de grau de interesse denominada LC quantifica a perda de cobertura dos exemplos entre duas regras R e R' , sendo definida pela Equação 5.25. O valor de LC para a regra original é o próprio número de exemplos.

$$LC(R') = \frac{(n(BH) + n(\bar{BH})) - (n(BH)' + n(\bar{BH})')}{n(BH) + n(\bar{BH})} \quad (5.25)$$

Q

Com o auxílio das medidas GanhoMAD e LC, uma medida de qualidade Q da regra R pode ser calculada a partir de constantes (pesos) atribuídas a GanhoMAD e a LC, conforme a equação:

$$Q(R) = GanhoMAD(R) * w_{ganho} + (1 - LC(R)) * (1 - w_{ganho}) \quad (5.26)$$

na qual, os pesos w_{ganho} e $(1 - w_{ganho})$ representam um balanceamento entre a generalidade e a corretude da regra.

Se o valor de w_{ganho} for alto, de acordo com a medida Q , as regras mais específicas terão um maior grau de interesse. Por outro lado, se o valor de w_{ganho} for baixo, as regras mais gerais serão as escolhidas, diminuindo-se a precisão (Torgo, 1995). A constante w_{ganho} pode ser especificada pelo usuário determinando um intervalo¹ $[w_1 \dots w_2]$ que consiste em um valor mínimo e máximo para minimizar ou maximizar o ganho, sendo que esses valores devem pertencer a um intervalo entre 0 e 1. O cálculo de w_{ganho} é mostrado na equação:

$$w_{ganho} = w_1 + (w_2 - w_1) * GanhoMAD(R) \quad (5.27)$$

Gabaritos (*Templates*) de Regras

Para a utilização desta medida são definidos gabaritos com o especialista/usuário e o analista e interessam somente as regras que “casam” com esses gabaritos (*Templates*) (Fu & Han, 1995; Klemettinen, Mannila, Ronkainen, Toivonen, & Verkamo, 1994).

¹O usuário também pode especificar um valor fixo w_{ganho} ao invés de um intervalo.

Cobertura de Regras Mínimas

Utilizando a cobertura de regras mínimas, algumas regras, ou seja, um conjunto mínimo de regras que cobrem o maior número de exemplos do conjunto de dados, são apresentadas ao usuário (Toivonen, Klemettinen, Ronkainen, Hätkönen, & Mannila, 1995).

Entre as medidas de grau de interesse subjetiva tem-se:

Inesperabilidade

A inesperabilidade avalia se as regras são excepcionais (não observadas) frente ao conhecimento do especialista do domínio (Silberschatz & Tuzhilin, 1996, 1995; Padmanabhan & Tuzhilin, 1999). A inesperabilidade pode ser interpretada no sentido estatístico, como uma alta causalidade que aparece sob uma suposição ou hipótese independente ou sob algum *bias* que seja contrário às crenças do usuário (Liu & Hsu, 1996; Liu, Jsu, Ma, & Chen, 1999).

Acionabilidade

A açãoabilidade avalia se o usuário pode utilizar as regras para obter vantagem na aplicação da regra, ou seja, as regras são consideradas interessantes, desde que o usuário possa obter vantagens ao utilizá-las (Piatetsky-Shapiro & Matheus, 1994).

5.3 *Estratégias para Determinar Matriz de Contingência para Regras de Regressão*

Visando o cálculo dos valores da matriz de contingência para regras de regressão, e consequentemente a possibilidade de utilização de todas as medidas apresentadas na Seção 5.1, foram propostas a utilização de três diferentes estratégias para calcular essa matriz. As estratégias são: (i) ajustamento, que visa encontrar um limiar entre o valor real e o valor predito, sendo essa estratégia por nós definida e implementada; (ii) pseudo-classe, que divide o intervalo de variação do atributo meta em pseudo-classes e verifica se o valor predito e o real estão ou não na mesma pseudo-classe; e (iii) discretização baseada em entropia, que também constrói intervalos para o atributo meta e verifica se o valor predito e o real se encontram no mesmo intervalo. Essas três estratégias serão detalhadas a seguir.

5.3.1 Ajustamento

Como o algoritmo de regressão retorna como classe ou cabeça (*Head*) da regra um valor numérico real ou uma equação matemática baseada em seus atributos, que gera como resultado final para cada exemplo um valor real, utilizou-se um limiar para a construção da matriz de contingência. Esse limiar (ϵ) foi apresentado na Equação 3.2 (página 25). Com a utilização do limiar ϵ é possível calcular o número de exemplos para o qual a cabeça da regra é verdadeira ou falsa, ou seja, se o valor referente a classe do exemplo pertence ao intervalo definido por esse limiar.

O ponto crítico nesta estratégia é a definição do valor do limiar. O limiar utilizado na estratégia, por nós definida e implementada, utiliza o cálculo do desvio padrão do erro da regra encontrado no conjunto de dados.

Com o objetivo de analisar como diferentes limiares, utilizando desvio padrão, influenciam nos valores da matriz de contingência três variações, por nós escolhidas, foram consideradas: somente o valor do desvio padrão, o desvio padrão mais a média do erro e o desvio padrão mais a mediana do erro. Com isso é possível analisar o comportamento de diferentes tamanhos de limiar, dado que utilizando somente o desvio padrão tem-se um limiar mais restrito.

Para cada regra são calculados o desvio padrão, a média e a mediana, que leva em consideração tanto os exemplos cobertos por essa regra, quanto os não cobertos. Ou seja, para cada regra tem-se dois desvios padrão, duas médias e duas medianas, que são utilizadas de acordo com a cobertura do exemplo.

Como exemplo, considere a seguinte regra, na sintaxe padrão (Pugliesi, Dosualdo, & Rezende, 2003), na qual *rings* é o atributo a ser predito:

```
[R0001] IF shell_weight <= 0.25
        THEN rings = 5 +21.4 shell_weight +1.7 diameter
```

Suponha que o conjunto de exemplos possui 12 exemplos cobertos por essa regra e 12 exemplos não cobertos por essa regra, como são apresentados nas Tabelas 5.1 e 5.2, respectivamente.

Para cada um dos exemplos da Tabela 5.1, foi calculado o valor predito pela regra, bem como o erro encontrado, ou seja, a diferença absoluta entre o valor predito e o real. Por fim, foram calculadas as medidas relacionadas a regra, sendo o desvio padrão igual a 1,86; a média igual a 1,36 e a mediana igual a 0,75.

Além disso, para cada um dos exemplos não cobertos pela regra, apresentados na

Tabela 5.1: Exemplos cobertos pela regra.

Atributos									Valor Preditivo	Erro
sex	length	diameter	height	whole weight	shucked weight	viscera weight	shell weight	rings		
M	0,46	0,37	0,10	0,51	0,22	0,10	0,15	15	8,83	6,17
M	0,35	0,27	0,09	0,23	0,10	0,05	0,07	7	6,95	0,05
F	0,53	0,42	0,14	0,68	0,26	0,14	0,21	9	10,20	1,20
M	0,44	0,37	0,13	0,52	0,22	0,11	0,16	10	8,94	1,06
I	0,33	0,26	0,08	0,21	0,09	0,04	0,06	7	6,61	0,39
I	0,43	0,30	0,10	0,35	0,14	0,08	0,12	8	8,08	0,08
M	0,48	0,37	0,13	0,51	0,22	0,11	0,17	9	9,16	0,16
F	0,53	0,38	0,14	0,61	0,19	0,15	0,21	14	10,10	3,90
M	0,43	0,35	0,11	0,41	0,17	0,08	0,14	10	8,48	1,52
M	0,49	0,38	0,14	0,54	0,22	0,10	0,19	11	9,71	1,29
F	0,54	0,41	0,15	0,68	0,27	0,17	0,21	10	10,10	0,10
F	0,47	0,36	0,10	0,48	0,17	0,08	0,19	10	9,56	0,44

Tabela 5.2: Exemplos não cobertos pela regra.

Atributos									Valor Preditivo	Erro
sex	length	diameter	height	whole weight	shucked weight	viscera weight	shell weight	rings		
F	0,53	0,42	0,15	0,78	0,24	0,14	0,33	20	12,80	7,20
F	0,55	0,43	0,13	0,77	0,29	0,15	0,26	16	11,30	4,70
F	0,55	0,44	0,15	0,89	0,31	0,15	0,32	19	12,60	6,40
F	0,57	0,44	0,16	0,94	0,43	0,21	0,27	12	11,50	0,50
F	0,62	0,48	0,17	1,16	0,51	0,30	0,31	10	12,30	2,30
F	0,56	0,44	0,14	0,93	0,38	0,19	0,30	11	12,20	1,20
F	0,58	0,45	0,19	1,00	0,39	0,27	0,29	11	11,90	0,90
M	0,59	0,45	0,14	0,93	0,36	0,23	0,28	12	11,70	0,30
M	0,61	0,48	0,18	0,94	0,39	0,22	0,30	15	12,10	2,90
M	0,58	0,47	0,17	1,00	0,39	0,24	0,33	10	12,90	2,90
F	0,68	0,56	0,17	1,64	0,61	0,28	0,46	15	15,80	0,80
M	0,67	0,53	0,17	1,34	0,55	0,36	0,35	18	13,40	4,60

Tabela 5.2, também foi calculado o valor preditivo, bem como o erro encontrado. E foram calculadas as medidas relacionadas a essa regra, sendo o desvio padrão igual a 2,36; a média igual a 2,89 e a mediana igual a 2,60.

Como esperado, os valores relacionados aos exemplos não cobertos pela regra apresentaram maiores valores de erro.

Considerando os cálculos apresentados, os limiares e para essa regra podem ser visualizados na Tabela 5.3. Assim, o exemplo é considerado corretamente predito pela regra caso o valor preditivo esteja entre o valor real \pm o limiar utilizado.

Tabela 5.3: Limiares ϵ para a Regra [R0001].

	Desvio Padrão	Desvio Padrão + Média	Desvio Padrão + Mediana
B (shell_weight $\leq 0,25$)	1,86	3,22	2,61
B (shell_weight $> 0,25$)	2,36	5,25	4,96

5.3.2 Pseudo-Classe

Um conjunto de “pseudo-classes” é gerado utilizando o algoritmo P-Class em um processo de discretização descrito em (Weiss & Indurkhy, 1995). O objetivo desse algoritmo é associar os valores do atributo-meta Y às pseudo-classes de modo a minimizar a distância entre cada y_i e a média de sua pseudo-classe.

O P-Class descreve como associar um conjunto de valores $\{y_i\}$ a k classes da seguinte maneira:

- ordena os valores de Y ;
- associa aproximadamente um número igual de valores de $\{y_i\}$ vizinhos para cada pseudo-classe;
- move um valor y_i para uma pseudo-classe vizinha quando isso reduz a distância de y_i para a média de sua pseudo-classe;
- concatena pseudo-classes com médias iguais.

Fornecido o valor de k , o algoritmo P-Class associa de maneira relativamente rápida os Y valores de modo que as distâncias totais sejam minimizadas. No entanto, uma questão-chave é como determinar k , ou seja, o número de pseudo-classes a serem geradas. Em (Weiss & Indurkhy, 1995) são realizados alguns experimentos para determinar o melhor número de pseudo-classes. Na Figura 5.1 é apresentado um gráfico do erro relativo *versus* o número de pseudo-classes. Como pode ser visualizado nesse gráfico, à medida que o número de pseudo-classes aumenta, os resultados melhoram, até alcançar o valor 6, sendo que logo em seguida os resultados começam a piorar.

Esse algoritmo P-Class foi desenvolvido com o intuito de transformar um problema de regressão em um problema de classificação. Para isso, os valores numéricos do atributo-meta passam por um processo de discretização, em que são geradas as pseudo-classes. Realizada a discretização dos valores (geradas as pseudo-classes), um algoritmo de indução de regras de decisão como o C4.5 (Quinlan, 1993) ou o CN2 (Clark & Niblett, 1989), pode

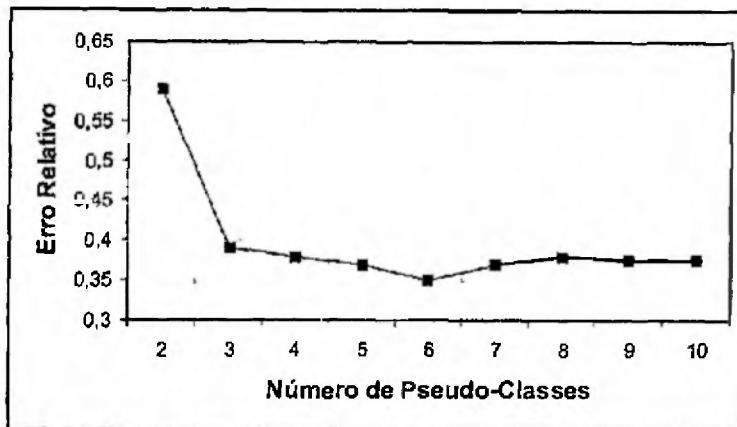


Figura 5.1: Desempenho de acordo com o número de pseudo-classes.

ser utilizado para gerar as regras. Com as regras induzidas, no processo proposto em (Weiss & Indurkhya, 1995) são aplicados então procedimentos de poda e otimização para produzir um conjunto otimizado de regras.

Na estratégia de pseudo-classes, a especificação das classes não utiliza informação alguma além da ordem de Y . Nenhuma suposição sobre a verdadeira natureza da função base é feita. O objetivo, com essa estratégia, é fazer com que a distância entre cada valor de Y e a média de sua pseudo-classe seja minimizada. Essa abordagem é relatada em (Weiss & Indurkhya, 1995), que a utiliza para transformar o problema de regressão em classificação. Porém, neste artigo utiliza-se essa abordagem com o propósito de calcular os valores da matriz de contingência, verificando se o valor predito e o real pertencem ou não à mesma pseudo-classe, sendo a implementação da mesma realizada por Dosualdo (2003).

Para a realização dos experimentos aqui apresentados utilizou-se 3 e 6 pseudo-classes. O número 6 foi escolhido, pois (Weiss & Indurkhya, 1995) mostra que com esse número de classes foram encontrados os melhores resultados nos testes empíricos realizados. Já o número 3 foi escolhido devido ao fato de estarmos utilizando, inicialmente, conjuntos de dados não muito grandes para a realização dos experimentos.

5.3.3 Discretização Baseada em Entropia

A estratégia de discretização baseada em entropia também utiliza apenas informações da ordem de Y . A discretização considera apenas o conjunto de dados original para construir os intervalos.

Essa estratégia requer que seja selecionado um atributo discreto. Os limites de cada intervalo são escolhidos de modo que a distribuição de atributos com faixas diferentes sejam a mais diferente possível. Essa abordagem continua criando limites que separem as

faixas até que nenhum intervalo adicional seja considerado significante.

A ferramenta utilizada para verificar essa estratégia foi a versão 2.6 do MineSetTM da Silicon Graphics (Rathjens, 1996). Essa ferramenta incorpora técnicas de Mineração de Dados que auxiliam o analista/especialista na exploração dos dados, e possibilita também discretização baseada em peso e escala uniforme. Para a discretização realizada com essa ferramenta foram utilizados os mesmos números de intervalos da estratégia anterior, ou seja, 3 e 6 intervalos. Além disso, também foram feitos experimentos com o MineSetTM gerando automaticamente a quantidade e os limites dos intervalos.

Na próxima seção serão mostrados os testes realizados para verificar a eficiência e eficácia dessas estratégias, bem como uma comparação entre elas e a medida MAD que não leva em consideração os valores da matriz de contingência.

5.4 Avaliação Experimental

Para avaliar os resultados das três estratégias propostas para calcular a matriz de contingência foram utilizadas 2 medidas. Uma dessas medidas é a precisão, derivada da matriz de contingência (Equação 5.1). Essa medida é utilizada para validar os resultados das três estratégias usadas para calcular a matriz de contingência.

Já a outra medida, que não considera a matriz de contingência, é a MAD, que quantifica o erro do modelo pela média dos desvios absolutos de suas previsões, isto é, a MAD consiste na média da diferença (em módulo) entre os valores reais e preditos para um atributo-meta, como é mostrado na Equação 5.20.

Para validar as três estratégias utilizando experimentação empírica, uma questão principal precisa ser respondida:

Quais erros baseados na matriz de contingência, utilizando as diferentes estratégias, possuem valores semelhantes aos encontrados utilizando a medida MAD?

Para a realização dos experimentos foi utilizada o algoritmo de regressão Cubist (Rulequest-Research, 2001) para encontrar os padrões. Além disso, se utilizou as bases de dados Abalone, Auto-MPG e Housing. Por terem sido retiradas do Repositório da UCI, não foi necessário realizar limpeza nos dados antes de iniciar o processo de extração de padrões.

5.4.1 Descrição dos Experimentos

Os experimentos foram realizados utilizando *10-fold cross-validation* e os resultados de todas as regras geradas foram coletados. Para avaliar as regras foi calculada a precisão a partir da matriz de contingência utilizando as três estratégias propostas, além das variações dentro de cada estratégia, ou seja, a estratégia ajustamento (somente o valor do desvio padrão, o desvio padrão mais a média do erro e o desvio padrão mais a mediana do erro); a pseudo-classe (3 e 6 pseudo-classes) e a discretização baseada em entropia (3 e 6 intervalos e automático). Para todas as regras também foi calculada a MAD como parâmetro chave de comparação.

Para que a comparação entre as medidas pudesse ser realizada, normalizou-se a medida MAD para que seus valores ficassem somente entre 0 e 1, como todas as medidas baseadas na matriz de contingência. Houve a necessidade também de padronizar as medidas de maneira que todas elas retratassem o erro cometido pela regra, uma vez que algumas medidas fornecem como resultado o valor da precisão e não do erro. Vale ressaltar que todas as medidas baseadas na matriz de contingência são comparadas com a MAD, para que se possa verificar quais medidas possuem resultados semelhantes a MAD, de modo a verificar qual das estratégias gerou bons resultados para o cálculo da matriz.

Para uma avaliação comparativa confiável, utilizou-se em todos os experimentos o Teste *t-Student* com intervalo de confiança de 95% para a análise estatística dos resultados. O Teste *t-Student* é um teste estatístico cujo objetivo é testar a igualdade entre duas médias. O teste supõe independência e normalidade das observações. As variâncias dos dois grupos podem ser iguais ou diferentes, havendo alternativas de teste para as duas situações.

As análises estatísticas foram feitas a fim de observar a não existência de uma diferença significativa das medidas encontradas para as regras de regressão. Quando um resultado é estatisticamente significante quer dizer que as diferenças encontradas são grandes o suficiente para não serem atribuídas ao acaso.

Para essa verificação foram realizados testes de hipóteses, por meio do qual expressasse determinado parâmetro da população e procura-se evidência para rejeitar ou não a hipótese nula (a da não diferença entre duas variáveis). Para tanto, foram utilizadas as seguintes hipóteses:

H_0 : Não existe diferença significativa entre as medidas de erro das regras.

H_1 . Existe diferença significativa entre as medidas de erro das regras.

A hipótese nula (H_0) é colocada à prova no teste de hipótese. Em geral indica uma igualdade a ser contestada. A hipótese alternativa é a hipótese que será considerada como aceitável, caso a hipótese nula seja rejeitada (Allen, 1990; Magalhães & de Lima, 2002).

Para a execução dos testes *t-Student* e geração de gráficos para melhor visualização dos resultados, utilizou-se o software estatístico MINITABTM². Foram gerados gráficos de todas as medidas *versus* a medida MAD, de modo a verificar quais não possuem diferença significativa com a MAD.

5.4.2 Resultados

Os testes foram realizados comparando a MAD com as três estratégias, considerando também todas as variações em cada estratégia.

Experimentos com a Base de Dados Abalone

Na Tabela 5.4 são mostrados alguns dos resultados obtidos para as regras geradas pela ferramenta Cubist. Esses resultados estão relacionados à primeira partição da base de dados Abalone. Como já esperado, para cada regra gerada os erros calculados considerando as variações em cada estratégia proposta são diferentes. Cada linha da tabela representa os resultados obtidos para uma regra de regressão.

Tabela 5.4: Valores para as diferentes medidas de erro para regras da base de dados Abalone.

Número Regra	MAD	Ajustamento		Pseudo-Classe		Discretização Baseada em Entropia			
		Desvio Padrão	DP e Média	DP e Mediana	3 pseudo- classes	6 pseudo- classes	3 intervalos MineSet	6 intervalos MineSet	MineSet automático
R0001	0,00	0,34	0,10	0,13	0,22	0,47	0,32	0,56	0,35
R0002	0,34	0,45	0,14	0,18	0,38	0,66	0,42	0,64	0,44
R0003	0,56	0,31	0,12	0,16	0,25	0,55	0,08	0,26	0,44
R0004	1,00	0,44	0,15	0,18	0,34	0,61	0,13	0,30	0,58
R0005	0,81	0,45	0,13	0,19	0,24	0,55	0,05	0,19	0,52

Na Figura 5.2 é mostrado um gráfico de quatro medidas: a MAD e uma para cada estratégia (desvio padrão, 6 pseudo-classes e 6 intervalos do MineSetTM). De cada estratégia foi escolhida a variação que apresentou o melhor resultado. Cada ponto no gráfico é o valor da medida em questão para uma dada regra. No gráfico são apresentadas 55 das 111 regras de regressão obtidas, sendo as 5 primeiras regras as mesmas apresentadas na Tabela 5.4.

Porém, pode-se perceber que somente a partir da análise visual do gráfico não se consegue chegar a uma conclusão confiável sobre os resultados, devido, em parte, à grande quantidade de informação nele representada.

Visando facilitar a análise dos resultados foram construídos gráficos *boxplot*. Na Figura 5.5 são apresentados todos os gráficos da diferença entre a medida MAD e as estratégias para a base de dados Abalone. Pode-se ver que a hipótese H_0 não foi rejeitada quando

²<http://www.minitab.com>

se utiliza a estratégia de desvio padrão e a 6 Intervalos Mineset, pois H_0 está dentro do intervalo de confiança (Figuras 5.5(a) e (g)).

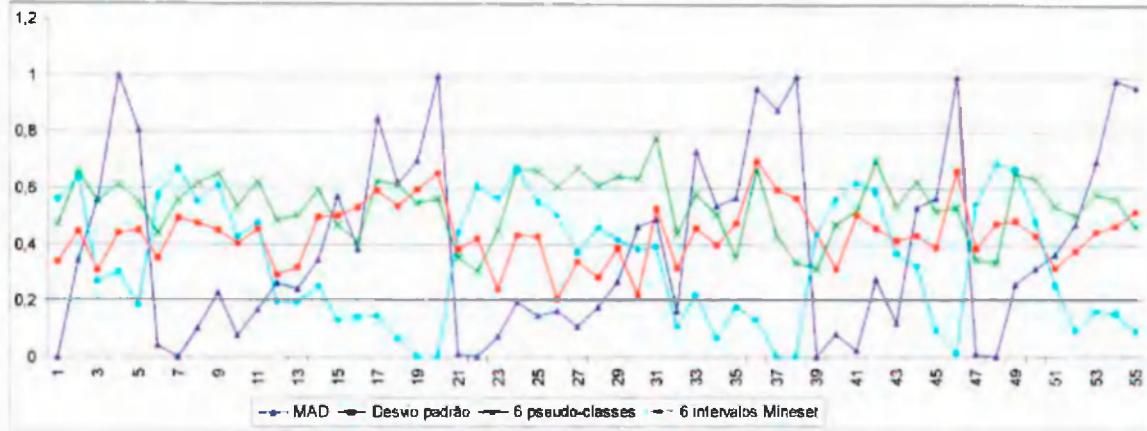


Figura 5.2: Gráfico de quatro medidas para as regras da base de dados Abalone.

Experimentos com a Base de Dados Auto-MPG

A Tabela 5.5 e o gráfico na Figura 5.3 apresentam os resultados, como os já explicados anteriormente, para a base de dados Auto-MPG. Nesse caso, na Tabela 5.5 também são apresentadas somente as regras obtidas a partir da primeira partição da base de dados Auto-MPG. Além disso, os resultados das primeiras 46 das 93 regras são apresentados na Figura 5.3 para as mesmas quatro medidas: MAD, desvio padrão, 6 pseudo-classes e 6 intervalos do Mineset.

Tabela 5.5: Valores para as diferentes medidas de erro para regras da base de dados Auto-MPG.

Número Regra	Ajustamento				Pseudo-Classe		Discretização Baseada em Entropia		
	MAD	Desvio Padrão	DP e Média	DP e Mediana	3 pseudo-classes	6 pseudo-classes	3 intervalos Mineset	6 intervalos Mineset	Mineset automático
R0001	0,17	0,51	0,14	0,16	0,06	0,35	0,00	0,09	0,32
R0002	0,00	0,58	0,25	0,25	0,08	0,25	0,00	0,17	0,25
R0003	0,18	0,19	0,00	0,05	0,38	0,43	0,00	0,29	0,33
R0004	0,32	0,43	0,14	0,18	0,14	0,39	0,04	0,14	0,43
R0005	0,31	0,50	0,13	0,13	0,06	0,41	0,22	0,22	0,50
R0006	0,56	0,50	0,14	0,14	0,32	0,32	0,27	0,27	0,50
R0007	0,10	0,59	0,13	0,16	0,19	0,22	0,03	0,03	0,31
R0008	0,08	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,75
R0009	0,53	0,63	0,16	0,18	0,03	0,16	0,24	0,26	0,63
R0010	1,00	0,60	0,13	0,13	0,00	0,00	0,40	0,67	0,87

Já na Figura 5.6 são apresentados todos os gráficos *boxplot* da diferença entre a medida MAD e todas as estratégias para base de dados Auto-MPG. Nesses casos, observa-se que a hipótese H_0 não foi rejeitada, ou seja, não existe diferença significativa entre as medidas para as estratégias que utiliza o somente o desvio padrão e o Mineset automático. Isso pode ser visto, uma vez que H_0 está contido no intervalo de confiança (Figuras 5.6(a) e (h)).

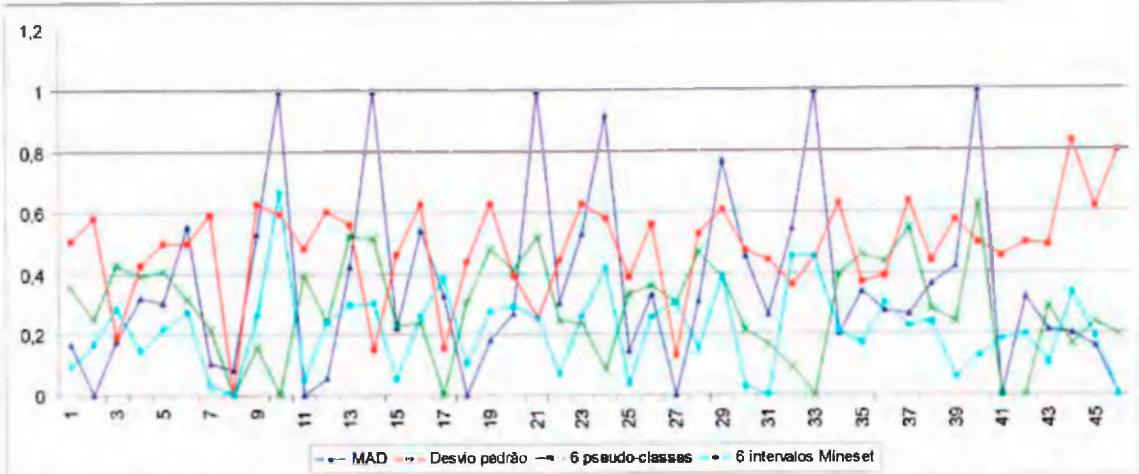


Figura 5.3: Gráfico de quatro medidas para as regras da base de dados Auto-MPG.

Experimentos com a Base de Dados Housing

A Tabela 5.6 e o gráfico na Figura 5.4 apresentam os resultados para a base de dados **Housing**. Nesse caso, na Tabela 5.6 também são apresentadas somente as regras obtidas a partir da primeira partição da base de dados **Housing**.

Além disso, os resultados das primeiras 45 das 91 regras geradas são apresentados na Figura 5.4 para as medidas MAD, desvio padrão, 6 pseudo-classes e 6 intervalos do Mineset.

Tabela 5.6: Valores para as diferentes medidas de erro para regras da base de dados **Housing**.

Número Regra	Ajustamento				Pseudo-Classe		Discretização Baseada em Entropia		
	MAD	Desvio Padrão	DP e Média	DP e Mediana	3 pseudo- classes	6 pseudo- classes	3 intervalos Mineset	6 intervalos Mineset	Mineset automático
R0001	0,18	0,01	0,00	0,00	0,05	0,28	0,30	0,48	0,50
R0002	0,00	0,51	0,13	0,17	0,05	0,21	0,19	0,36	0,49
R0003	1,00	0,60	0,20	0,20	0,20	0,40	0,20	0,60	0,60
R0004	0,57	0,28	0,02	0,05	0,10	0,28	0,01	0,06	0,68

Já na Figura 5.7 os gráficos *boxplot* representam a diferença entre a medida MAD e todas as estratégias utilizadas para a base de dados **Housing**. Para essa base, nada pode ser concluído, pois a hipótese H_0 foi rejeitada para todas as estratégias, ou seja, existe diferença significativa entre as medidas. Nos gráficos isso é mostrado pelo fato de H_0 não estar contido no intervalo de confiança.

5.5 Considerações Finais

Entre os tópicos de Mineração de Dados que têm recebido grande atenção por diversos pesquisadores estão a avaliação tanto do desempenho quanto da qualidade do conheci-

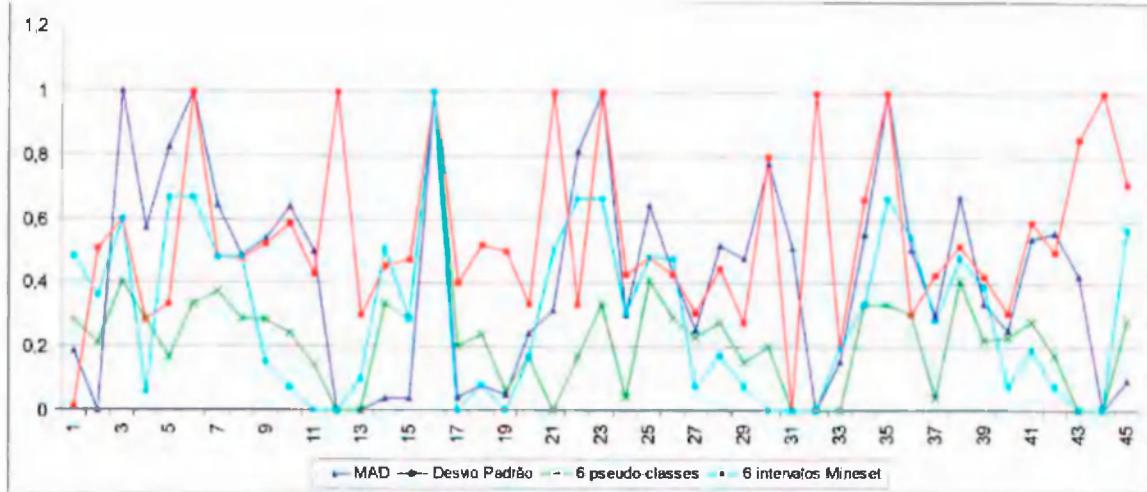


Figura 5.4: Gráfico de quatro medidas para as regras da base de dados Housing.

mento obtido.

Se é qualificada como conhecimento, a saída de um algoritmo de aprendizado deve ser precisa, estável e compreensível. A fim de verificar se o conhecimento possui tais características, após adquirido, um pós-processamento deve ser realizado para avaliar o desempenho e a qualidade do mesmo. Essa avaliação procura investigar a precisão dos algoritmos, a representação do modelo, a complexidade e a dificuldade de entendimento do conhecimento extraído.

Uma das formas de se obter diversas medidas para avaliação de conhecimento é por meio da matriz de contingência. Porém, para problemas de regressão, a construção dessa matriz não é trivial por ser o atributo meta contínuo. Visando solucionar o problema da matriz de contingência para regressão, foram propostas três estratégias para construção dessa matriz. Em cada estratégia também foram consideradas algumas variações.

Após a execução de todos os experimentos para três bases de dados, pode-se verificar que, comparando-se com a medida MAD, a única medida que não rejeitou a hipótese H_0 , ou seja, não houve diferença significativa entre elas em duas das três bases de dados foi a que utilizou a estratégia de ajustamento considerando somente o desvio padrão.

Assim, a matriz de contingência calculada usando a estratégia de ajustamento aqui proposta leva a um comportamento semelhante ao de uma medida consagrada para avaliar o erro de regras de regressão, que é a medida MAD.

Com isso, há um aumento no número de medidas que podem ser usadas para avaliar regras de regressão, uma vez que todas as já definidas e derivadas da matriz de contingência podem então ser utilizadas.

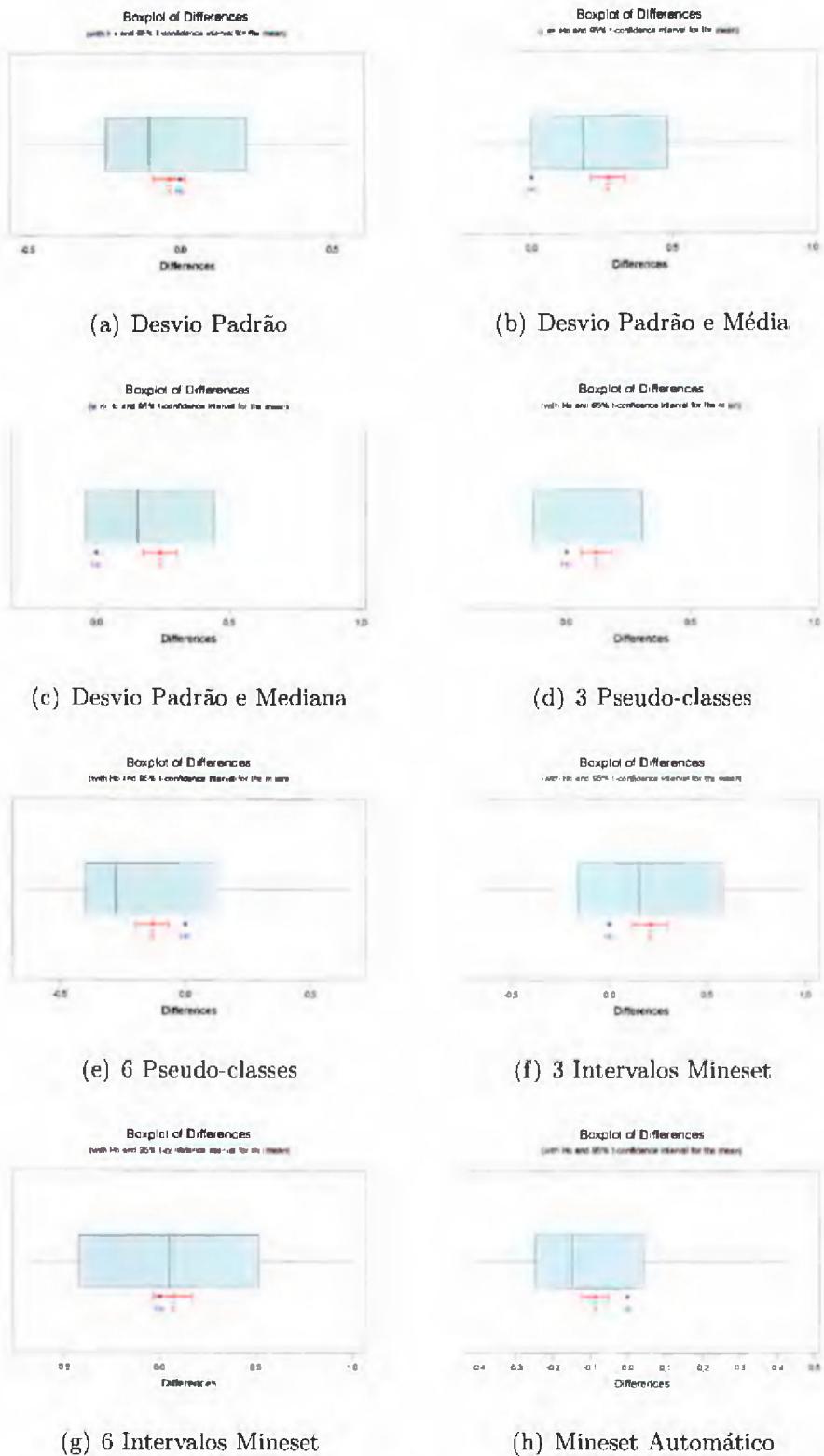
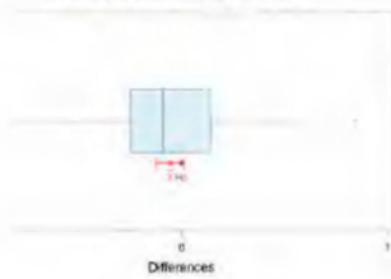


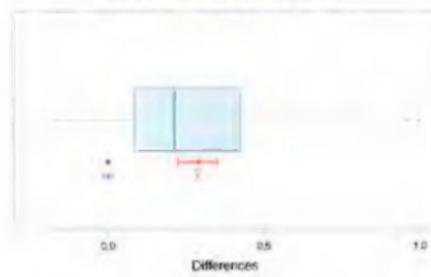
Figura 5.5: Boxplot da base de dados **Abalone**.

Boxplot of Differences
(with H_0 and 95% t-confidence interval for the mean)



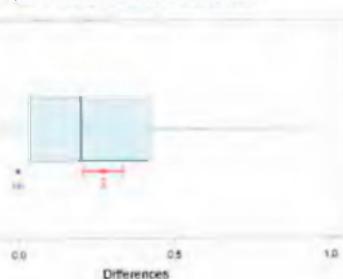
(a) Desvio Padrão

Boxplot of Differences
(with H_0 and 95% t-confidence interval for the mean)



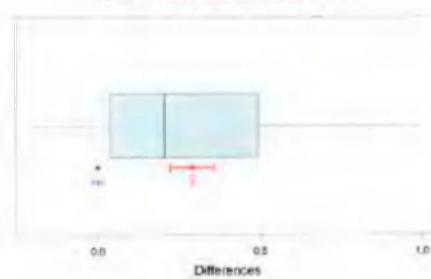
(b) Desvio Padrão e Média

Boxplot of Differences
(with H_0 and 95% t-confidence interval for the mean)



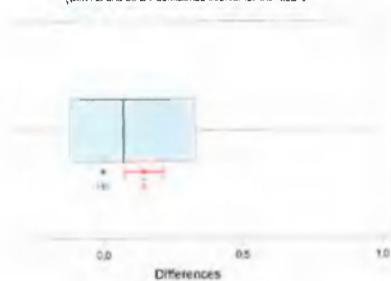
(c) Desvio Padrão e Mediana

Boxplot of Differences
(with H_0 and 95% t-confidence interval for the mean)



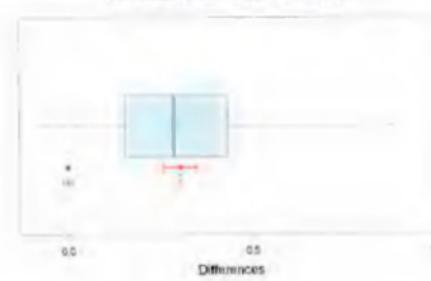
(d) 3 Pseudo-classes

Boxplot of Differences
(with H_0 and 95% t-confidence interval for the mean)



(e) 6 Pseudo-classes

Boxplot of Differences
(with H_0 and 95% t-confidence interval for the mean)



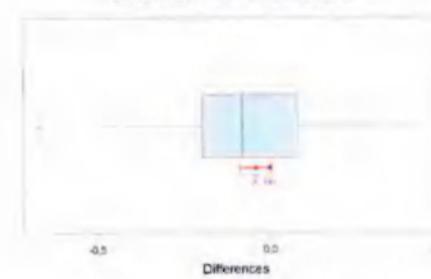
(f) 3 Intervalos Mineset

Boxplot of Differences
(with H_0 and 95% t-confidence interval for the mean)



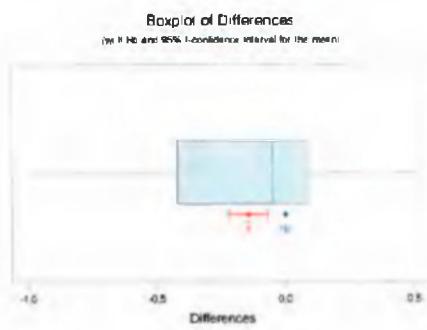
(g) 6 Intervalos Mineset

Boxplot of Differences
(with H_0 and 95% t-confidence interval for the mean)

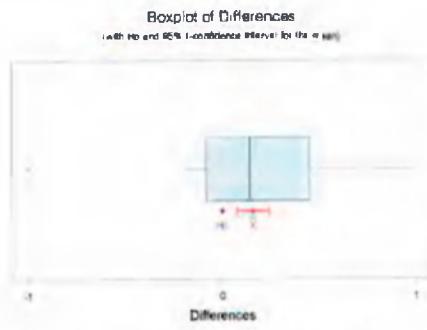


(h) Mineset Automático

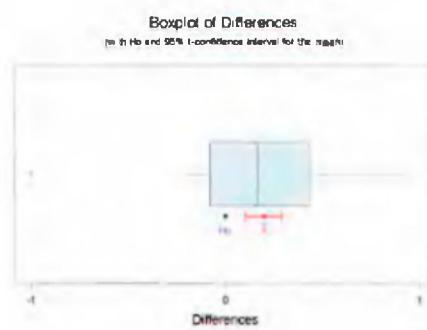
Figura 5.6: Boxplot da base de dados Auto-MPG.



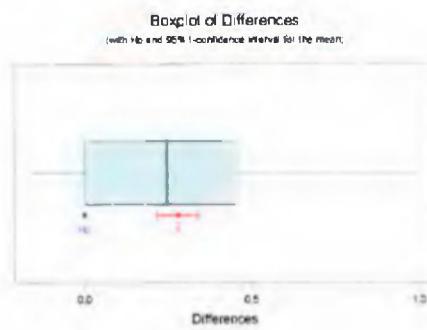
(a) Desvio Padrão



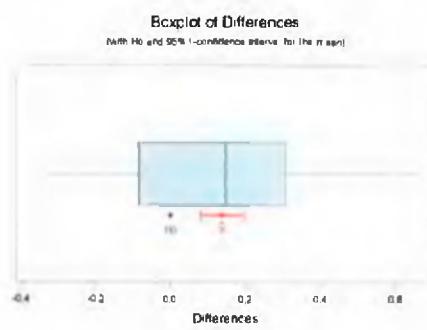
(b) Desvio Padrão e Média



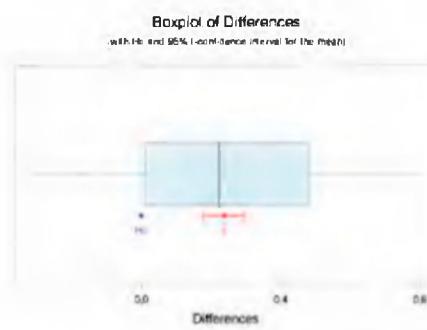
(c) Desvio Padrão e Mediana



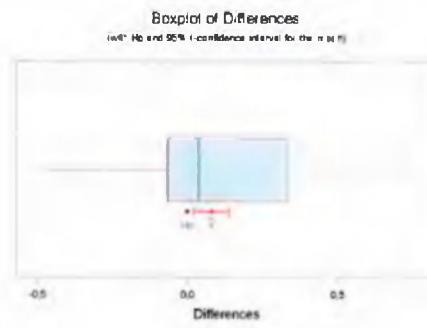
(d) 3 Pseudo-classes



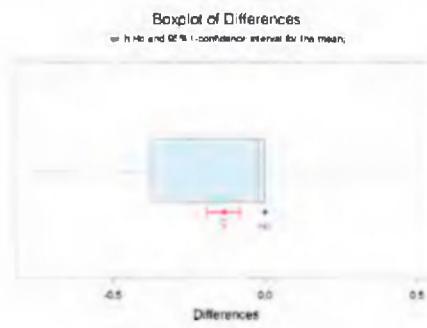
(e) 6 Pseudo-classes



(f) 3 Intervalos Mineset



(g) 6 Intervalos Mineset



(h) Mineset Automático

Figura 5.7: Boxplot da base de dados *Housing*.

Combinação de Regressores

Durante a etapa de pós-processamento, normalmente, é selecionado o preditor que fornece a maior precisão (ou menor erro) para a aplicação. Porém, se for interessante aumentar a precisão do conhecimento extraído, pode ser utilizada a combinação da saída de diversos preditores. Quando essa estratégia é utilizada, o preditor resultante dessa combinação é chamado de *ensemble* (LeBlanc & Tibshirani, 1996; Breiman, 2000a).

A combinação de preditores tem sido recomendada principalmente para problemas que não foram resolvidos satisfatoriamente por um único preditor e/ou no qual o desempenho apresentado pelo método de predição deve ser estável. Pesquisas com a utilização de *ensembles* para regressão e classificação têm mostrado que é possível obter, por meio da combinação de preditores, resultados com maior precisão quando comparados com o resultado obtido com um único preditor. Vários métodos para realizar combinações vêm sendo pesquisados. Porém, os resultados obtidos são essencialmente empíricos, com pouca teoria para explicar o porque e como o erro de generalização é menor na maioria dos conjuntos de dados (Breiman, 2000a).

Segundo Dietterich (Dietterich, 2000a), no geral é possível construir bons *ensembles*, que possuam maior precisão que um preditor simples. Isso se deve fundamentalmente a três razões.

A primeira delas é estatística. Um algoritmo de aprendizado faz uma busca no espaço H de hipóteses para identificar a melhor. O problema estatístico aparece quando o conjunto de treinamento é muito pequeno comparado ao espaço de hipóteses. Sem uma

quantidade suficiente de dados, o algoritmo de aprendizado pode encontrar diferentes hipóteses em H com a mesma precisão no conjunto de treinamento. Construindo um *ensemble* com todos esses preditores, o algoritmo pode fazer uma média dos resultados dos mesmos e reduzir o risco de não escolher o preditor certo.

Na Figura 6.1(a), a situação descrita, envolvendo o problema estatístico, pode ser visualizada. A curva externa representa o espaço de hipóteses H enquanto o conjunto de hipóteses que oferecem uma boa precisão no conjunto de treinamento é representado pela curva interna. O ponto f é a hipótese verdadeira. Pode-se notar que fazendo uma média entre as hipóteses que possuem maior precisão, obtém-se uma boa aproximação para f .

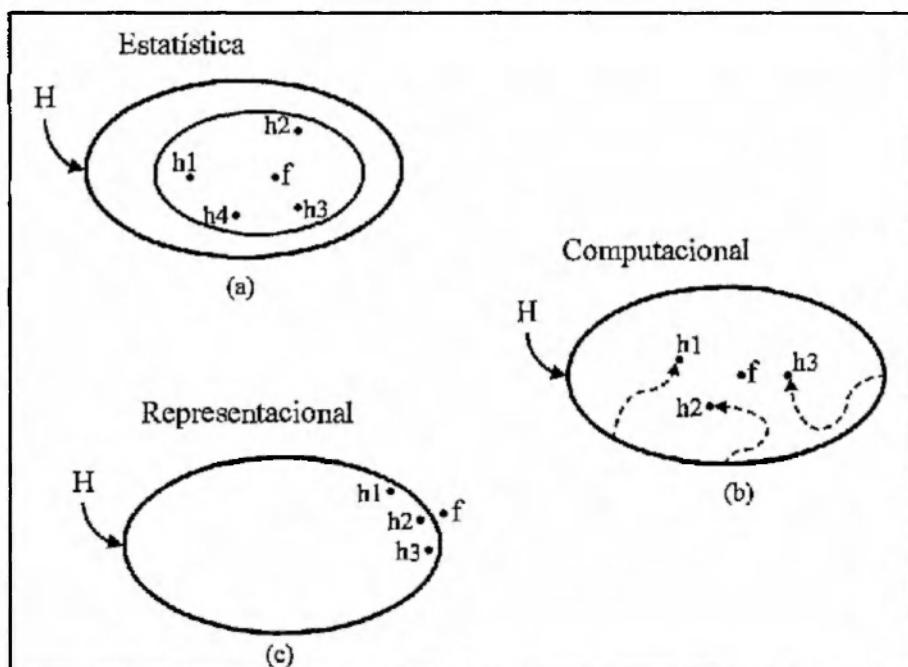


Figura 6.1: Três razões do porque um *ensemble* pode trabalhar melhor que um preditor simples (Dietterich, 2000a).

A segunda razão é computacional. Muitos algoritmos de aprendizado utilizam alguma forma de busca restrita que pode parar em um ótimo local. Nos casos em que se possui dados suficientes no conjunto de treinamento (problema estatístico é ausente), ainda assim pode ser muito difícil para o algoritmo de aprendizado encontrar a melhor hipótese. Um *ensemble* construído com a busca local partindo de diferentes pontos pode obter uma melhor aproximação da verdadeira função desconhecida que qualquer um dos preditores individuais, como pode ser visto na Figura 6.1(b).

A terceira razão é representacional. Na maioria das aplicações de Aprendizado de Máquina, a hipótese verdadeira f pode não ser representada por nenhuma das hipóteses no espaço H . Calculando uma soma ponderada entre as hipóteses retiradas de H , pode ser possível ampliar o espaço das funções representáveis. A Figura 6.1(c) ilustra essa

situação.

O tópico representacional é um pouco sutil, pois há muitos algoritmos de aprendizado para os quais H é, em princípio, o espaço de todos os possíveis preditores. Dado um conjunto de treinamento suficiente, eles exploram esse espaço. Por outro lado, com uma amostra de treinamento finita, esses algoritmos exploram apenas um conjunto de hipóteses finito e finalizam a busca quando encontram uma hipótese que satisfaça o conjunto de treinamento. Por essa razão, na Figura 6.1, deve-se considerar o espaço H como o espaço das hipóteses efetivamente pesquisado pelo algoritmo de aprendizado para um dado conjunto de treinamento.

O uso de *ensembles* tendem a reduzir esses três tipos de falhas dos algoritmos de aprendizado.

O preditor combinado é caracterizado pela seleção de alguns preditores e pela função de combinação. Como a combinação de preditores depende desses dois fatores, vários caminhos podem ser utilizados visando um bom resultado. Os preditores a serem combinados podem ser extraídos a partir do mesmo conjunto de treinamento ou de várias amostras de uma base de dados. Além disso, pode-se gerar preditores utilizando o mesmo algoritmo de Aprendizado de Máquina, chamados de preditores homogêneos, ou fazendo uso de diferentes algoritmos, sendo chamados de preditores heterogêneos.

A combinação de preditores é uma maneira de pós-processamento de conhecimento que pode ser realizada quando se tem como objetivo principal a melhora da precisão do preditor. Como o foco dessa tese é em tarefa de regressão, neste capítulo é apresentada a combinação de regressores, visando melhorar a precisão dos mesmos.

Quanto à organização, este capítulo está dividido da seguinte maneira: nas Seções 6.1 e 6.2 são abordadas, respectivamente, técnicas para combinação de preditores homogêneos e heterogêneos. Na Seção 6.3 são apresentados os métodos *voting* e *não-voting*, bem como os lineares e não-lineares para combinação de preditores. Na Seção 6.4 são apresentados os algoritmos implementados e utilizados para a execução de três técnicas para combinação de regressores: *bagging*, *boosting* e *stacking*. A seleção por problemas de regressão se deve ao fato de ser esse tipo de problema o foco principal desta tese. Os experimentos realizados, bem como os resultados obtidos são apresentados na Seção 6.5. Por fim, na Seção 6.6 encontram-se as considerações finais deste capítulo.

6.1 Combinando Preditores Homogêneos

Nessa seção serão analisadas as técnicas que combinam modelos gerados por um único tipo de algoritmo de aprendizado. Levando-se em consideração que a diversidade é um dos requisitos quando se utiliza combinação de preditores, várias estratégias têm sido pro-

postas para a geração de diferentes preditores usando o mesmo algoritmo de aprendizado. A maioria delas manipula o conjunto de treinamento para gerar múltiplas hipóteses. O algoritmo de aprendizado é executado várias vezes, e em cada uma delas é usada uma distribuição diferente dos exemplos de treinamento, como mostrado na Figura 6.2. Essa técnica funciona especialmente bem para algoritmos de aprendizado instável, ou seja, algoritmos cujo preditor de saída sofre grandes mudanças em resposta a pequenas alterações nos dados de treinamento (Breiman, 1996a; Dietterich, 2000b).

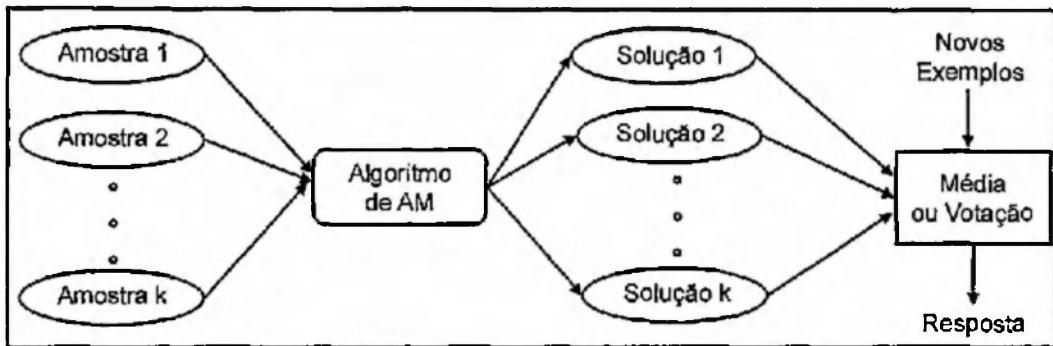


Figura 6.2: Combinação de preditores homogêneos.

Nas próximas seções serão descritas algumas técnicas para combinação de preditores homogêneos, bem como alguns algoritmos referentes a essas técnicas, visto que cada técnica pode ser implementada de uma ou mais maneiras.

6.1.1 *Bagging*

A técnica descrita por Breiman em 1996 (Breiman, 1996a), que recebeu o nome de *bootstrap aggregating* (ou simplesmente *bagging*), realiza a combinação de preditores gerados pelo mesmo algoritmo de Aprendizado de Máquina. De acordo com essa técnica, dado um conjunto de treinamento S , com n exemplos, são geradas várias amostras desse conjunto, chamadas *bootstrap*. Cada *bootstrap* é formado tomando aleatoriamente n elementos de S , com substituição. Assim, eles possuem o mesmo número de exemplos de S , sendo que alguns exemplos podem aparecer mais de uma vez e outros podem não aparecer.

Os *bootstraps* formados são usados como novos conjuntos de treinamento, dando origem a vários preditores, como apresentado na Figura 6.3. O algoritmo *bagging* gera preditores em paralelo. A fim de gerar o preditor final, que é utilizado para prever o conjunto de teste, todos os preditores são combinados por meio de uma votação não ponderada (no caso de classificação) ou do cálculo da média (no caso de regressão), como pode ser visto no Algoritmo 1.

Em um *bootstrap*, a probabilidade de um exemplo ser selecionado pelo menos uma vez dentre os n exemplos do conjunto de treinamento é dada por $1 - (1 - 1/n)^n$. Para um n

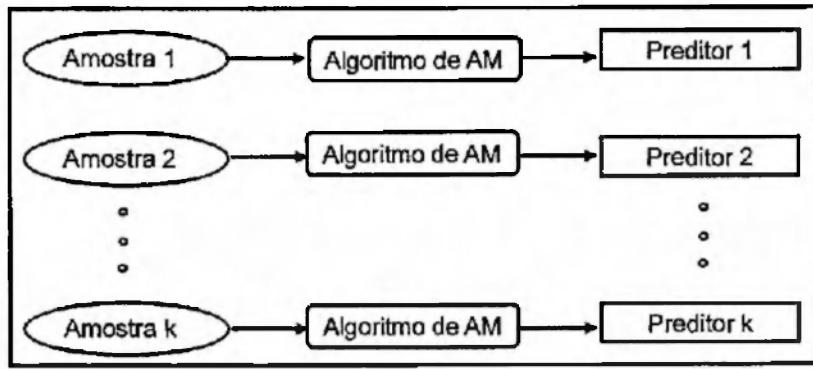


Figura 6.3: *Bagging* – Geração paralela de preditores.

Algorithm 1 Algoritmo *Bagging* (Fonte: (Bauer & Kohavi, 1999)).

Entradas:

conjunto de treinamento S

algoritmo de aprendizado I

constante inteira T {número de *bootstrap*}

```

1: for  $i := 1$  to  $T$  do
2:    $S' := \text{bootstrap}(S)$ 
3:    $P_i := I(S')$ 
4: end for
5:  $P^*(x) := \arg \max_{y \in Y} \sum_{i=1}^T P_i(x) = y$ 

```

Saída: Predictor P^*

grande, essa probabilidade é cerca de $1 - 1/e$. Isso significa que, em média, cada *bootstrap* possui 63,2% do conjunto de treinamento original, com muitos exemplos aparecendo várias vezes (Bauer & Kohavi, 1999).

Geralmente, a técnica *bagging* aumenta o desempenho de algoritmos instáveis, ou seja, algoritmos cuja saída seja um preditor que obtenha grandes mudanças em resposta a pequenas alterações ocorridas no conjunto de treinamento. São exemplos de algoritmos instáveis as árvores de decisão e as Redes Neurais. Se o algoritmo de AM expressa o conceito como árvore de decisão, *bagging* gera ao menos duas situações, devido a resposta do preditor a pequenas mudanças no conjunto de treinamento. A primeira é a escolha do ponto de corte, para atributos contínuos. A outra é a escolha do atributo de corte de cada nó. Uma pequena mudança no conjunto de treinamento pode mudar a escolha do atributo e, consequentemente todas as sub-árvores descendentes também mudarão.

O motivo pelo qual *bagging* reduz a taxa de erro de algoritmos instáveis tem sido tema de alguns trabalhos. Domingos (1997) realizou testes empíricos com o objetivo de conhecer o motivo pelo qual a taxa de erro de árvores de decisão e outros algoritmos é reduzida com o uso de *bagging*. Para isso, duas alternativas baseadas na teoria de Apren-

dizado Bayesiano foram testadas. A primeira hipótese é que *bagging* funciona bem por ser uma aproximação do ótimo procedimento do modelo Bayesiano médio, com uma prévia aproximação implícita. A segunda é que *bagging* funciona bem, pois troca efetivamente o modelo espacial do algoritmo de Aprendizado de Máquina e/ou a prévia distribuição para uma que melhor se encaixe no domínio. Os experimentos realizados por esse pesquisador contradizem a primeira, confirmado a segunda hipótese.

Assim, *bagging* pode ser uma maneira simples de melhorar a precisão em problemas de Mineração de Dados (Gama, 1999). Para isso, adiciona-se um *loop* antes de selecionar o *bootstrap* e mandá-lo para o algoritmo de aprendizado e, por fim, fazer a combinação dos preditores gerados.

Vale ressaltar que, com o uso de *bagging*, ao mesmo tempo em que se ganha precisão obtém-se, comparado às árvores de decisão, a perda de uma estrutura simples e interpretável.

6.1.2 Boosting

A técnica *Boosting* (Freund & Shapire, 1996) engloba uma família de técnicas utilizadas para aumentar o desempenho de qualquer algoritmo de aprendizado. Na teoria, *boosting* pode ser usado para reduzir o erro de um algoritmo “fraco”¹, cujos preditores devem ser apenas um pouco melhor que um preditor que prediz de forma aleatória.

Boosting combina preditores gerados pelo mesmo algoritmo de Aprendizado de Máquina utilizando diferentes distribuições do conjunto de treinamento. A distribuição do conjunto de treinamento é alterada de acordo com os erros cometidos pelo preditor anterior, ou seja, a probabilidade de selecionar um exemplo não é igual para todos os exemplos do conjunto de treinamento. Portanto, *boosting* é uma tentativa de produzir novos preditores, que são melhores, principalmente, por predizerem corretamente exemplos para os quais a eficiência dos preditores comuns é ruim.

O uso da técnica *boosting* normalmente é útil em problemas que possuem as seguintes propriedades: o algoritmo de aprendizado é sensível a mudanças no conjunto de treinamento e os exemplos observados possuem vários níveis de dificuldade, propriedade que pode ser encontrada em problemas reais.

Um dos algoritmos de *boosting* mais utilizado é o *AdaBoost* (*Adaptive Boosting*), apresentado por Freund e Schapire (Freund & Schapire, 1997). Dado um conjunto de treinamento S , o algoritmo *AdaBoost* atribui um peso $w_1(n)$ a cada um de seus n exemplos, gerando um novo conjunto de treinamento S'_1 . Em cada iteração k , os pesos $w_k(n)$ são alterados e um novo conjunto S'_k é gerado, com base em S'_{k-1} . A partir de cada um desses

¹Algoritmo de aprendizado que possui um desempenho apenas um pouco melhor que a estimativa aleatória.

conjuntos de treinamento um preditor é gerado. Note que *AdaBoost* gera preditores em série, como é ilustrado na Figura 6.4, de maneira diferente do *bagging*, que os gera em paralelo.

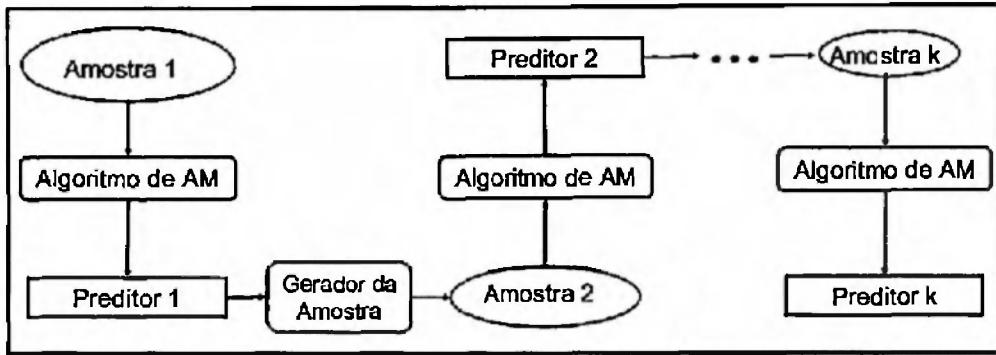


Figura 6.4: *Boosting* — Geração de preditores em série.

O *AdaBoost* usa os pesos para construir um novo conjunto de treinamento S'_n , sendo que duas maneiras têm se mostrado muito eficazes (Quinlan, 1996). Na primeira, denominada *boosting* por amostragem, os exemplos são tomados de S'_{n-1} com substituição e probabilidade proporcional aos seus pesos. A segunda, *boosting* por peso, pode ser usado com um algoritmo de AM que aceite diretamente um conjunto de treinamento ponderado.

Em qualquer um dos casos, a cada iteração os pesos são ajustados de acordo com o desempenho do preditor anterior. Os exemplos que foram preditos de forma errada têm o peso aumentado e os que foram preditos corretamente têm o peso diminuído. Com isso, os preditores focalizam os exemplos mais difíceis de serem preditos, ou seja, aqueles que foram preditos de forma errônea pelo preditor anterior. Desta maneira, o algoritmo *AdaBoost* torna os preditores cada vez mais “fortes”.

Após todas as iterações serem realizadas, a resposta do preditor final é formada por meio de uma votação ou média ponderada. Nessa votação, o peso do voto de cada preditor depende do seu desempenho no conjunto de treinamento usado para construí-lo. Uma descrição do algoritmo *AdaBoost* é apresentado no Algoritmo 2.

6.1.3 Windowing

A técnica *windowing* seleciona um subconjunto de exemplos do conjunto de treinamento (uma janela) e gera uma hipótese a partir desse subconjunto. Essa hipótese é então utilizada para prever os exemplos de treinamento restantes, ou seja, aqueles que não foram incluídos na janela. Caso existam exemplos mal preditos, alguns deles são adicionados à janela inicial e uma segunda hipótese é construída a partir da janela ampliada. Esse ciclo é repetido até que a hipótese construída a partir da janela corrente classifique corretamente todos os exemplos de treinamento fora da janela ou o número de

Algorithm 2 Algoritmo *AdaBoost* (Fonte: (Schapire, 1999)).

Entradas:
conjunto de treinamento S
algoritmo “fraco” I
constante inteira T {número de *bootstrap*}

```
1: for  $i := 1$  to  $n$  do
2:    $w_1(i) = 1/n$  {inicializa os pesos dos exemplos de  $S$ }
3: end for
4: for  $i := 1$  to  $T$  do
5:    $P_i := I(S)$  {utilizando a distribuição  $D_i$ }
6:    $E_i := \sum_{j: P_i(x_j) \neq y_j} D_i(j)$ 
7:    $\alpha_i := \frac{1}{2} \ln \frac{1-E_i}{E_i}$ 
8:   for  $k := 1$  a  $n$  do
9:      $w_{i+1}(k) := \frac{w_i(k) \exp(-\alpha_i y_k P_i(x_k))}{Z_i}$  { $Z_i$  é um fator normalizador}
10:    end for
11: end for
12:  $P^* := \text{sign}\left(\sum_{i=1}^n \alpha_i P_i(x)\right)$ 
```

Saída: Predictor P^*

ciclos exceda um valor pré-definido, como apresentado no Algoritmo 3.

O que frequentemente ocorre ao se utilizar *windowing* com um algoritmo de aprendizado de regras é que as regras boas têm que ser descobertas várias vezes em iterações subsequentes. Mesmo que as regras corretamente aprendidas não acrescentem exemplos à janela corrente, elas tem que ser re-aprendidas na próxima iteração enquanto a teoria corrente não estiver completa e consistente com todo o conjunto de treinamento.

Foi desenvolvida uma nova versão da técnica *windowing*, que explora o fato de que regiões do espaço de exemplos que já estão cobertas por regras boas não necessitam mais ser consideradas em iterações subsequentes (Furnkranz, 1998). Devido à integração sucessiva de regras de aprendizado na teoria final, essa nova versão foi denominada *Integrative Windowing* e pode ser vista com mais detalhes no Algoritmo 4.

6.1.4 Wagging

Uma variação da técnica *bagging* é o *wagging* (*Weight Aggregation*). Essa técnica procura adicionar ruídos repetidamente ao conjunto de treinamento, como faz o *bagging*, porém ao invés de fazer uma amostragem a partir dele, *wagging* adiciona ruído Gaussiano a cada peso com média zero e um desvio padrão dado (por exemplo, dois) (Bauer & Kohavi, 1999). Para cada teste, inicia-se com pesos uniformes para todos os exemplos, adiciona ruído aos pesos e induz o preditor. Esta técnica possui a característica de que

Algorithm 3 Algoritmo *Windowing* (Fonte: (Fürnkranz, 1998)).

Entradas:

conjunto de treinamento S
tamanho da amostra inicial Tam
número máximo de exemplos que podem ser adicionados a cada iteração Inc

```
1:  $Jan := \text{Amostragem}(S, Tam)$ 
2:  $C := S \setminus Jan$  {o símbolo \ é a diferença entre conjuntos}
3: repeat
4:    $P := I(Jan)$ 
5:    $Jan' := \emptyset$ 
6:    $C' := \emptyset$ 
7:   for  $x_i \in C$  do
8:      $C := C \setminus x_i$ 
9:     if  $\text{Prediz}(P, x_i) \neq \text{Classe}(x_i)$  then
10:       $Jan' := Jan' \cup x_i$ 
11:    else
12:       $C' := C' \cup x_i$ 
13:    end if
14:    if  $|Jan'| = Inc$  then
15:      saída-para
16:    end if
17:   end for
18:    $C := \text{Append}(C, C')$ 
19:    $Jan := Jan \cup Jan'$ 
20: until  $Jan' = \emptyset$ 
```

Saída: Preditor P

Algorithm 4 Algoritmo *Integrative Windowing* (Fonte: (Fürnkranz, 1998)).

Entradas:

conjunto de treinamento S

tamanho da amostra inicial Tam

número máximo de exemplos que podem ser adicionados a cada iteração Inc

```
1:  $Jan := \text{Amostragem}(S, Tam)$ 
2:  $C := S \setminus Jan$ 
3:  $RegrasVelhas := \emptyset$ 
4: repeat
5:    $RegrasNovas := I(Jan)$ 
6:    $P := RegrasNovas \cup RegrasVelhas$ 
7:    $Jan' := \emptyset$ 
8:    $C' := \emptyset$ 
9:   for  $x_i \in C$  do
10:     $C := C \setminus x_i$ 
11:    if  $\text{Prediz}(P, x_i) \neq \text{Classe}(x_i)$  then
12:       $Jan' := Jan' \cup x_i$ 
13:    else
14:       $C' := C' \cup x_i$ 
15:    end if
16:    if  $|Jan'| = Inc$  then
17:      saída-para
18:    end if
19:   end for
20:    $C := \text{Append}(C, C')$ 
21:    $Jan := Jan \cup Jan' \cup \text{Cobertura}(RegrasVelhas)$ 
22:    $RegrasVelhas := \emptyset$ 
23:   for  $R \in P$  do
24:     if  $\text{Consistente}(R, Jan')$  then
25:        $RegrasVelhas := RegrasVelhas \cup \{R\}$ 
26:        $Jan := Jan \setminus \text{Cobertura}(R)$ 
27:     end if
28:   end for
29: until  $Jan' = \emptyset$ 
```

Saída: Predictor P

pode-se trocar *bias* e variância: aumentando o desvio padrão do ruído, mais exemplos terão seus pesos reduzidos a zero e desaparecem, dessa maneira aumentando o *bias* e reduzindo a variância.

6.1.5 Arcing

O termo *Arcing* (*Adaptively Resample and Combine*) foi definido por Breiman (1998) para descrever a família de algoritmos que adaptativamente amostra e combina. As técnicas *arcing* trabalham seqüencialmente, sendo o usuário quem define o número de iterações que serão realizadas.

Inicialmente um peso $w(n)$ é atribuído a cada exemplo do conjunto de treinamento. Em cada iteração, um novo conjunto é selecionado usando esses pesos $w(n)$ e um preditor é gerado. Os pesos $w(n)$ são então atualizados de acordo com a predição feita pelo último preditor. Após todas as iterações os preditores são combinados.

AdaBoost, que é chamado de *arc-fs*, é o principal exemplo de um algoritmo *arcing* e é baseado no algoritmo *boosting*. Outro importante algoritmo *arcing* é chamado de *arc-x4*, mostrado no Algoritmo 5, que é uma invenção *ad hoc* de Breiman (1998). Uma diferença entre esses algoritmos é que o primeiro combina os preditores gerados com uma votação ponderada enquanto o outro não utiliza os pesos nessa combinação.

6.1.6 Randomization

Randomization é uma técnica para combinação de preditores que não conta com a instabilidade de um algoritmo. A idéia é simples: tornar aleatória as decisões internas do algoritmo de aprendizado (Breiman, 2000a; Dietterich, 2000b).

A técnica *randomization* estudada em (Dietterich, 2000b) é bem simples: calcula-se os vinte melhores atributos para ramificar um nó de uma árvore de decisão, e depois selecione-se aleatoriamente um atributo e um valor de corte. O próximo passo consiste em fazer com que a probabilidade de se selecionar um atributo para ramificação seja proporcional ao ganho de informação daquela ramificação. Um outro refinamento seria executar uma *randomization* com “discrepância limitada”, ou seja, no máximo k ramificações seriam randomizadas em uma árvore (para um valor de k especificado).

O valor de k poderia ser ajustado por *cross-validation*. O algoritmo poderia explicitamente fazer 0, 1, 2, ..., k ramificações aleatórias. Isso asseguraria que a “melhor” árvore (isto é, a árvore produzida pelo algoritmo de aprendizado) estaria incluída no *ensemble*. Finalmente, devido ao fato de que a *randomization* possa produzir árvores com diferentes precisões, valeria a pena considerar a adoção de um voto com peso, com o peso determinado pela precisão da árvore nos dados de treinamento.

Algorithm 5 Algoritmo *Arcing* (Fonte: (Breiman, 1998)).

Entradas:

conjunto de treinamento S
algoritmo de aprendizado I
constante inteira T {número de preditores}

```
1: for  $i := 1$  to  $n$  do
2:    $w_i := 1/n$  {inicializa os pesos}
3:    $v_i := 0$  {número de exemplos que não foram preditos corretamente}
4: end for
5: for  $i := 1$  to  $T$  do
6:    $S_i := bootstrap(n, w, S)$ 
7:    $P_i := I(S_i)$ 
8:   for  $j := 1$  to  $n$  do
9:      $v_j := v_j + \|P_i(x_j) \neq y_j\|$ 
10:    end for
11:    for  $j := 1$  to  $n$  do
12:       $w_j := \frac{(1+v_j^4)}{\left(\sum_{l=1}^n (1+v_l^4)\right)}$ 
13:    end for
14: end for
15:  $P^*(x) := \arg \max_{y \in P_1, P_2, \dots, P_k} \sum_{i=1}^T \|P_i(x) = y\|$ 
```

Saída: Predictor P^*

A técnica *randomization* é paradoxal, pois à primeira vista parece aumentar a variância por meio da inserção deliberada de variação nas ramificações da árvore de decisão. Entretanto, também pode ser visto como uma maneira de suavizar os efeitos de várias ramificações igualmente boas por meio da amostragem e posterior votação das mesmas (Dietterich & Kong, 1995).

6.1.7 Error-Correcting Output Codes

A técnica *Error-Correcting Output Codes* (ECOC) foi originalmente projetada para resolver problemas de várias classes pela solução de problemas de duas classes (Dietterich & Bakiri, 1995). ECOC representa as classes com um conjunto de *bits* de saída, no qual cada *bit* codifica uma tarefa de predição binária correspondendo a uma única partição das classes. Algoritmos que utilizam ECOC aprendem uma função correspondendo a cada *bit*. Todas as funções são então combinadas para gerar as predições das classes.

6.2 Combinando Preditores Heterogêneos

A combinação de preditores heterogêneos refere-se à combinação de preditores gerados por diferentes algoritmos de aprendizado, como é ilustrado na Figura 6.5. Já que a diversidade dos preditores é garantida pelo uso de diferentes algoritmos de Aprendizado de Máquina, não é necessário manipular o conjunto de treinamento, podendo ser utilizado o mesmo conjunto para todos os algoritmos de aprendizado.

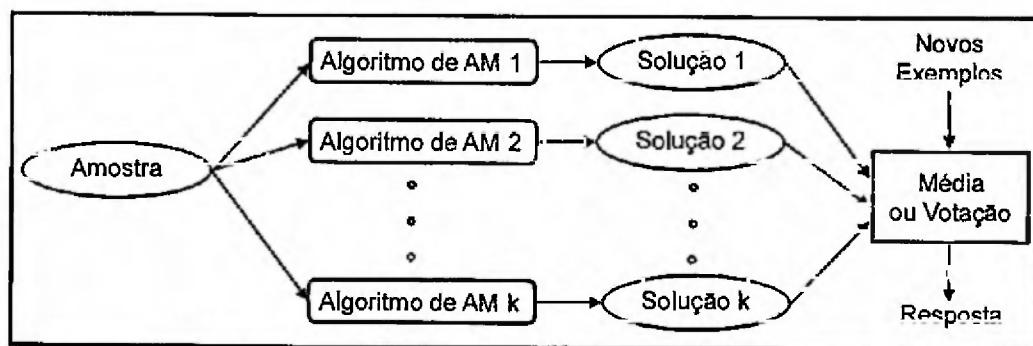


Figura 6.5: Combinação de preditores heterogêneos.

6.2.1 Stacking

Na técnica *stacking*, também chamada de *stacked generalization*, diferentes algoritmos de AM geram preditores com o conjunto de treinamento original. Em seguida, as descrições dos exemplos de treinamento são estendidas para incluir os resultados da predição

desses preditores, formando um novo conjunto de treinamento. Esse novo conjunto é então analisado por outros algoritmos de Aprendizado de Máquina a fim de produzir futuros preditores, e assim por diante.

Wolpert (1992) propôs o framework *stacked generalization*, que é uma arquitetura em camadas. Preditores no nível-0 recebem como entrada o conjunto de dados original e cada um fornece uma predição. Camadas sucessivas recebem como entrada as predições das camadas imediatamente anteriores e fornecem a saída para a próxima camada. Um único preditor no nível mais alto fornece a predição final.

Stacking é uma tentativa de minimizar o erro de generalização por meio do uso de preditores em camadas mais altas para aprender o tipo de erros cometidos pelos preditores imediatamente abaixo. Por essa perspectiva, ele pode ser visto como uma extensão para o modelo de métodos de seleção *cross-validation*. Esse modelo utiliza a estratégia “ganhador fica com tudo”, com isso, apenas um preditor com o menor erro é selecionado. A idéia do *stacking* é que pode existir uma maneira mais inteligente de usar um conjunto de preditores. O papel dos preditores posteriores é aprender como os anteriores erram, em quais exemplos eles concordam ou discordam e usar esse conhecimento ao fazer novas predições (Ting & Witten, 1997).

Trabalhos em arquitetura *stacking*, como (Gama, 1999; Merz, 1998), concentram-se em arquiteturas de duas camadas, como é ilustrada na Figura 6.6.

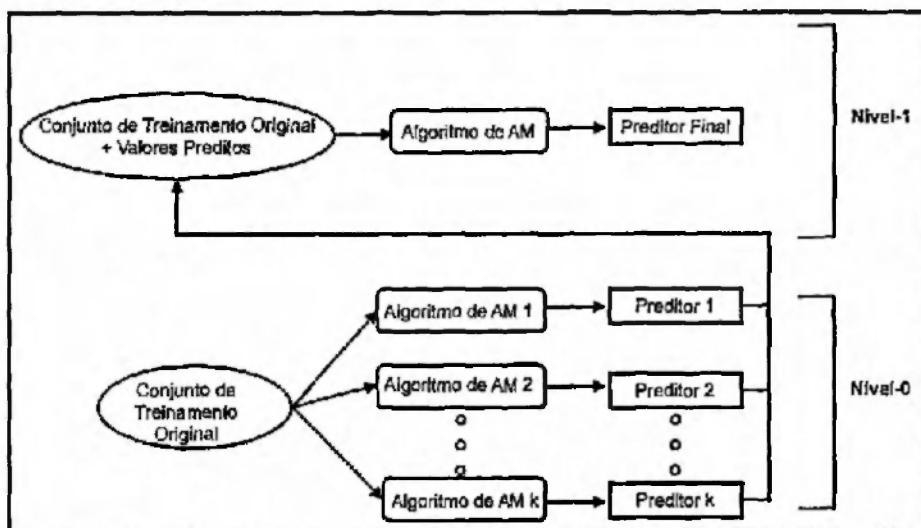


Figura 6.6: *Stacking* (Arquitetura de duas camadas) — Treinamento.

Segundo Gama (1999), experimentos com essa técnica podem ser realizados em duas fases: a fase de treinamento e a fase de aplicação.

A fase de treinamento começa com a geração do conjunto de k pares $S' = \{(z_1, v_1), (z_2, v_2), \dots, (z_k, v_k)\}$, a partir de um conjunto de treinamento S , com n exemplos. Tais pares podem ser gerados de duas maneiras diferentes. Na primeira maneira, proposta por

(Fan, Chan, & Stolfo, 1996), são gerados n pares, ou seja, utiliza-se $k = n$. O primeiro conjunto de cada par (z_i) é formado por um único elemento de S e o segundo (v_i) é formado pelos $n - 1$ elementos restantes. A segunda maneira, utilizada em (Ting & Witten, 1997), gera um número definido de pares k , sendo $k < n$. Divide-se o conjunto S em k partes, sendo cada parte o primeiro conjunto de cada par (z_i). O segundo conjunto (v_i) será formado pelos elementos de S retirando-se os elementos contidos em z_i .

Os algoritmos de nível-0 são utilizados para gerar preditores tendo como conjuntos de treinamento cada conjunto v_i . Os preditores gerados fornecem uma resposta para o(s) exemplo(s) contido(s) no respectivo conjunto z_i .

Para todos os elementos contidos em cada conjunto z_i forma-se um exemplo para o conjunto de treinamento do algoritmo de nível-1. Os exemplos devem conter, além da classe correta, os valores preditos pelos preditores de nível-0 para aquele exemplo. Assim, o número de exemplos do conjunto de treinamento do algoritmo de nível-1 será o mesmo do conjunto de treinamento original. O conjunto é então utilizado para construir um preditor de nível-1, com a utilização de um único algoritmo no nível-1. A resposta desse preditor será a resposta do preditor combinado.

Para explorar completamente o conjunto de treinamento original, todos os preditores de nível-0 são novamente treinados usando todo o conjunto de treinamento (Figura 6.6). Os modelos gerais são utilizados para prever os exemplos do conjunto de teste.

Na fase de aplicação, como é ilustrado na Figura 6.7, quando um novo exemplo é apresentado, ele é calculado por todos os preditores de nível-0. Um vetor de previsões é formado e utilizado no nível-1, que fornece a previsão final do exemplo.

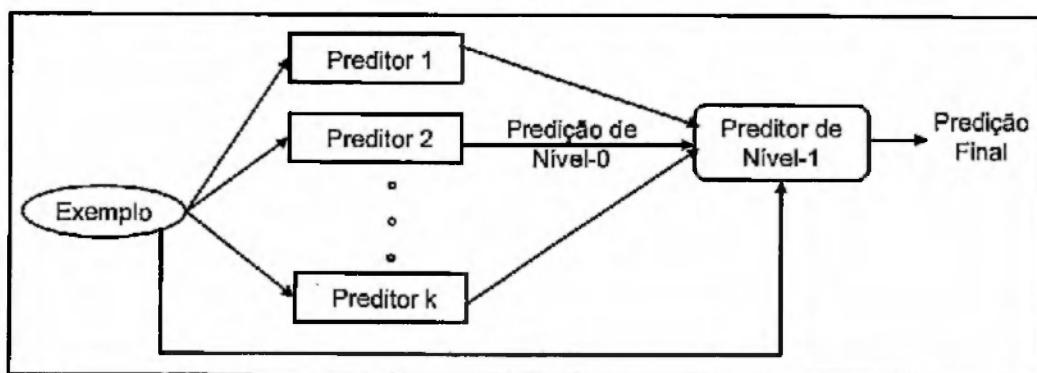


Figura 6.7: *Stacking* — Aplicação.

O *framework* geral não é restrito ao modelo básico descrito. Por exemplo, Breiman (Breiman, 1996b) verificou que em alguns problemas de regressão melhores resultados são obtidos usando *10-fold cross-validation* ao invés de *leave-one-out*.

Uma técnica que pertence à família dos algoritmos *stacking*, no sentido de que o aprendizado ocorre em diversas camadas, é *cascade generalization*.

Cascade generalization é uma composição sequencial de preditores. Em cada passo da sequência, um algoritmo gera um modelo a partir do conjunto de dados de entrada deste passo. Para o passo seguinte é passada a informação do conjunto de dados assim como a informação sobre o modelo criado pelo algoritmo utilizado. Essa informação assume a forma de novos atributos cujos valores correspondem à instanciação do modelo para cada um dos exemplos do conjunto de dados de entrada. Se um dos algoritmos utilizados nos passos seguintes da sequência utilizar um dos atributos gerados em passos anteriores então esse algoritmo estará utilizando termos da linguagem de representação do algoritmo que gerou o atributo.

6.2.2 *Meta Learning*

Chan & Stolfo (1995, 1997) apresentam dois esquemas para combinação de preditores: arbitrado e combinado. Ambos os esquemas são baseados em *meta learning*, no qual um meta preditor é gerado a partir de meta dados, baseado nas previsões dos preditores bases. O árbitro é também um preditor e é utilizado como intermediário entre as previsões geradas pelos diferentes preditores base. O conjunto de treinamento do árbitro é obtido a partir de todos os dados disponíveis usando uma regra de seleção. Um exemplo de uma regra de seleção é "Selecione os exemplos dos quais a previsão não pode ser prevenida consistentemente usando os preditores base". O árbitro, junto a uma regra de arbitragem, determina a previsão final com base nos preditores base. Um exemplo de uma regra de arbitragem é "Utilize a previsão do árbitro quando os preditores base não puderem obter a maioria".

Skalak (1997) discute técnicas para combinação de preditores, e fornece uma análise completa de vários algoritmos, a maioria deles baseada em *stacked generalization*.

6.3 *Métodos para Combinação de Preditores*

No processo de combinação de preditores a função de combinação é baseada em diferentes métodos: os *voting* e não-*voting* (para tarefa de classificação), e os lineares e não-lineares (para tarefa de regressão).

6.3.1 *Métodos Voting versus Não-Voting*

O *voting* é um dos métodos mais comum para combinar preditores. Os algoritmos *voting* têm como entrada um algoritmo de Aprendizado de Máquina e um conjunto de treinamento. Eles geram preditores com diferentes versões desse conjunto, que são combinados para criar um preditor final e este será utilizado para prever o conjunto de teste.

Normalmente o preditor final tem maior precisão do que qualquer um dos preditores que o gerou.

Os algoritmos *voting* podem ser divididos em algoritmos que alteram a distribuição do conjunto de treinamento baseado no desempenho dos preditores anteriores (como o *boosting*) e aqueles que não o fazem (como o *bagging*) (Aas & Eikvil, 1999; Bauer & Kohavi, 1999).

Na decisão, o voto de cada hipótese deve ser ponderado por uma probabilidade posterior àquela hipótese considerando os dados de treinamento. Algumas variantes dos métodos de *voting* podem ser encontradas na literatura de Aprendizado de Máquina. Uma delas é o *voting* uniforme no qual a opinião de todos os preditores bases contribuem com a mesma força para a predição final. Uma outra variação comum é o *voting* com peso, no qual cada preditor base possui um peso associado, que pode alterar com o tempo, e dessa forma fortalecer o atributo de predição para um preditor “ótimo”.

Uma outra forma de combinar preditores é por meio dos métodos não-*voting*. Uma melhora no *voting* uniforme é obtida quando cada preditor pode produzir estimativas de classe-probabilidade ao invés de uma simples decisão de predição. Uma estimativa da probabilidade para cada uma das classes de um dado exemplo é a probabilidade que a classe verdadeira seja k . As probabilidades da classe de todos os modelos podem ser combinadas, e então a probabilidade do *ensemble* pode ser considerada como: $\frac{1}{L} \sum_{i=1}^L p_i$ (Gama, 1999).

6.3.2 Métodos Lineares versus Não-Lineares

Quando a saída de um preditor é um valor real, uma função de combinação linear é comumente utilizada (Breiman, 1996b). A função linear pode ser: simples, média sem peso ou uma combinação de preditores com peso. Algoritmos simples de regressão linear são geralmente utilizados para calcular pesos no componente de predição dos preditores.

As vantagens dos métodos de combinação linear são que eles fazem análise formal mais facilmente; geralmente trabalham bem; e podem diminuir a variância da predição composta.

Por outro lado, uma combinação linear simples pode não refletir a especialidade local de um preditor componente para um exemplo em uma região particular do espaço de atributos. Diante disso, a contribuição de cada componente é a mesma, sem levar em consideração a entrada. Funções lineares falham no suporte de especialidade local, que é um aspecto importante em combinação de preditores.

Apesar da existência de trabalhos em combinar componentes de preditores não-lineares, algoritmos de combinação não-lineares têm recebido bem menos atenção. Funções de com-

binação não-lineares são normalmente implementadas como Redes Neurais, tais como *perceptrons* multicamada ou redes de funções *radial basis*. Não existe *a priori* razão alguma para crer que métodos lineares são adequados para modelar a combinação de regressores, existindo até evidências do contrário. Devido a superioridade seletiva dos preditores em diferentes regiões, é esperado que os componentes terão contribuições relativas distintas em diferentes partes do espaço. Uma função linear dos resultados dos preditores sozinha pode não traçar as contribuições seletivas dos componentes (Skalak, 1997).

Na próxima seção são apresentados os algoritmos utilizados e implementados para combinação de regressores, sendo duas técnicas para combinação de regressores homogêneos (*bagging* e *boosting*) e uma para combinação de regressores heterogêneos (*stacking*).

6.4 Algoritmos para Combinação de Regressores

O objetivo principal deste capítulo é investigar o comportamento das técnicas para combinação de modelos, tanto homogêneos quanto heterogêneos, em problemas de regressão.

A principal dificuldade na combinação de preditores, para problemas de regressão, refere-se ao *boosting*, uma vez que a cada nova iteração deve-se verificar quais exemplos foram cobertos corretamente pelo algoritmo, e regras de regressão não predizem um valor categórico definido e sim, um valor numérico, envolvendo um grau de incerteza quanto à classe predita.

Então, para realizar a combinação de regressores, isto é, preditores com atributo meta numérico, são necessárias algumas adaptações para a utilização das técnicas apresentadas para combinação de preditores (Seções 6.1 e 6.2).

Nesta seção são apresentadas as implementações realizadas para a combinação de regressores homogêneos usando *bagging* e *boosting*. Já para a combinação de regressores heterogêneos a ferramenta Weka disponibiliza uma implementação do *stacking* juntamente com vários algoritmos para problemas de regressão (Witten & Frank, 1999), não sendo necessária uma nova implementação do mesmo. Vale ressaltar que todas as implementações foram realizadas na forma de *script* em PERL. Mais detalhes sobre a combinação de regressores podem ser encontrados em (Pugliesi, Sinoara, & Rezende, 2003). Além desse, detalhes a respeito da teoria e experimentos sobre combinação de classificadores estão detalhados em (Sinoara, Pugliesi, & Rezende, 2003).

6.4.1 Bagging

Para a implementação da técnica *bagging* foram desenvolvidos *scripts* para geração de amostras e para combinação de regressores.

Para a geração de amostras, são criados os arquivos necessários para a posterior execução do algoritmo de regressão em todas as amostras. Conforme o que foi apresentado na Seção 6.1.1, cada amostra, ou *bootstrap*, é formada tomando aleatoriamente elementos do conjunto de dados original, com substituição, sendo as amostras do mesmo tamanho do conjunto de dados original.

Assim, para geração de amostras o *script* recebe como entrada três arquivos (*.data*, *.names* e *.test*), além do número de amostras que devem ser geradas e o nome dos arquivos de saída. Vale ressaltar que esses três arquivos são o conjunto de treinamento, a descrição dos atributos do conjunto de dados e o conjunto de teste, e devem estar na sintaxe padrão de exemplos e de regras de regressão do DISCOVER, definida no Capítulo 4, que são utilizados para gerar um regressor.

A implementação para a combinação dos regressores usando *bagging* é apresentada no Algoritmo 6. Esse *script* recebe como entrada o número de regressores que serão combinados e os arquivos gerados pelo algoritmo de regressão. Esses arquivos contém os valores reais e os preditos por determinado regressor para cada exemplo do conjunto de teste.

A resposta do regressor combinado é a média das previsões de cada regressor e o erro é calculado por meio do módulo da diferença entre esse valor e a previsão real. Dividindo-se o erro dos regressores combinados, que é a soma dos erros dos exemplos, pelo número total de exemplos tem-se a taxa de erro da combinação de regressores. Observe que com a utilização de combinação de regressores a comprehensibilidade do modelo é prejudicada.

6.4.2 Boosting

Com relação à combinação de regressores usando a técnica *boosting* não foi possível utilizar o algoritmo *AdaBoost*, que trata somente de problemas de classificação. Para tanto, a técnica *boosting* foi implementada considerando as particularidades dos problemas de regressão.

Por se tratar de um problema de regressão, como já citado, não é trivial verificar se a previsão está correta ou não. Essa verificação foi realizada com base na metodologia, descrita na Seção 5.3, que utiliza um limiar para definir se um dado exemplo foi corretamente coberto pelo regressor. Com a utilização desse limiar é possível calcular o número de exemplos para o qual a cabeça da regra é verdadeira ou falsa. O ponto crítico nessa

Algorithm 6 Algoritmo para Combinação de Regressores

Entradas:

nome dos arquivos de entrada
constante inteira T {número de regressores a serem combinados}

```
1: Lê os arquivos de entrada
2: for  $i := 1$  a  $n$  do
3:    $preditoFinal := 0$ 
4:    $erroFinal := 0$ 
5:   {Para cada regressor}
6:   for  $i := 1$  a  $T$  do
7:      $predito :=$  valor predito pelo regressor
8:      $preditoFinal := preditoFinal + predito$ 
9:   end for
10:  {Calcula valor predito pelo regressor combinado para o exemplo}
11:   $preditoFinal := \frac{preditoFinal}{T}$ 
12:   $valorReal :=$  valor real da classe do exemplo
13:  {Calcula o erro para o exemplo}
14:   $erro := |valorReal - preditoFinal|$ 
15:   $erroFinal := erroFinal + erro$ 
16: end for
17: {Calcula o erro do regressor combinado}
18:  $erroFinal := \frac{erroFinal}{n}$ 
```

Saída: Taxa de Erro do Regressor Combinado — $erroFinal$

estratégia é a definição do valor do limiar. O limiar utilizado leva em consideração o cálculo do desvio padrão do erro da regra encontrado no conjunto de dados (Pugliesi & Rezende, 2003).

Na técnica *boosting* implementada, a cada exemplo é atribuído um peso. Inicialmente, todos os exemplos recebem um peso $w_i = 1/n$, sendo n o número de exemplos do conjunto de treinamento. Esses pesos são atualizados após a geração de cada regressor, sendo diminuídos os pesos dos exemplos preditos de maneira correta.

Dessa maneira, a cada iteração e atualização dos pesos dos exemplos, um novo conjunto de treinamento é formado, sendo que a probabilidade de determinado exemplo ser selecionado para esse novo conjunto é proporcional ao seu peso. Os exemplos que não foram preditos corretamente pelo regressor têm maior probabilidade de pertencer ao novo conjunto. Essa seleção foi realizada utilizando o método da roleta. De acordo com esse método, um exemplo irá ocupar, em uma roleta, uma área proporcional ao seu peso. Assim, exemplos com pesos maiores ocuparão áreas maiores e, consequentemente, terão maior probabilidade de serem selecionados. Mais detalhes sobre o método da roleta podem ser encontrados em (Carvalho, Braga, & Ludermir, 2003). A implementação do método da roleta, aqui utilizado, foi realizada por meio de uma reta numérica. Cada exemplo possui um intervalo nessa reta, com extensão igual ao seu peso. Para selecionar um exemplo, é sorteado, de maneira aleatória, um número pertencente a essa reta. O exemplo selecionado será aquele cujo intervalo possui o número sorteado.

Por fim, após determinado número de regressores serem gerados, estes são combinados de maneira semelhante à combinação realizada no *bagging*, descrita no Algoritmo 6. Porém, as iterações para gerar novos regressores a serem combinados é interrompida caso o erro do último regressor atinja um valor especificado.

Assim, a combinação utilizando *boosting*, bem como o *bagging*, podem ser divididos em três fases: geração de amostras, geração de regressores e combinação de regressores. Porém, no caso do *boosting*, as amostras, bem como os regressores, são gerados em série. A cada iteração, uma nova amostra (ou conjunto de treinamento) e, consequentemente, um novo regressor são gerados, de acordo com o desempenho do regressor anterior.

Os primeiros passos para a realização dos experimentos com a técnica *boosting* são a atribuição de pesos citada anteriormente e a geração de um regressor a partir de todos os dados do conjunto de treinamento original. Tendo esse primeiro regressor, a geração de amostras é realizada como descrito no Algoritmo 7.

O *script* para geração de amostras recebe como entrada três arquivos (*.data*, *.names* e *.rules*) que devem estar na sintaxe padrão de exemplos (Batista, 2002) e de regras de regressão do DISCOVER (Pugliesi, Dosualdo, & Rezende, 2003), apresentadas no Capítulo 4. Vale ressaltar que o arquivo *.data* contém os exemplos de treinamento, o *.names*

Algorithm 7 Algoritmo para Geração de Amostras — *Boosting*

Entrada:

nome dos arquivos de entrada {arquivos .data, .names e .rules}

```
1: Lê os arquivos de entrada
2: Verifica o tamanho do conjunto de treinamento e o atribui a variável numExemplos
3: for i := 1 a n do
4:   Verifica se o exemplo foi predito corretamente pelo conjunto de regras
5:   {Calcula o erro do regressor}
6:   if exemplo foi predito erroneamente then
7:     erro := erro + wi
8:   end if
9: end for
10:  $\beta := \text{erro}/(1 - \text{erro})$ 
11: {Atualiza os pesos}
12: for i := 1 a n do
13:   if exemplo foi predito corretamente then
14:     wi := wi *  $\beta$ 
15:   end if
16: end for
17: Normaliza os pesos dos exemplos
18: Seleciona os exemplos para o novo conjunto de treinamento, utilizando os pesos como
    probabilidade (método da roleta)
```

Saída: Novo conjunto de treinamento

possui as descrições dos atributos e no arquivo `.rules` encontram-se o regressor gerado convertido para a sintaxe padrão de regras de regressão. Com esses arquivos, verifica-se quais exemplos foram preditos de forma incorreta e quais foram preditos de forma correta pelo regressor, atualiza os pesos dos exemplos e seleciona os exemplos do novo conjunto de treinamento com base nesses pesos.

Com o novo conjunto de treinamento formado é iniciada uma nova iteração. A partir desse conjunto será gerado um regressor e, de acordo com seu desempenho, um novo conjunto de treinamento, e assim sucessivamente. Essa fase de geração de amostras e regressores é finalizada após o número de iterações requeridas pelo usuário ser atingido ou um regressor obter um erro maior que 0,5 (critério de parada que indica que o regressor atual é semelhante ao anterior) ou igual a zero. Em seguida, é realizada a combinação dos regressores.

A implementação do *script* para combinação dos regressores para a técnica *boosting* é similar à combinação utilizando a técnica *bagging*. A resposta do regressor combinado é dada pela média das respostas dos regressores gerados. O *script* utilizado nesta etapa recebe como entrada o número de regressores que devem ser combinados e os arquivos contendo as previsões geradas (Pugliesi, Sinoara, & Rezende, 2003).

6.5 Avaliação Experimental

Nesta seção são apresentados os experimentos realizados com a combinação de regressores homogêneos por meio das técnicas *bagging* e *boosting*, bem como a combinação de regressores heterogêneos utilizando a técnica *stacking*.

Além da precisão, avaliar a comprehensibilidade do conhecimento extraído é importante especialmente para viabilizar um melhor entendimento do modelo. Porém, quando se utiliza combinação de regressores normalmente ganha-se em precisão e perde-se em comprehensibilidade.

Para a realização dos experimentos com combinação de regressores foram utilizados algoritmos disponíveis nas ferramentas Cubist e Weka, descritas na Seção 4.1.3, as implementações do *bagging* e *boosting*, descritas na Seção 6.4, e três bases de dados (Abalone, CPU Performance e Housing) apresentadas na Seção 4.1.3.

6.5.1 Descrição dos Experimentos

Nesses experimentos a combinação de regressores homogêneos foi realizada de acordo com os seguintes passos:

1. Geração de Amostras: foram geradas amostras a partir dos conjuntos de treinamento, formando grupos compostos por 3, 5, 7 e 10 amostras para cada base de dados.
2. Geração de Regressores: para cada amostra foi gerado um regressor utilizando um algoritmo de regressão.
3. Combinação de Regressores: os regressores gerados a partir de cada grupo de amostras foram combinados utilizando o conjunto de teste, obtendo assim, quatro regressores combinados, um para cada grupo de amostras do item 1.

Para facilitar a análise e comparação dos resultados são apresentados nas Figuras 6.8 e 6.9 esquemas sintetizando as combinações realizadas, nas quais podem ser observados os três passos descritos anteriormente.

A Figura 6.8 representa a técnica *bagging* para combinação de regressores, que gera os regressores em paralelo, sendo que Resposta é a saída final da combinação dos regressores. Para esses experimentos foram geradas amostras para combinar 3, 5, 7 e 10 regressores, e utilizadas, para todos os conjuntos de amostras, as ferramentas Cubist e Weka e a implementação do *bagging* descrita. Todo esse esquema é repetido três vezes, uma para cada base de dados utilizada: Abalone, CPU Performance e Housing.

Já na Figura 6.9 é mostrada a técnica *boosting*, a qual gera os regressores em série. Nessa figura é apresentada a geração de amostras e combinação de 3 regressores, tanto para a ferramenta Cubist quanto para a Weka. Para a combinação de 5, 7 e 10 regressores, que também foram utilizadas nos experimentos, a idéia é a mesma, aumentando-se o número de passos necessários. Esses experimentos também foram realizados para as mesmas três bases de dados utilizadas no *bagging*.

Para a combinação de regressores heterogêneos com a técnica *stacking*, foi utilizada a implementação disponível na ferramenta Weka, que já possui vários algoritmos para problemas de regressão, necessários para a combinação de regressores heterogêneos. Foi também utilizado a combinação de 3, 5, 7 e 10 regressores, porém, nesse caso, esses valores representam o número de regressores gerados por cada algoritmo de regressão de nível-0.

A seguir são apresentados os resultados obtidos nas combinações de regressores realizadas com as três bases de dados. Vale ressaltar que todos esses resultados foram obtidos utilizando-se a média final dos erros dos regressores.

6.5.2 Resultados

No geral, a técnica *bagging* foi que apresentou melhores resultados em todos os experimentos realizados. A técnica *boosting* não chegou a gerar resultados para algumas

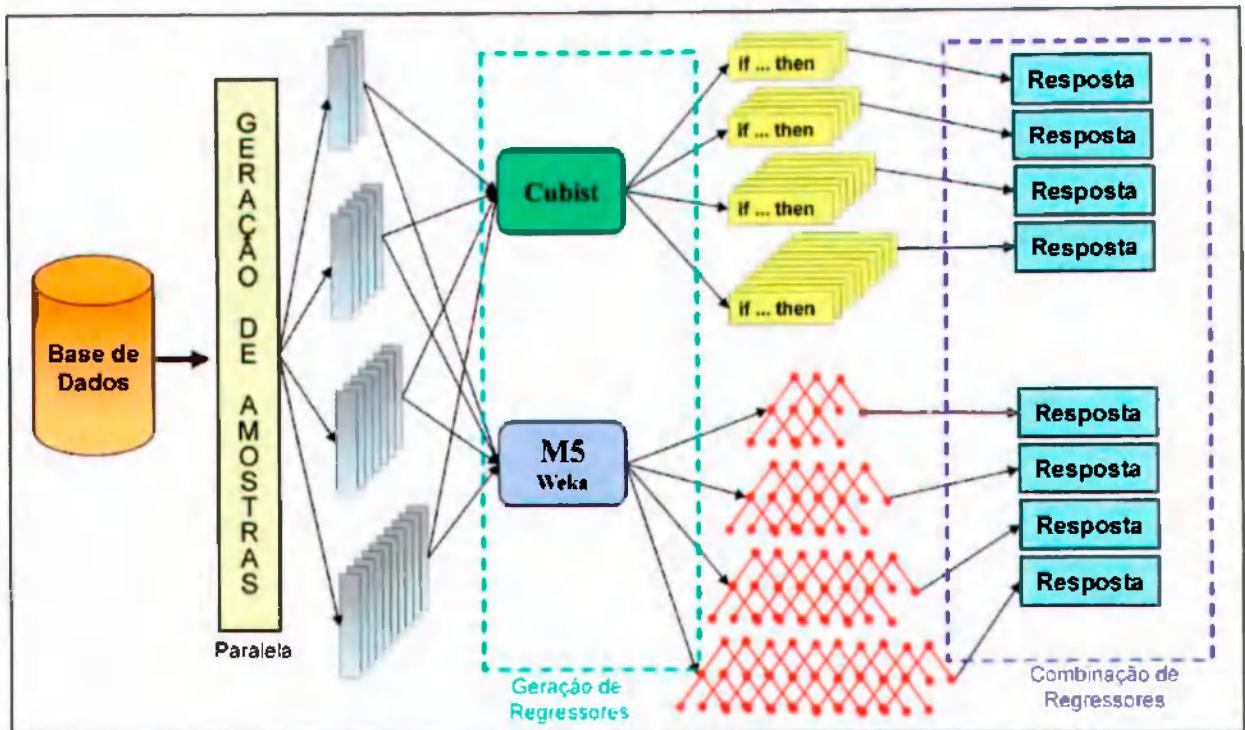


Figura 6.8: Esquema dos experimentos — *Bagging*.

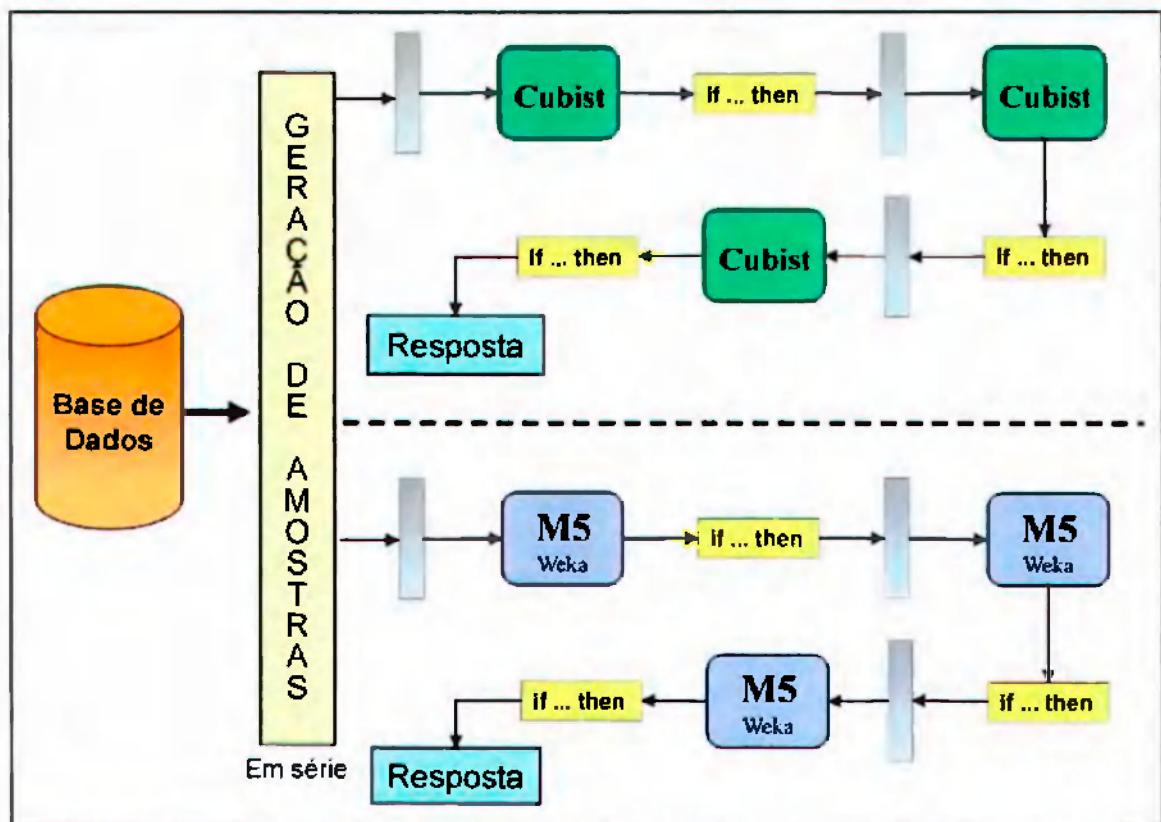


Figura 6.9: Esquema dos experimentos — *Boosting*.

das combinações dos experimentos realizados, pois o regressor obteve um erro maior que 0,5 ou igual a zero, antes de gerar o número total de combinações solicitadas.

Os resultados obtidos utilizando as técnicas *bagging* e *boosting* com a ferramenta Cubist, nas três bases de dados, são apresentados nas Tabelas 6.1, 6.2 e 6.3. Os valores apresentados representam a média final dos erros.

Tabela 6.1: Resultados dos experimentos com Cubist para a base de dados Abalone.

	Regressor único	1,5
	<i>Bagging</i>	<i>Boosting</i>
3 regressores	1,5017	1,9714
5 regressores	1,4918	2,3156
7 regressores	1,4918	—
10 regressores	1,4710	—

Tabela 6.2: Resultados dos experimentos com Cubist para a base de dados CPU Performance.

	Regressor único	57,3
	<i>Bagging</i>	<i>Boosting</i>
3 regressores	53,4256	54,4737
5 regressores	44,7892	56,8211
7 regressores	53,3055	—
10 regressores	46,7817	—

Tabela 6.3: Resultados dos experimentos com Cubist para a base de dados Housing.

	Regressor único	8,97
	<i>Bagging</i>	<i>Boosting</i>
3 regressores	3,9078	6,6143
5 regressores	4,9352	5,6064
7 regressores	5,9040	4,7097
10 regressores	4,7323	—

Já os resultados obtidos utilizando as técnicas *bagging*, *boosting* e *stacking* e a ferramenta Weka, nas três bases de dados, são apresentados nas Tabelas 6.4, 6.5 e 6.6.

Na Tabela 6.1 o melhor resultado da combinação é com a técnica *bagging* usando 10 regressores, e todas as combinações usando a técnica *boosting* não melhoraram a precisão para a base de dados Abalone comparado com o resultado obtido a partir de um único regressor. Pode-se notar que a técnica *boosting* não gerou resultados para a combinação de 7 e 10 regressores, pois o erro do regressor 5 foi maior do que 0,5 e a combinação parou.

Tabela 6.4: Resultados dos experimentos com Weka para a base de dados Abalone.

	Regressor único	1,525	
	<i>Bagging</i>	<i>Boosting</i>	<i>Stacking</i>
3 regressores	1,5126	1,9158	1,6526
5 regressores	1,4952	—	1,6591
7 regressores	1,5863	—	1,6542
10 regressores	1,5540	—	1,6492

Tabela 6.5: Resultados dos experimentos com Weka para a base de dados CPU Performance.

	Regressor único	58,2753	
	<i>Bagging</i>	<i>Boosting</i>	<i>Stacking</i>
3 regressores	44,4937	45,6213	52,2104
5 regressores	42,9138	—	52,1741
7 regressores	44,2828	—	55,1591
10 regressores	44,9673	—	52,0249

Um comportamento semelhante pode ser verificado para a Tabela 6.4, na qual foi utilizada a ferramenta Weka. Neste caso, o melhor resultado também foi o da combinação utilizando a técnica *bagging*, porém com 5 regressores. Tanto o *boosting* quanto o *stacking* não obtiveram resultados melhores que o regressor único. O *boosting* não gerou resultados para as combinações com 5, 7 e 10 regressores, inviabilizando a geração dos resultados finais.

Todos os resultados para as duas combinações de regressores usando a base de dados CPU Performance, apresentados na Tabela 6.2, são melhores que o obtido com um único regressor, ou seja, obtiveram menores erros. O melhor resultado para esse experimento é o com 5 regressores utilizando a técnica *bagging*. Também não há resultados para a combinação de 7 e 10 regressores para a técnica *boosting*.

Utilizando a ferramenta Weka, Tabela 6.5, todos os resultados também são melhores

Tabela 6.6: Resultados dos experimentos com Weka para a base de dados Housing.

	Regressor único	4,5816	
	<i>Bagging</i>	<i>Boosting</i>	<i>Stacking</i>
3 regressores	6,6656	4,5456	6,0024
5 regressores	5,1377	—	6,0588
7 regressores	3,9145	—	6,0715
10 regressores	3,8390	—	6,1839

que o obtido a partir de um único regressor. Assim como com a ferramenta Cubist, o melhor deles é o que utiliza a técnica *bagging* com 5 regressores. Neste caso, a técnica *boosting* não gerou resultados para 5, 7 e 10 regressores.

Na Tabela 6.3 são apresentados os resultados da base de dados Housing. O melhor resultado foi também obtido utilizando a técnica *bagging*. Além disso, os resultados de todas as combinações, tanto do *bagging* quanto do *boosting* foram melhores que o regressor único para todos os experimentos.

Já na Tabela 6.6 o melhor resultado foi obtido utilizando a técnica *bagging* com 10 regressores. A técnica *boosting* gerou resultado apenas para a combinação de 3 regressores, sendo esse um pouco menor que o regressor único. Já os resultados gerados pela técnica *stacking* foram todos piores que o regressor único.

Como dito anteriormente, a técnica *bagging* obteve melhores resultados em todos os experimentos realizados e a técnica *boosting* não gerou resultados para todas as combinações de regressores realizadas. Pode-se notar que a técnica *boosting* utilizando a estratégia por nós definida teve bons resultados, tendo um erro menor que o classificador único em todos os experimentos das bases de dados CPU Performance e Housing, podendo ser utilizado em outros experimentos. Os erros foram maiores apenas para a base de dados Abalone, mas essa base de dados também não obteve uma melhora significativa em nenhuma das combinações realizadas.

A combinação de regressores mostrou que, no geral, ganha-se em precisão, porém perde-se em comprehensibilidade, pois a compreensão pelo usuário de um único modelo é maior, e ao se combinar vários regressores não se tem mais um único regressor para ser analisado. Desse modo, não se sabe exatamente de onde vem os resultados, pois o mesmo é uma combinação dos regressores anteriores.

6.6 Considerações Finais

Sabe-se que o processo de Extração de Conhecimento de Bases de Dados não termina com a extração de padrões, pois após essa etapa o conhecimento deve ser avaliado, antes de ser utilizado. Nessa última etapa, denominada de pós-processamento, deve-se obter a precisão do conhecimento obtido a partir de determinado algoritmo, bem como avaliar o conhecimento quanto a sua qualidade e/ou utilidade para que, em caso positivo, seja utilizado para apoio a algum processo de tomada de decisão. Quando é identificado, na etapa de pós-processamento, que a precisão obtida não é boa, pode-se optar por fazer combinação de regressores. A abordagem de múltiplos regressores para análise de padrões vem tendo um expressivo crescimento e muitos casos de sucesso.

Como mencionado, em problemas de regressão, o cálculo da cobertura das regras é

complexo, pois o atributo meta é contínuo. Assim, necessita-se de um tratamento diferente para técnica *boosting*. Para isso foi utilizado a abordagem proposta, implementada e validada que utiliza um limiar para verificar a predição de novos exemplos em modelos de regressão (Pugliesi & Rezende, 2003; Pugliesi, Dosualdo, & Rezende, 2003). Como o *boosting* trabalha com pesos para identificar a probabilidade do exemplo ser selecionado e compor o novo conjunto de dados foi implementado o método da roleta que viabiliza a amostragem aleatória com peso. Com a utilização dessa abordagem, limiar e roleta, é viabilizado a combinação de regressores em série.

Neste capítulo são apresentados os experimentos realizados com a combinação de regressores utilizando três bases de dados retiradas do Repositório da UCI. Foram realizados experimentos combinando os regressores gerados pela ferramenta Cubist por meio de implementações das técnicas *bagging* e *boosting*. Outro conjunto de experimentos foi realizado com o auxílio da ferramenta Weka, que gerou e combinou regressores utilizando as técnicas *bagging* e *stacking*. Os regressores gerados por essa ferramenta também foram combinados utilizando a implementação proposta do *boosting*.

O *bagging* obteve melhores resultados que o *boosting* e o *stacking*. Porém, pode-se observar que, salvo algumas poucas exceções, as combinações de regressores utilizando as três técnicas obtiveram erros menores que a execução de um único regressor. Ressalta-se que a combinação de regressores apesar de, em geral, incrementar a precisão perde-se muito em comprehensibilidade.

Os resultados obtidos nos experimentos realizados para este capítulo confirmaram que a combinação de regressores pode aumentar a precisão do conhecimento extraído de bases de dados comparados a um único regressor. Entretanto, pode-se notar que não é possível determinar um padrão que mostre qual número de regressores é o melhor.

Integração de Regras de Regressão

Integração de conhecimento refere-se ao processo de incorporar novo conhecimento a um corpo de conhecimento já existente. Isso envolve determinar como o conhecimento novo e o existente interagem, e como o conhecimento existente deve ser modificado para acomodar novos conhecimentos. Integração de conhecimento difere significativamente de abordagens tradicionais para aprender a partir de instruções, pois nenhum desempenho específico da tarefa é assumido. Consequentemente, o sistema de aprendizado deve acessar o valor do novo conhecimento para determinar como o conhecimento existente deve ser modificado para acomodá-lo.

A integração de conhecimento aborda vários assuntos críticos que surgem quando desenvolve-se bases de conhecimento. Modificações que pretendiam corrigir um defeito na base de conhecimento podem entrar em conflito com o conhecimento já existente e introduzir problemas. Alternativamente, novos conhecimentos podem interagir sinergicamente com o conhecimento existente. Um tipo comum de interação benéfica ocorre quando o novo conhecimento explica o conhecimento existente. Reconhecer a interação entre o conhecimento novo e o anterior permite ao sistema explicar melhor suas conclusões (Murray, 1996).

Além disso, em domínios dinâmicos, o próprio conhecimento pode mudar com o passar do tempo. Obviamente, é importante saber o que mudou desde o aprendizado anterior. Um modelo computacional de integração de conhecimento, proposto em (Murray, 1996), inclui três atividades: (i) reconhecimento, que identifica o conhecimento relevante; (ii) elaboração, que aplica as regras relevantes para determinar a consequência do novo co-

nhecimento; e (iii) adaptação, que modifica a base de conhecimento para acomodar o novo conhecimento.

Em outras palavras, pode-se dizer que a integração de conhecimento envolve a construção de bases de conhecimento a partir de vários sistemas de aprendizado e posterior *match* do conhecimento. Um ponto importante é como fazer uso de BCs que já estão construídas (Brazdil & Torgo, 1990, 1991; Navathe & Donahoo, 1995).

Assim, o processo de integração de conhecimento consiste em juntar diferentes conhecimentos, gerados a partir de amostras diferentes da mesma base de dados, em uma única base de conhecimento, que seja mais completa que cada uma das bases individuais e mantenha a consistência. Essa base de conhecimento única também pode ser denominada de teoria integrada ou conjunto de regras integrado.

Além disso, no pós-processamento também se objetiva a uma filtragem do que foi aprendido, eliminando o conhecimento espúrio, sem valor ou de situações óbvias guiadas pelo senso comum. Essa seleção de regras também pode ser utilizada mesmo quando não houver integração de diferentes bases de conhecimento, pois com o aumento do tamanho das bases de dados mineradas, o volume de conhecimento produzido pelos algoritmos de aprendizado normalmente cresce, podendo inclusive sobrecarregar os usuários. Fornecer ao usuário uma grande quantidade de padrões descobertos a partir dos dados não é produtivo, uma vez que, normalmente, ele procura por uma pequena lista de padrões interessantes (em sua opinião).

Isso é especialmente verdade para descoberta de regras de regressão, na qual os conjuntos de regras não são facilmente comprehensíveis, e uma análise de regras pode ser muito útil. Visando a melhora da qualidade do conhecimento obtido utiliza-se a integração de conhecimento também para melhorar a precisão da predição de novos exemplos. Além disso, medidas de grau de interesse e comprehensibilidade também estão sendo utilizadas, a fim de se obter/escolher as regras mais apropriadas para um determinado problema, tendo em vista os objetivos do usuário final.

Para realizar essa avaliação, algumas medidas podem ser usadas em problemas de regressão, como as derivadas e as não derivadas da matriz de contingência, apresentadas no Capítulo 5.

Neste capítulo a integração de regras consistirá da junção de diferentes conjuntos de regras e posterior avaliação e seleção de regras, gerando, para cada integração, um único conjunto de regras.

O objetivo principal deste capítulo é explorar a integração de bases de conhecimento (conjuntos de regras), sendo que as bases geradas utilizam conjuntos de dados ou algoritmos diferentes. Para tanto, um método para avaliação do modelo e de regras foi definido. Para aumentar a comprehensibilidade do modelo, experimentos empíricos foram realizados

para eliminar as piores e selecionar as melhores regras de regressão. Este capítulo está organizado da seguinte maneira: na Seção 7.1 são apresentadas duas abordagens para integração de regras de regressão, sendo que uma delas visa a escolha das melhores regras e a outra a eliminação das piores regras. Essa segunda abordagem é denominada poda de regras. Na Seção 7.2 é apresentada a metodologia para integração de regras de regressão por nós proposta e implementada, sendo os experimentos realizados para validação dessa metodologia apresentados na Seção 7.3. Na Seção 7.4 são apresentados os experimentos realizados e os resultados obtidos utilizando a abordagem de seleção das melhores regras e construção iterativa do conjunto de regras. Finalmente, na Seção 7.5 são apresentadas as considerações finais deste capítulo.

7.1 Abordagens na Integração de Regras de Regressão

A necessidade de integração de conhecimento pode surgir, por exemplo, quando o conhecimento é adquirido por meio de diferentes fontes de conhecimento, como vários especialistas do domínio ou algoritmos de aprendizado. Como as opiniões dos especialistas podem diferir e os algoritmos de aprendizado possuem *bias* diferentes, as bases de conhecimento construídas também podem ser diferentes.

Assim, a integração visa construir um único preditor que explore todo o conhecimento disponível, possua um bom desempenho e explique melhor e de maneira sucinta os dados fornecidos.

Encontrar quais padrões são interessantes a um usuário, visando tanto o desempenho quanto a qualidade, é um problema difícil. Isso pode ser realizado por meio da análise individual das regras e seleção das que farão parte do conjunto de regras final. Essa seleção pode ser realizada por meio de duas abordagens. Uma delas é eliminar do conjunto de regras integrado as que não são interessantes (Sahar, 1999; Liu, Jsu, Ma, & Chen, 1999). Essa estratégia visa focar a atenção nas “partes fracas” do conjunto de regras, não gastando muito esforço em modificações desnecessárias. E a outra é selecionar as melhores regras e ir construindo o conjunto de regras integrado. Um dos critérios de parada pode ser, por exemplo, continuar a acrescentar regras até que o conjunto de regras integrado cubra todo o conjunto de dados (Brazdil & Torgo, 1990, 1991).

Tanto a seleção das piores quanto a das melhores regras de regressão pode ser realizada por meio das diferentes medidas de avaliação apresentadas no Capítulo 5. Essa seleção pode ser realizada ordenando-se as regras em ordem crescente ou decrescente, baseado em uma ou mais medidas de avaliação (Hilderman & Hamilton, 2001; Freitas, 1998a).

Vale ressaltar, que em alguns casos, o conhecimento descoberto a partir de uma base de dados pode, com a integração do conhecimento de uma nova base de dados, se tornar

inconsistente ou inválido. A noção de robustez pode ser definida como a probabilidade da base de dados estar em um estado consistente com o conhecimento descoberto. Essa probabilidade é diferente da precisão da predição, pois a precisão mede a probabilidade do conhecimento ser consistente com dados não vistos selecionados aleatoriamente ao invés de um completo estado da base de dados. Essa diferença é significante em bases de dados que são interpretadas usando a hipótese de mundo fechado (Hsu & Knoblock, 1996).

7.2 Metodologia para Integração de Regras de Regressão

A metodologia para Integração de Regras de Regressão (*IRR*) tem como objetivo gerar um conjunto final de regras de regressão por meio da junção de diferentes bases de conhecimento e posterior poda das piores regras.

Na Figura 7.1 é apresentado um esquema que ilustra a metodologia *IRR*.

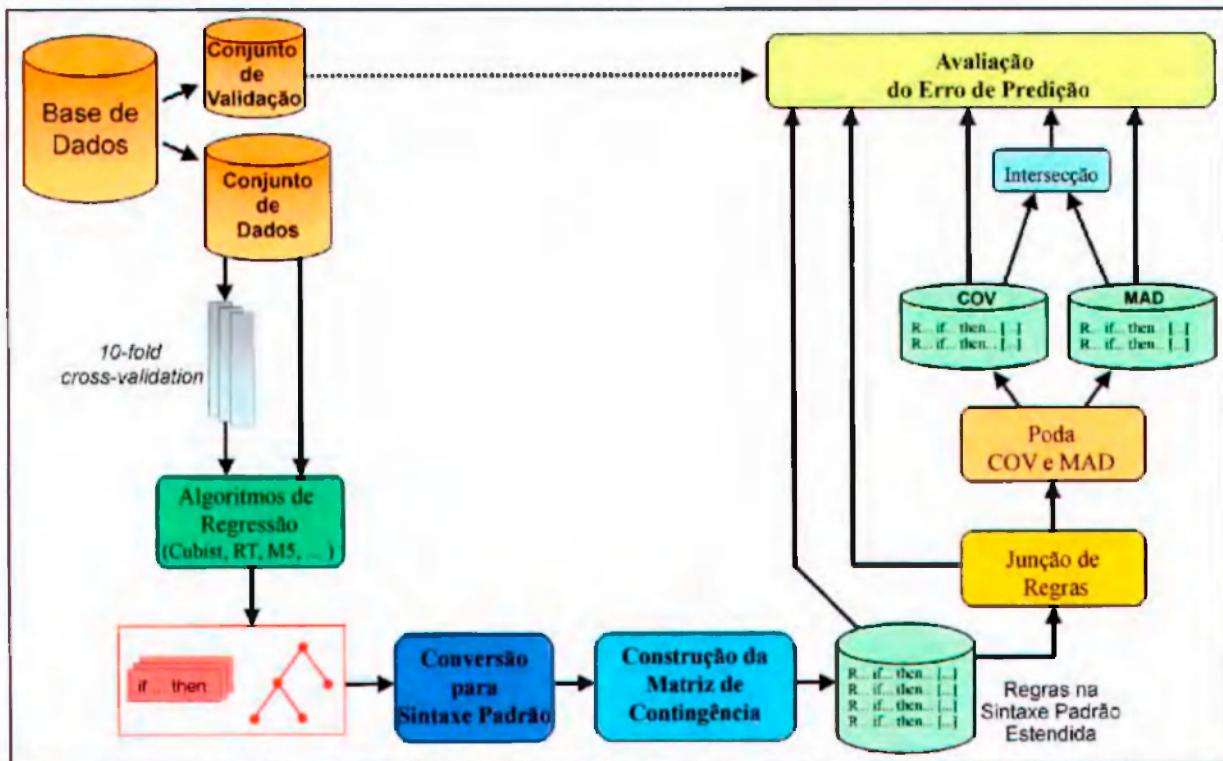


Figura 7.1: Esquema da metodologia para integração de regras de regressão.

Nos experimentos realizados, cada base de dados foi dividida em dois conjuntos: um com 75% dos exemplos usado para treinamento e teste, e outro com 25% dos exemplos que é o conjunto de validação utilizado para a avaliação dos conjuntos de regras. Deve ser ressaltado que o conjunto de validação não foi utilizado nem na fase de geração dos regressores e nem no cálculo das medidas relacionadas a cada regra de regressão.

Na metodologia *IIRR*, a integração de regras foi realizada de duas maneiras distintas, sendo os resultados de cada uma, a princípio, analisados em separado:

- utilizando *10-fold cross-validation* no conjunto de dados com 75% dos exemplos. As regras dos 10 conjuntos de regras foram reunidas em um único conjunto de regras integrado;
- utilizando o conjunto de dados com 75% dos exemplos para cada um dos três algoritmos de regressão (Cubist, RT e M5). As regras geradas pelos três algoritmos foram reunidas em um único conjunto de regras.

Além disso, um outro regressor foi gerado a partir de todo o conjunto de dados com 75% dos exemplos, para cada um dos três algoritmos, a fim de comparar seus resultados com os dos conjuntos de regras integrados.

Todo o processo de integração, baseado na metodologia *IIRR*, é detalhado a seguir:

1. separação em dois conjuntos, um com 75% dos exemplos e o outro com 25% restantes;
2. utilização de todo o conjunto de dados e também execução de *10-fold cross-validation*;
3. todas as amostras são submetidas aos três algoritmos de regressão, sendo posteriormente feita a predição do conjunto de regras gerados com 75% dos dados;
4. cada base de conhecimento é convertida para a sintaxe padrão de regras de regressão;
5. são calculadas as medidas e a matriz de contingência, gerando conjuntos de regras na sintaxe padrão estendida para regras de regressão definida na Seção 4.2;
6. junção de regras gerando dois diferentes conjuntos integrados de regras (utilizando *10-fold cross-validation* e três algoritmos integrados) e cálculo da predição para esse conjunto;
7. poda de regras, que elimina automaticamente 5%, 10%, 25% e 50% das piores regras. Essa eliminação é baseada em duas medidas descritas a seguir. Em seguida é avaliada a predição para o conjunto de regras resultante;
8. intersecção dos conjunto de regras gerados utilizando as medidas MAD e COV, ou seja, são selecionadas apenas as regras que aparecem em ambos os conjunto de regras.

Previvamente, foi definido um método para avaliar o erro de predição de um conjunto de regras qualquer, de maneira a viabilizar a comparação dos resultados obtidos com as

abordagens utilizadas. Quando mais de uma regra cobre o exemplo, é realizada uma média dos valores preditos para a predição final do exemplo. Porém, se o exemplo não é coberto por nenhuma regra, uma função de aproximação é utilizada. Essa função considera um peso médio baseado nas duas regras mais próximas ao exemplo, ou seja, as regras que “quase” cobrem o exemplo.

Para a realização da poda de regras, a eliminação das piores regras é baseada em duas medidas diferentes. Uma medida é a MAD (Equação (5.20), página 81), e a outra é a cobertura, que utiliza os valores da matriz de contingência para problemas de regressão (Seção 5.3) e calcula a porcentagem dos exemplos cobertos que são corretos (Equação (7.1)).

$$\text{COV}(R) = \frac{n(BH) - n(\bar{BH})}{n} \quad (7.1)$$

7.3 Avaliação Experimental Utilizando a Metodologia *IIRR*

7.3.1 Descrição dos Experimentos

De forma a avaliar a metodologia *IIRR* proposta, alguns experimentos foram realizados utilizando três algoritmos de regressão (Cubist, RT e *M5*) e sete bases de dados (Auto MPG, Housing, Abalone, Wisconsin Breast Cancer, Servo, MV e Delta Ailerons).

Ressalta-se que os algoritmos RT e *M5* geram árvores de regressão, ao invés de regras de regressão. Assim, essas árvores de regressão foram convertidas em regras de regressão na sintaxe padrão de regras de regressão.

Toda a metodologia *IIRR* descrita acima foi realizada para cada base de dados separadamente.

7.3.2 Resultados

Nas Tabelas 7.1 à 7.21 são apresentados os erros dos diferentes conjuntos de regras. Essa avaliação é feita usando o método proposto anteriormente. A coluna Único é o erro do regressor único obtido com 75% da base de dados. A coluna #R representa o número de regras da coluna anterior.

A coluna 0% apresenta o erro do conjunto de regras integrado, ou seja, todas as regras geradas pelo *10-fold cross-validation* ou pelos três algoritmos. As colunas 5%, 10%, 25% e 50% são os erros do conjunto de regras com a eliminação de 5%, 10%, 25% e 50% das piores regras, respectivamente. Todos os erros são calculados no conjunto de validação,

que representa 25% da base de dados.

Para cada uma das bases de dados existem três tabelas e uma figura com quatro gráficos. A primeira tabela representa os resultados utilizando-se *10-fold cross-validation*. A segunda mostra os resultados originados da integração dos três algoritmos de regressão. Para ambas as tabelas são apresentados tanto os resultados da poda utilizando a medida de cobertura COV quanto a MAD. Já na terceira tabela são apresentados os resultados obtidos por meio da intersecção das regras geradas com poda utilizando as duas medidas. As três primeiras linhas representam cada algoritmo separadamente (referentes à primeira tabela) e a última linha mostra os resultados da intersecção quando se utilizou os três algoritmos (referente à segunda tabela).

Nos gráficos (Figuras 7.2 à 7.8) são apresentados os resultados obtidos de cada algoritmo e com a junção das regras geradas pelos três algoritmos. Em cada gráfico tem-se uma linha para a medida COV, uma para a MAD e uma terceira para a intersecção (denotada por INTER na legenda do gráfico).

Nas Tabelas 7.1, 7.2 e 7.3 são apresentados todos os resultados da base de dados Auto-MPG, ou seja, o erro e o número de regras para cada caso. Pode-se notar que o erro diminui quando todas as regras são integradas. Com a eliminação de regras, tem-se ainda em alguns casos um ganho em precisão. Isso pode ser explicado pelo fato de que ao predizer novos exemplos quando existe um grande número de regras, a chance do exemplo não ser coberto por nenhuma regra é pequeno, ou seja, maior chance de usar a média de diferentes regras que cobrem o exemplo, e menor chance de usar a função de proximidade para exemplos não cobertos.

Entretanto, usando a medida MAD como métrica para excluir as piores regras, o erro aumenta consideravelmente. Esse aumento é de aproximadamente 70% para o RT e 120% para o M5 quando 50% das regras são eliminadas. A tendência é um pouco diferente quando a medida de cobertura é utilizada como métrica para eliminar as piores regras. Em quase todos os casos o erro diminui ou permanece próximo do erro da coluna 0%. Apenas em alguns casos o erro aumenta um pouco quando mais regras são eliminadas. Para a integração utilizando os três diferentes algoritmos, os erros mantiveram a mesma tendência, sendo melhores os resultados para a medida COV.

A outra base de dados que obteve comportamento semelhante é a MV, que também teve considerável aumento do erro em alguns casos, porém os piores resultados foram com a medida COV. Seus resultados podem ser vistos nas Tabelas 7.16, 7.17 e 7.18.

A maioria dos erros para a base de dados Housing, mostrados nas Tabelas 7.4, 7.5 e 7.6 são melhores que o conjunto de regras único. Nessa base de dados também o conjunto de regras integrado apresenta erro menor que no conjunto de regras único. Apesar dos erros terem sido grandes quando se integra as regras geradas pelos três algoritmos, o menor erro

dentre todos os experimentos foi utilizando essa integração com a medida MAD e podando 25% das regras. Os resultados obtidos com a intersecção manteve o mesmo comportamento da integração com *10-fold cross-validation*.

Nas Tabelas 7.7, 7.8 e 7.9 são mostrados os erros para a base de dados Abalone. Os métodos de eliminação de regras tiveram erros semelhantes ao conjunto de regras integrado. Observando as colunas Único e 0% percebe-se que o erro diminuiu nos três algoritmos utilizados. Essa redução foi aparentemente pequena: com o Cubist, o ganho foi de 3,33% comparado com o regressor único; com o RT, foi de 3,75% e com o *M5*, foi de 6,25%. Entretanto, o número de regras na integração sem poda é drasticamente maior quando comparado com o número de regras do regressor único.

Nas Tabelas 7.10, 7.11 e 7.12 são apresentados os resultados obtidos para a base de dados Wisconsin Breast Cancer. Apesar dos valores dos erros da predição serem semelhantes, ressalta-se que o menor erro foi o que utilizou a intersecção com poda de 50% para o algoritmo *M5*, sendo o número de regras um dos menores entre todas as integrações.

Os resultados da base de dados Servo, mostrados nas Tabelas 7.13, 7.14 e 7.15 seguem os mesmos padrões apresentados nas bases anteriores. Nos resultados obtidos com a intersecção observa-se um grande aumento na taxa de erro para as maiores porcentagens de poda.

Nas Tabelas 7.19, 7.20 e 7.21 são mostrados os resultados da base de dados Delta Ailerons, sendo que na maioria dos casos a variação do erro é pequena quando se analisa as várias integrações para um mesmo algoritmo, ou seja, analisando os resultados a cada linha. Porém, pode-se observar alguns bons resultados na integração e intersecção, como os apresentados na intersecção de três algoritmos.

Analizando os resultados de todas as bases de dados pode-se perceber que na maioria das integrações existe um ganho em precisão, porém, quase sempre, com um aumento no número de regras. Isto é, ganha-se em precisão e perde-se em comprehensibilidade, porém não em interpretabilidade.

Pode-se observar também que, para os resultados da intersecção utilizando os três algoritmos, em praticamente todas as bases de dados, houve um ganho em precisão sem um grande aumento no número de regras, sendo que em alguns casos houve até redução no número médio de regras.

Auto-MPG

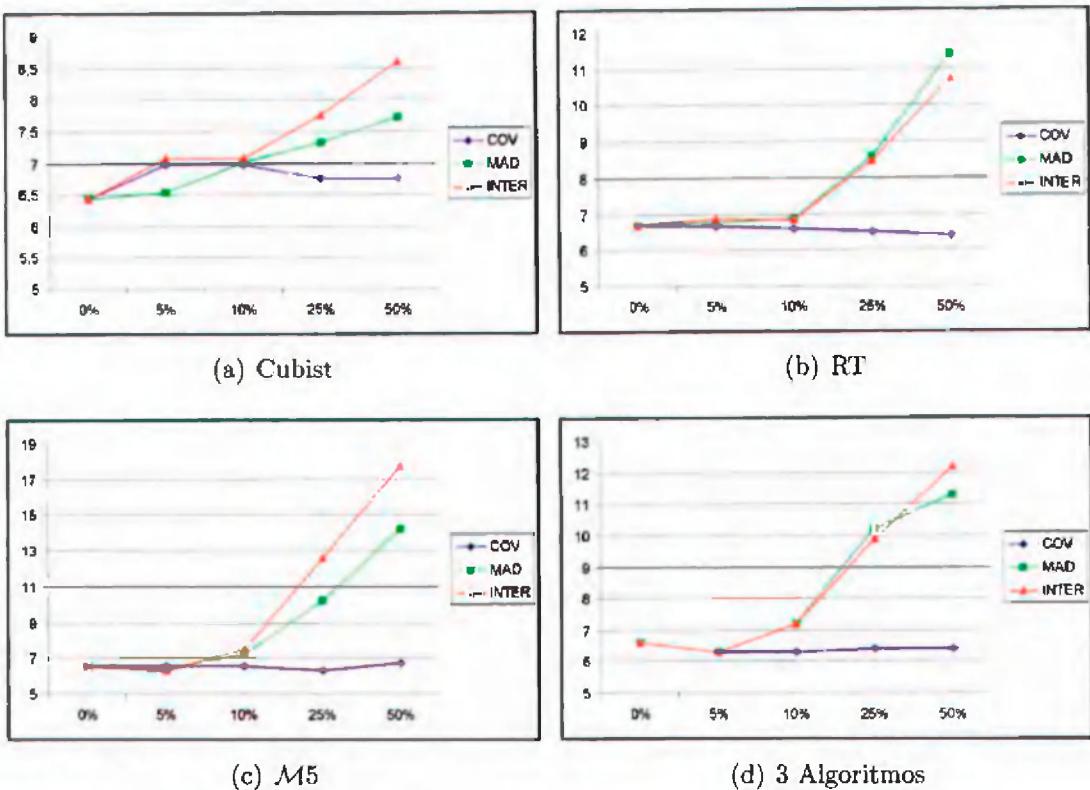


Figura 7.2: Gráficos da base de dados Auto-MPG.

Tabela 7.1: Resultados para a base de dados Auto-MPG com *10-fold cross-validation*.

	Único	#R	0%	#R	COV					MAD					
					5%	10%	25%	50%	5%	10%	25%	50%	5%	10%	
Cubist	8,10	9	6,44	93	6,97	6,97	6,75	6,75	6,54	7,01	7,33	7,73			
RT	6,92	4	6,68	58	6,64	6,60	6,50	6,37	6,77	6,87	8,59	11,39			
M5	6,50	10	6,50	114	6,50	6,50	6,30	6,70	6,40	7,20	10,20	14,19			

Tabela 7.2: Resultados para a base de dados Auto-MPG integrando os 3 algoritmos.

	COV					MAD				
	0%	#R	5%	10%	25%	50%	5%	10%	25%	50%
	6,60	23	6,30	6,30	6,40	6,40	6,30	7,20	10,20	11,30

Tabela 7.3: Resultados da intersecção para a base de dados Auto-MPG.

	0%	#R	5%	#R	10%	#R	25%	#R	50%	#R
Cubist	6,44	93	7,07	38	7,07	36	7,76	26	8,59	11
RT	6,68	58	6,85	24	6,82	23	8,47	18	10,73	7
M5	6,50	114	6,30	14	7,50	13	12,60	10	17,70	5
3 Algoritmos	6,60	23	6,30	19	7,20	17	9,90	12	12,20	4

Housing

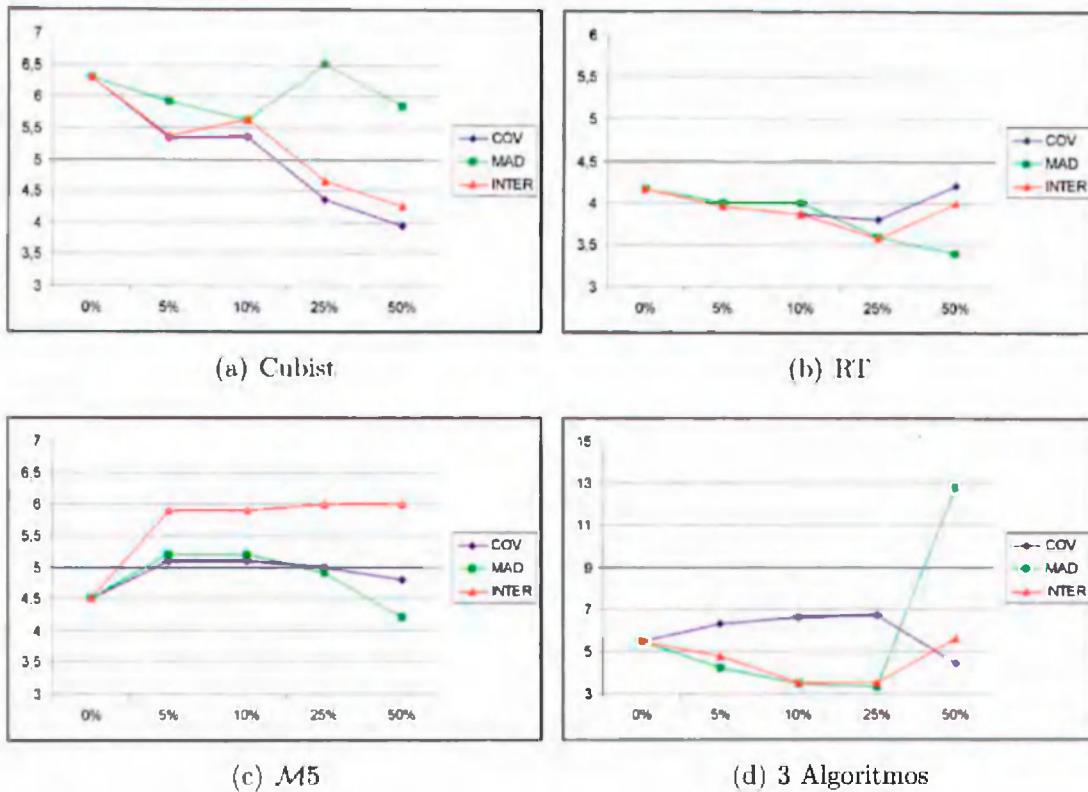


Figura 7.3: Gráficos da base de dados Housing.

Tabela 7.4: Resultados para a base de dados Housing com *10-fold cross-validation*.

	Unico	#R	COV					MAD				
			0%	#R	5%	10%	25%	50%	5%	10%	25%	50%
Cubist	9,06	10	6,30	89	5,34	5,35	4,37	3,94	5,91	5,62	6,51	5,84
RT	4,06	21	4,17	196	3,96	3,86	3,80	4,20	3,99	3,99	3,59	3,39
M5	5,20	22	4,50	186	5,10	5,10	5,00	4,80	5,20	5,20	4,90	4,20

Tabela 7.5: Resultados para a base de dados Housing integrando os 3 algoritmos.

0%	#R	COV					MAD				
		5%	10%	25%	50%	5%	10%	25%	50%		
5,47	53	6,30	6,60	6,70	4,40	4,18	3,46	3,30	12,80		

Tabela 7.6: Resultados da intersecção para a base de dados Housing.

	0%	#R	5%	#R	10%	#R	25%	#R	50%	#R
Cubist	6,30	89	5,37	40	5,63	37	4,66	26	4,25	15
RT	4,17	196	3,96	26	3,86	25	3,57	17	4,00	7
M5	4,50	186	5,90	22	5,90	20	6,00	14	6,00	6
3 Algoritmos	5,47	53	4,80	34	3,50	30	3,50	20	5,60	10

Abalone

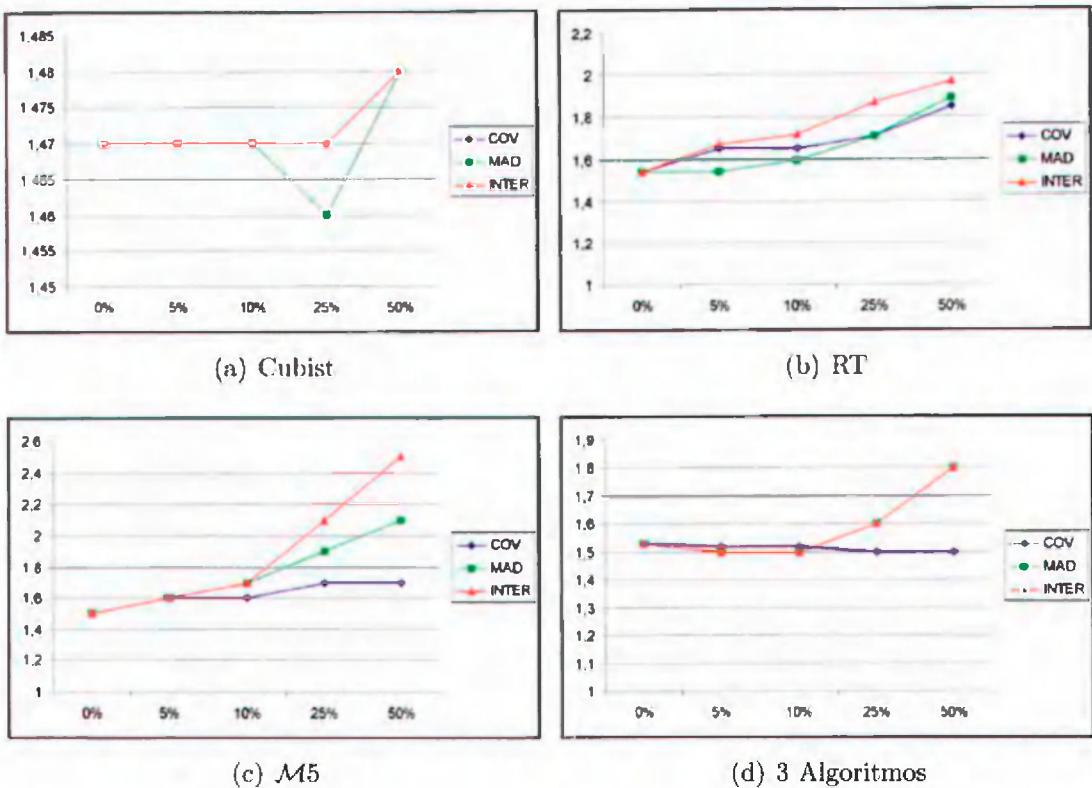


Figura 7.4: Gráficos da base de dados Abalone.

Tabela 7.7: Resultados para a base de dados Abalone com *10-fold cross-validation*.

	Unico	#R	0%	#R	COV				MAD			
					5%	10%	25%	50%	5%	10%	25%	50%
Cubist	1,50	13	1,47	111	1,47	1,47	1,47	1,48	1,47	1,47	1,46	1,48
RT	1,60	48	1,54	737	1,65	1,65	1,71	1,85	1,54	1,59	1,71	1,89
M5	1,60	47	1,50	357	1,60	1,60	1,70	1,70	1,60	1,70	1,90	2,10

Tabela 7.8: Resultados para a base de dados Abalone integrando os 3 algoritmos.

0%	#R	COV				MAD			
		5%	10%	25%	50%	5%	10%	25%	50%
1,53	108	1,52	1,52	1,50	1,50	1,50	1,50	1,60	1,80

Tabela 7.9: Resultados da intersecção para a base de dados Abalone.

	0%	#R	5%	#R	10%	#R	25%	#R	50%	#R
Cubist	1,47	111	1,47	75	1,47	68	1,47	57	1,48	25
RT	1,54	737	1,67	86	1,72	79	1,87	51	1,97	29
M5	1,50	357	1,60	66	1,70	59	2,10	36	2,50	20
3 Algoritmos	1,53	108	1,50	81	1,50	71	1,60	48	1,80	23

Wisconsin Breast Cancer

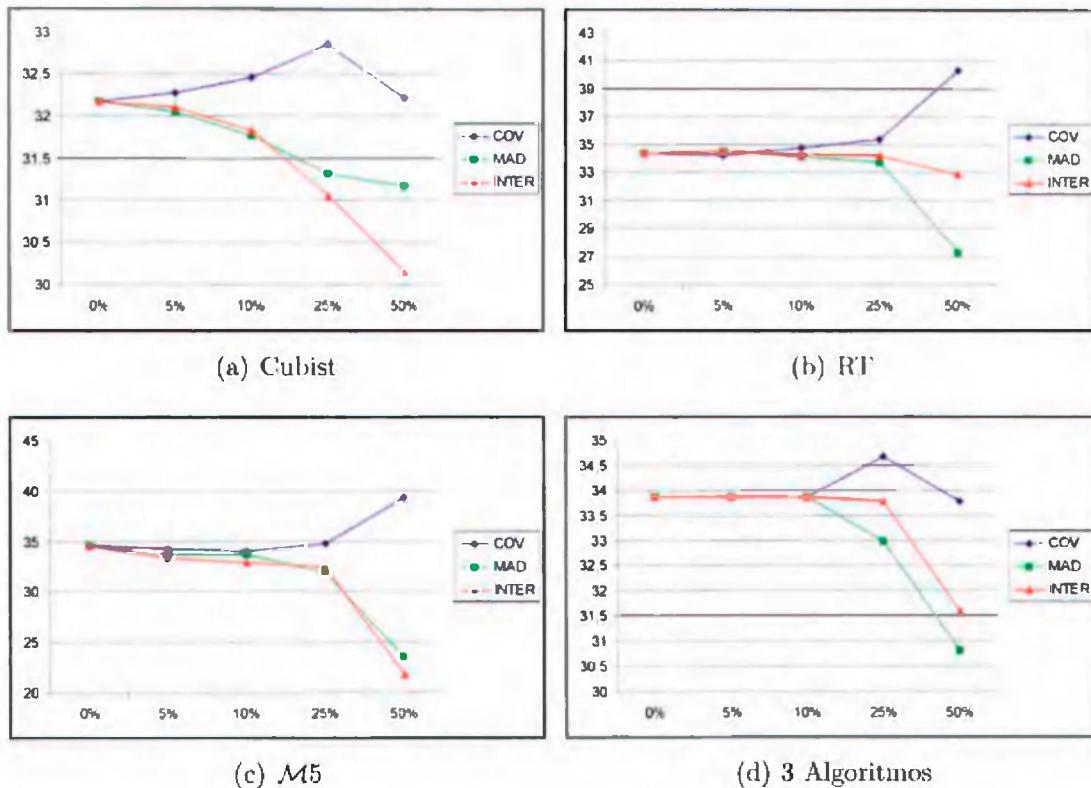


Figura 7.5: Gráficos da base de dados Wisconsin Breast Cancer.

Tabela 7.10: Resultados para a base de dados Wisconsin Breast Cancer com *10-fold cross-validation*.

	Único	#R	COV				MAD			
			0%	#R	5%	10%	25%	50%	5%	10%
Cubist	32,34	2	32,18	14	32,28	32,46	32,85	32,22	32,06	31,77
RT	34,02	2	34,33	26	34,22	34,76	35,36	40,29	34,50	34,14
M5	36,80	1	34,50	22	34,20	34,00	34,70	39,30	33,70	33,60

Tabela 7.11: Resultados para a base de dados Wisconsin Breast Cancer integrando os 3 algoritmos.

0%	#R	COV				MAD			
		5%	10%	25%	50%	5%	10%	25%	50%
33,86	5	33,86	33,86	34,67	33,78	33,86	33,86	32,97	30,80

Tabela 7.12: Resultados da intersecção para a base de dados Wisconsin Breast Canc

	0%	#R	5%	#R	10%	#R	25%	#R	50%	#R
Cubist	32,18	14	32,11	11	31,84	9	31,05	6	30,14	5
RT	34,33	26	34,54	17	34,24	15	34,16	11	32,87	5
M5	34,50	22	33,35	15	32,90	13	32,40	8	21,80	2
3 Algoritmos	33,86	5	33,86	5	33,86	5	33,78	3	31,60	2

Servo

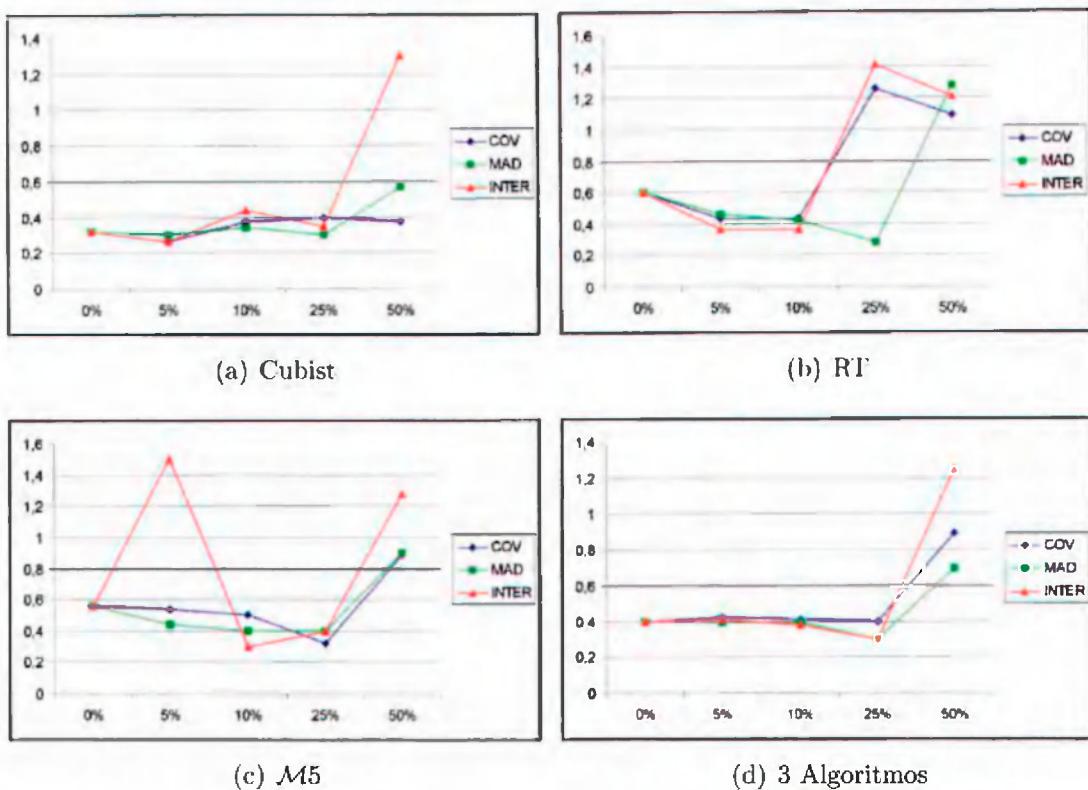


Figura 7.6: Gráficos da base de dados Servo.

Tabela 7.13: Resultados para a base de dados Servo com *10-fold cross-validation*.

	Unico	#R	0%	#R	COV					MAD					
					5%	10%	25%	50%	5%	10%	25%	50%	5%	10%	
Cubist	0,37	8	0,32	79	0,26	0,37	0,40	0,37	0,30	0,34	0,30	0,57			
RT	0,68	3	0,60	28	0,43	0,43	1,26	1,09	0,46	0,42	0,28	1,28			
M5	0,40	9	0,56	69	0,54	0,50	0,32	0,89	0,44	0,40	0,40	0,90			

Tabela 7.14: Resultados para a base de dados Servo integrando os 3 algoritmos.

0%	#R	COV					MAD				
		5%	10%	25%	50%	5%	10%	25%	50%	5%	10%
0,40	20	0,42	0,41	0,40	0,89	0,39	0,39	0,30	0,70		

Tabela 7.15: Resultados da intersecção para a base de dados Servo.

	0%	#R	5%	#R	10%	#R	25%	#R	50%	#R
Cubist	0,32	79	0,26	21	0,44	18	0,35	12	1,30	2
RT	0,60	28	0,36	7	0,36	7	1,41	3	1,21	1
M5	0,56	69	1,50	24	0,30	23	0,40	16	1,28	8
3 Algoritmos	0,40	20	0,41	13	0,38	11	0,30	8	1,25	3

Servo

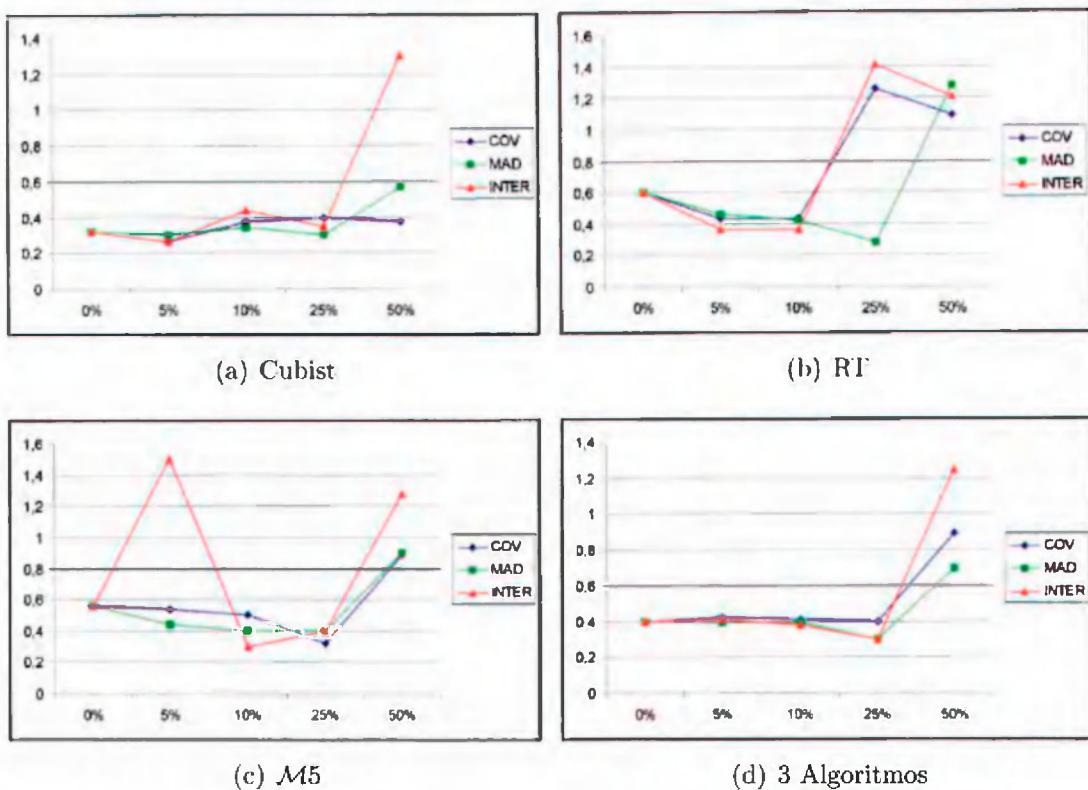


Figura 7.6: Gráficos da base de dados Servo.

Tabela 7.13: Resultados para a base de dados Servo com *10-fold cross-validation*.

	Unico	#R	0%	#R	COV					MAD					
					5%	10%	25%	50%	5%	10%	25%	50%	5%	10%	
Cubist	0,37	8	0,32	79	0,26	0,37	0,40	0,37	0,30	0,34	0,30	0,57			
RT	0,68	3	0,60	28	0,43	0,43	1,26	1,09	0,46	0,42	0,28	1,28			
M5	0,40	9	0,56	69	0,54	0,50	0,32	0,89	0,44	0,40	0,40	0,90			

Tabela 7.14: Resultados para a base de dados Servo integrando os 3 algoritmos.

0%	#R	COV					MAD				
		5%	10%	25%	50%	5%	10%	25%	50%	5%	10%
0,40	20	0,42	0,41	0,40	0,89	0,39	0,39	0,30	0,70		

Tabela 7.15: Resultados da intersecção para a base de dados Servo.

	0%	#R	5%	#R	10%	#R	25%	#R	50%	#R
Cubist	0,32	79	0,26	21	0,44	18	0,35	12	1,30	2
RT	0,60	28	0,36	7	0,36	7	1,41	3	1,21	1
M5	0,56	69	1,50	24	0,30	23	0,40	16	1,28	8
3 Algoritmos	0,40	20	0,41	13	0,38	11	0,30	8	1,25	3

Delta Ailerons

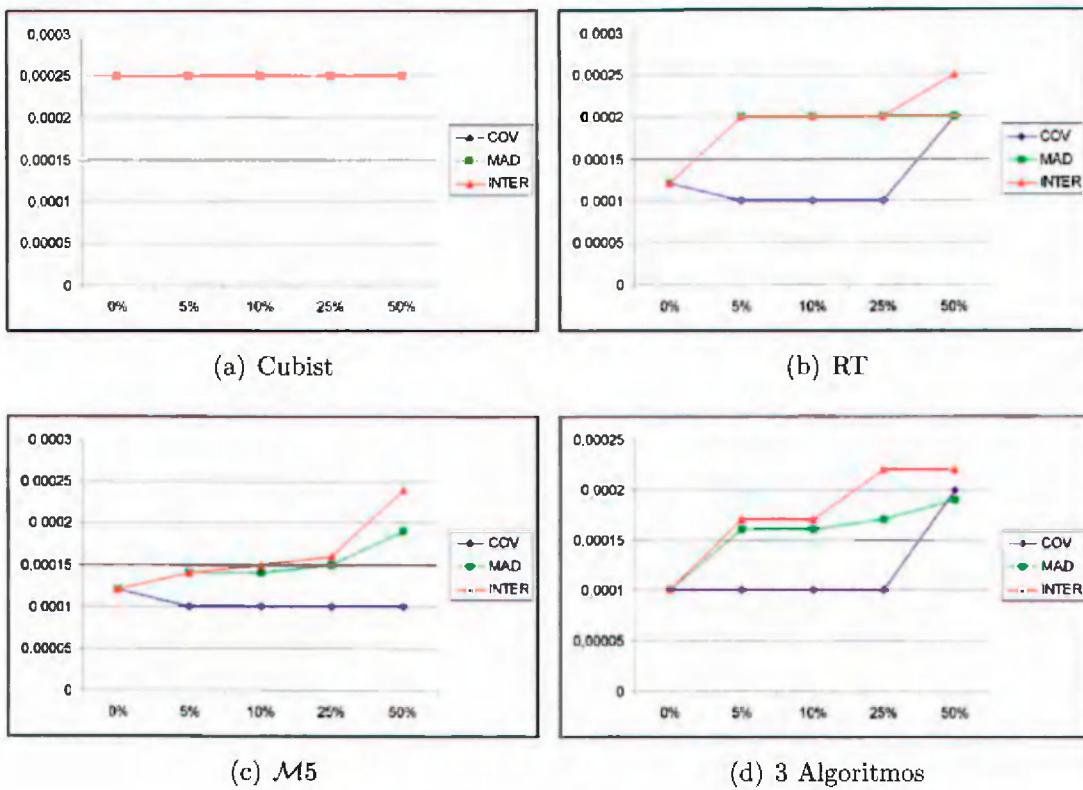


Figura 7.8: Gráficos da base de dados Delta Ailerons.

Tabela 7.19: Resultados para a base de dados Delta Ailerons com *10-fold cross-validation*.

	Único	#R	COV				MAD				
			0%	5%	10%	25%	50%	5%	10%	25%	50%
Cubist	0,00025	1	0,00025	0,00025	0,00025	0,00025	0,00025	0,00025	0,00025	0,00025	0,00025
RT	0,00010	175	0,00012	1274	0,00010	0,00010	0,00010	0,00020	0,00020	0,00020	0,00020
M5	0,00012	58	0,00012	534	0,00010	0,00010	0,00010	0,00010	0,00014	0,00014	0,00015

Tabela 7.20: Resultados para a base de dados Delta Ailerons integrando os 3 algoritmos.

0%	#R	COV				MAD			
		5%	10%	25%	50%	5%	10%	25%	50%
0,00010	234	0,00010	0,00010	0,00010	0,00020	0,00016	0,00016	0,00017	0,00019

Tabela 7.21: Resultados da intersecção para a base de dados Delta Ailerons.

	0%	#R	5%	#R	10%	#R	25%	#R	50%	#R
Cubist	0,00025	10	0,00025	1	0,00025	1	0,00025	1	0,00025	1
RT	0,00012	1274	0,00020	20	0,00020	20	0,00020	20	0,00020	12
M5	0,00012	534	0,00014	21	0,00015	19	0,00016	15	0,00024	7
3 Algoritmos	0,00010	234	0,00017	24	0,00017	22	0,00022	15	0,00022	8

7.4 Avaliação Experimental Utilizando Seleção das Melhores Regras

Nesta seção é descrito o trabalho realizado durante os três meses de estágio realizado no exterior.

O estágio no exterior foi realizado durante o período de 10/09/2002 à 10/12/2002 e visou um aprimoramento do conhecimento da bolsista, bem como uma aquisição de novas técnicas, algoritmos e conhecimentos utilizados por outros grupos que desenvolvem pesquisas nessa área ou em áreas a fim. Assim, esse estágio contribuiu de forma a enriquecer o trabalho de doutorado realizado pela aluna, bem como o contato com um novo grupo de pesquisadores.

Para realizar as implementações foi estudada e utilizada a ferramenta R que é voltada para Mineração de Dados. R é um ambiente estatístico e gráfico gratuito. Sua capacidade e o grande conjunto de pacotes disponíveis torna essa ferramenta uma excelente alternativa para as ferramentas de Mineração de Dados existentes e, geralmente, caras.

R é visto como um dialeto da linguagem S (desenvolvida pela AT&T) que premiou John Chambers com o *1998 Association for Computing Machinery (ACM) Software Award*. Ele mencionou que esta linguagem “irá alterar para sempre como as pessoas analisam, visualizam e manipulam dados”.

R pode ser bastante útil apenas usando sua parte interativa. Além disso, usos mais avançados do sistema levará o usuário a desenvolver suas próprias funções para sistematizar as tarefas repetitivas, ou até mesmo para adicionar ou alterar algumas funcionalidades dos pacotes existentes, tirando proveito de ser um código aberto.

Mais informação sobre a ferramenta R pode ser obtida em <http://www.r-project.org/> ou em <http://www.liacc.up.pt/~ltorgo/DataMiningWithR/>, livro que está sendo escrito pelo Prof. Dr. Luis Fernando Raíño Alves Torgo, supervisor da bolsista durante estágio no exterior.

O objetivo deste trabalho foi a integração do conhecimento para problemas de regressão visando explorar as vantagens de se possuir várias fontes de conhecimento. Com isso, pretende-se obter um ganho em precisão e não obter perda de interpretabilidade.

Para que essa verificação fosse viável, foi implementado um algoritmo para integração de conhecimento utilizando a ferramenta R e avaliado o método em diferentes bases de dados, a fim de comparar os resultados com outras abordagens de modelos múltiplos, como o *bagging*.

7.4.1 Descrição dos Experimentos

Nestes experimentos a integração foi realizada com *10-fold cross-validation* e utilizando-se um algoritmo que gera árvores de regressão da ferramenta R. Todas as árvores foram transformadas em regras e agrupadas em um único conjunto de regras. As melhores regras foram selecionadas baseado na medida de erro MAD e em uma medida de especificidade, que verifica quantos exemplos são cobertos por aquela regra e nenhuma outra regra. A função utilizada para a escolha da melhor regra é apresentada na Equação 7.2.

$$f(MAD, especif) = w_{MAD} \times \frac{Max(MAD) - MAD}{Max(MAD) - Min(MAD)} + (1 - w_{MAD}) \times especif \quad (7.2)$$

Na Figura 7.9 pode ser observado cada execução do *10-fold cross-validation*, sendo que esse procedimento foi realizado em cinco iterações, a fim de verificar a média.

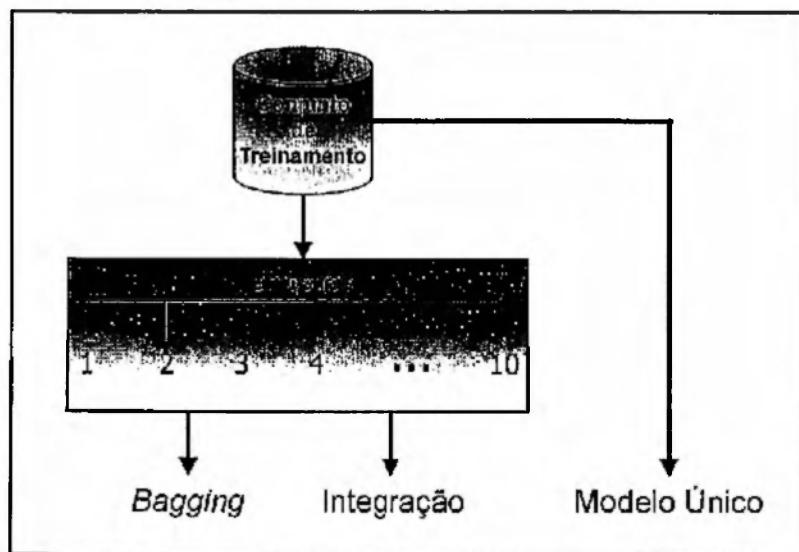


Figura 7.9: Esquema da metodologia utilizando a seleção das melhores regras.

7.4.2 Resultados

A fim de se realizar uma comparação, foram realizados quatro tipos de estratégias: duas utilizando o método de combinação de regressores *bagging*, porém com parâmetros diferentes, a terceira utilizando a técnica de integração descrita e a última executando-se o algoritmo de regressão apenas uma vez. Ressalta-se que a integração foi realizada com as regras geradas pelo segundo *bagging* (o *Bagging2* mostrado nas tabelas). Nas Tabelas 7.22 e 7.23 são apresentados os resultados encontrados em dez bases de dados distintas. O erro foi medido utilizando a medida RMSE (Equação 5.22) e o número de regras.

Tabela 7.22: Resultados para as bases de dados Servo, Auto-MPG, Housing, Abalone e Elevators.

	Servo		Auto-MPG		Housing		Abalone		Elevators	
	Erro	#R	Erro	#R	Erro	#R	Erro	#R	Erro	#R
Bagging1	0,2008 ± 0,17	120,38 ± 3,13	0,1354 ± 0,05	292,84 ± 4,7	0,1778 ± 0,10	388,44 ± 5,8	0,4577 ± 0,03	3170,9 ± 15,2	0,3787 ± 0,02	7203,2 ± 23,8
Bagging2	0,2188 ± 0,17	42,7 ± 1,9	0,1616 ± 0,05	82,8 ± 3,8	0,2043 ± 0,11	88,24 ± 3,9	0,5005 ± 0,03	110,26 ± 4,2	0,4220 ± 0,01	73,68 ± 1,9
Integração	0,2738 ± 0,22	17,06 ± 8,3	0,1926 ± 0,07	39,36 ± 11,3	0,2423 ± 0,11	30,54 ± 12,2	0,5308 ± 0,04	76,84 ± 14,8	0,4287 ± 0,01	35,5 ± 11,5
Modelo Único	0,3091 ± 0,19	—	0,2099 ± 0,08	—	0,2735 ± 0,12	—	0,5600 ± 0,04	—	0,4408 ± 0,01	—

Tabela 7.23: Resultados para as bases de dados Computer Activity, Kinematics, Pole, Ailerons e Census.

	Computer Activity		Kinematics		Pole		Ailerons		Census	
	Erro	#R	Erro	#R	Erro	#R	Erro	#R	Erro	#R
Bagging1	0,0274 ± 0,01	5779,1 ± 21,5	0,3301 ± 0,01	6162 ± 21,1	0,0171 ± 0,002	3049,7 ± 29,1	0,1687 ± 0,01	10368 ± 30,8	0,3857 ± 0,02	15858 ± 44,9
Bagging2	0,0728 ± 0,01	50,06 ± 0,23	0,6313 ± 0,02	80,84 ± 4,8	0,1140 ± 0,01	77,38 ± 2,7	0,3198 ± 0,02	97,64 ± 2,3	0,5569 ± 0,03	133,48 ± 5,2
Integração	0,0792 ± 0,01	22,96 ± 6,9	0,6279 ± 0,02	53,88 ± 12,1	0,1180 ± 0,01	46,4 ± 8,7	0,3297 ± 0,02	62,92 ± 19	0,6037 ± 0,03	87,46 ± 25,2
Modelo Único	0,0826 ± 0,02	—	0,6690 ± 0,02	—	0,1351 ± 0,01	—	0,3475 ± 0,02	—	0,6214 ± 0,03	—

Os resultados obtidos para esta estratégia de integração mostrou um comprometimento entre a precisão e a interpretabilidade. Tais modelos foram mais precisos que o modelo único mas não tão interpretáveis, e menos precisos que o método *bagging* mas mais interpretáveis.

Mais experimentos com diferentes métodos e parâmetros para a integração devem ser explorados com o objetivo de melhorar a precisão.

7.5 Considerações Finais

Entre os tópicos de Data Mining que têm recebido grande atenção por diversos pesquisadores está a integração do conhecimento obtido a partir de um ou mais preditores, considerando ainda a qualidade e desempenho desse conhecimento, além de sua robustez, consistência e interpretabilidade. Esse processo vem sendo estudado com mais ênfase na forma como a integração ocorre, e com menos atenção no conhecimento integrado.

O processo de integração de conhecimento consiste em juntar diferentes conhecimentos, gerados a partir de amostras diferentes da mesma base de dados, em uma única base de conhecimento, que seja mais completa que cada uma das bases em separado e mantenha a consistência. Isso envolve determinar como o conhecimento proveniente de diversas fontes interage e como o conhecimento deve ser modificado para acomodar novos conhecimentos.

Para avaliar o conhecimento gerado tanto de regressores simples quanto por meio da integração de conhecimento, foram considerados o desempenho e a qualidade do conheci-

mento obtido, utilizando-se como conjunto de treinamento diferentes amostras da mesma base de dados, sendo que todas elas podem ser submetidas ao mesmo algoritmo de aprendizado ou cada uma delas a um algoritmo diferente. Devido ao grande número de padrões, é difícil para o usuário compreendê-los e identificar quais são interessantes.

Os experimentos foram realizados com várias bases de dados retiradas principalmente do repositório de dados da UCI e três algoritmos. Os resultados obtidos são animadores e mostram que a abordagem por nós proposta para integrar regras de regressão é promissora. Em todos os casos, quando as regras são integradas em um conjunto de regras único, os erros são menores que uma única execução do algoritmo.

Para a definição da estratégia de eliminação de regra, levou-se em consideração as medidas MAD e cobertura para regras de regressão. Nossa contribuição é que a abordagem proposta, como uma maneira de integrar regras em um conjunto de regras final, melhora em vários casos a precisão. Além disso, pode-se aumentar a comprehensibilidade do modelo eliminando as piores regras.

Conclusões

A grande quantidade de dados que pode ser armazenada nos dias de hoje a um custo relativamente baixo e a necessidade das empresas de extrair conhecimento a partir de suas bases de dados impulsionaram o processo de Mineração de Dados, responsável por identificar padrões ou modelos embutidos em grandes conjuntos de dados. O processo de Mineração de Dados possui duas grandes atividades: a predição e a descrição. A predição, por sua vez, apresenta dois tipos de problemas, que são a classificação e a regressão. A regressão tem como objetivo encontrar uma relação entre um conjunto de atributos de entrada e um atributo-meta que seja contínuo. No entanto, a maioria das pesquisas realizadas nas áreas de Aprendizado de Máquina e Mineração de Dados são voltadas para os problemas de classificação, que são mais encontrados na vida real do que os problemas de regressão. Porém, como muitos problemas são de regressão, é importante também estudar métodos para a exploração de tarefas desse tipo.

O pós-processamento do conhecimento do processo de Mineração de Dados pode ser realizado de acordo com diversos objetivos. Este trabalho de doutorado explorou o pós-processamento de regras de regressão. Dentre as atividades de pós-processamento está a integração de conhecimento obtido de diversas amostras e/ou diferentes métodos utilizando métricas de desempenho e cobertura que possam ser usadas para verificar quão bom é o conhecimento. Para isso, foram utilizadas medidas de avaliação de regras de regressão, que também é uma atividade de pós-processamento. Dentre essas medidas, várias delas são derivadas da matriz de contingência, sendo proposta nesta tese estratégias para o cálculo da mesma. Outra forma de se realizar pós-processamento é por meio da combinação de regressores, que visa uma melhora na precisão do modelo.

Visando auxiliar na construção de um modelo de regressão simbólico e o pós-processamento do mesmo, foi proposto e desenvolvido o Ambiente *REPPE*. Esse ambiente oferece suporte ao pós-processamento de regras de regressão, e foi implementado de acordo com as especificações do DISCOVER, apresentadas na Seção 4.1.2, que é um ambiente de maior porte para apoiar todas as etapas do processo de Mineração de Dados.

Dentre as funcionalidades que o Ambiente *REPPE* disponibiliza estão:

- definição da sintaxe padrão para regras de regressão e de conversores, visando a conversão das saídas dos algoritmos de regressão para uma sintaxe padrão do DISCOVER;
- proposta de estratégias para calcular os valores da matriz de contingência para regras de regressão, o que torna possível a utilização de diversas medidas que são nela baseadas;
- avaliação de regras de regressão por meio do cálculo de diferentes medidas, tanto as derivadas da matriz de contingência quanto as não derivadas, e a possibilidade de realização de um tratamento semântico de modo a melhorar a comprehensibilidade das regras;
- combinação de regressores homogêneos e heterogêneos com o objetivo de aumentar a precisão dos resultados;
- integração de regras de regressão provenientes de diferentes amostras e diferentes algoritmos em uma base de conhecimento única, visando a diminuição do erro do regressor final;
- seleção das melhores regras de regressão, após a integração de regras, ou poda das piores regras com o objetivo de melhorar tanto a precisão quanto a comprehensibilidade do modelo gerado.

8.1 Contribuições desta Tese

O desenvolvimento do Ambiente *REPPE* objetivou o pós-processamento de regras de regressão com o apoio de módulos com funcionalidades específicas voltadas para: avaliação da qualidade do conhecimento obtido, aumento do desempenho, obtenção de uma única base de conhecimento sem perder a interpretabilidade da mesma, e geração de modelos que viabilizem a resposta para um exemplo desconhecido. O conhecimento extraído pode ser obtido por meio de diversas maneiras, como: amostras variadas do conjunto de

treinamento e um único algoritmo de aprendizado, diversas amostras do conjunto de treinamento e distintos algoritmos de aprendizado, uma amostra do conjunto de treinamento e diferentes algoritmos de aprendizado.

Para a avaliação de regras uma série de medidas podem ser utilizadas. Muitas delas baseadas nos valores da matriz de contingência. Uma das principais contribuições dessa tese refere-se a proposta, implementação e validação de três diferentes estratégias para cálculo da matriz de contingência para regras de regressão. Dentre essas estratégias, duas foram adaptações de propostas existentes para resolver outros problemas como o uso de pseudo-classe, que foi proposta por (Weiss & Indurkha, 1995) com a finalidade de transformar um problema de regressão em classificação por meio da discretização do atributo meta, e nesta tese foi adaptada para calcular valores da matriz de contingência. Também nessa linha foram utilizados métodos de discretização disponíveis no MineSetTM, e a estratégia por nós criada que utiliza um ajustamento a fim de encontrar um limiar entre o valor real e o valor predito. Ressalta-se que a estratégia por nós desenvolvida obteve o melhor comportamento nos experimentos realizados e que foram apresentados no Capítulo 5. Com a viabilização do cálculo da matriz de contingência, todas as medidas derivadas dessa matriz foram disponibilizadas para regras de regressão no Ambiente *REPPÉ* e no DISCOVER.

Com relação a melhora de precisão dos regressores foram implementados os métodos de *bagging* e *boosting* para problemas de regressão. A principal contribuição, desta tese, nesse aspecto refere-se ao *boosting* uma vez que a sua implementação necessita verificar se a predição do modelo para um exemplo está correta ou não. Porém, essa verificação não é trivial quando se trabalha com problemas de regressão. Para implementar a combinação de regressores homogêneos usando *boosting* foi utilizada a estratégia de ajustamento para verificar se o exemplo é corretamente coberto pela regra, apresentada no Capítulo 5. Adicionalmente, utilizou-se o método da roleta para a seleção dos exemplos que fariam parte do próximo conjunto de dados, uma vez que o peso de cada exemplo era diferente. Com a avaliação experimental pode-se confirmar que a combinação de regressores pode aumentar a precisão do conhecimento extraído quando comparada a um único regressor, porém perde-se a interpretabilidade da resposta. A descrição dos métodos e dos experimentos foi apresentada no Capítulo 6.

Uma outra contribuição desta tese é a integração de conhecimento que foi apresentada no Capítulo 7. O processo de integração de conhecimento consiste em unir diferentes conhecimentos, gerados a partir de várias amostras da mesma base de dados, em uma única base de conhecimento, que seja mais completa que cada uma das bases individuais. A metodologia que viabiliza a integração de regras de regressão foi apresentada na Figura 7.1.

A integração de conhecimento aqui proposta elimina as piores regras considerando duas medidas. Uma refere-se a uma medida de cobertura, que foi por nós definida, baseada

na matriz de contingência e a outra utiliza a medida de erro MAD que é uma medida já consagrada em avaliação de precisão em regressão. Os resultados obtidos são animadores e mostram que as abordagens definidas são promissoras. Além disso, pode-se aumentar a comprehensibilidade do modelo por meio da poda das piores regras.

Observa-se que essa integração também é um alternativa para a melhora da precisão de regressores, como é a combinação de regressores, com a vantagem de não perder a interpretabilidade dos resultados.

Ressalta-se também como contribuição dessa tese, que o Ambiente *REPPE* vem incrementar a potencialidade do DISCOVER no tratamento de problemas de regressão. Para isso foi definida a sintaxe padrão para representar regras de regressão, e foram implementados os conversores de dados da sintaxe padrão de dados para o formato de dados dos algoritmos, e os conversores da saída desses algoritmos para a sintaxe padrão de regras de regressão. O Ambiente *XEPPE* auxilia na preparação dos dados para a construção de um regressor simbólico e no pós-processamento do conhecimento obtidos desses regressores, que inclui um tratamento semântico das regras aumentando, assim, a comprehensibilidade do regressor final. Além disso, disponibiliza um tratamento semântico para regras de regressão, que é muito importante especialmente para as regras que são originalmente convertidas de uma árvore de regressão.

8.2 *Trabalhos Futuros*

A seguir são apresentadas algumas propostas de trabalhos futuros que visam complementar, incrementar e dar sequência a esta tese:

- validação da proposta de construção da matriz de contingência para regras de regressão por meio da utilização de outras bases de dados focando em características diferentes;
- análise de outras métricas para integração e poda a serem incorporadas na metodologia *IRR* do Ambiente *REPPE*, verificando-se o seu comportamento;
- implementação de conversores para apoiar a adição de novos algoritmos de regressão a serem utilizados para os diferentes módulos do Ambiente *REPPE* no pós-processamento;
- realização de experimentos utilizando outros conjuntos de dados para melhor validação dos módulos do Ambiente *REPPE*;
- utilização da poda de regras proposta na metodologia *IRR*, para um único regressor, sem integração, de modo a deixar o mínimo de regras para análise do usuário

final;

- disponibilização, na metodologia *IIRR*, de mecanismos para apoiar o usuário no momento de verificar se o conhecimento pode se tornar inconsistente ou inválido com a integração de conhecimento;
- realização de poda de regras baseada no interesse do usuário final;
- seleção de regras de regressão apoiada por um especialista do domínio;
- avaliação da comprehensibilidade de regras de regressão com apoio de ferramenta para visualização de conhecimento, incrementando assim o Ambiente *REPPE* com um *framework* de técnicas de Mineração Visual de Dados para apoiar na interpretação do conhecimento.

Referências Bibliográficas

- Aas, K. & L. Eikvil (1999). Text categorization: A survey. Relatório Técnico 941, Disponível em: <http://www.nr.no/research/samba/textmining.html> [10/02/2000]. 117
- Agrawal, R. & R. Srikant (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases*. <http://www.almaden.ibm.com/u/ragrawal/pubs.html>. 16
- Allen, A. O. (1990). *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Academic Press, Inc. 92
- Atkeson, C. G., A. W. Moore, & S. Schaal (1997). Locally weighted learning. *Artificial Intelligence Review* 11(1-5), 11-73. 29
- Baeza-Yates, R. & R.-N. B. (1999). *Modern Information Retrieval*. Addison-Wesley. 21
- Baranauskas, J. A. (2001). *Extração Automática de Conhecimento por Múltiplos Indutores*. Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 46
- Baranauskas, J. A. & G. E. A. P. A. Batista (2000). O projeto DISCOVER: Idéias iniciais (comunicação pessoal). 42
- Batista, G. E. A. P. A. (2002). Definição da sintaxe DSX. Disponível em: <http://www.icmc.usp.br/~gbatista/Discover/SintaxePadraoFinal.html> [28/01/2004]. 44, 45, 46, 121
- Batista, G. E. A. P. A. (2003). *Pré-processamento de Dados em Aprendizado de Máquina Supervisionado*. Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 12, 44, 46, 47

- Batista, G. E. A. P. A., A. C. P. L. F. Carvalho, & M. C. Monard (2000). Applying one-sided selection to unbalanced datasets. In *Proceedings of the Mexican International Conference on Artificial Intelligence – MICAI 2000*, Lecture Notes in Artificial Intelligence, pp. 315–325. Springer Verlag. 13
- Batista, G. E. A. P. A. & M. C. Monard (2003). Descrição da arquitetura e do projeto do ambiente computacional Discover Learning Environment – DLE. Relatório Técnico 187, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos, São Carlos. Disponível em: ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_187.pdf [28/01/2004]. 46, 47
- Bauer, E. & R. Kohavi (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning* 36, 105–142. 105, 108, 117
- Bernardini, F. C. (2002). Combinação de classificadores para melhorar o poder preditivo e descritivo de ensembles. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Blake, C. L. & C. J. Merz (1998). UCI repository of machine learning databases. Disponível em: <http://www.ics.uci.edu/~mlearn/MLRepository.html> [28/01/2004]. 51
- Bradley, P., U. Fayyad, & O. Mangasarian (1998). Data mining: Overview and optimization opportunities. Relatório Técnico MSR-TR-98-04, Microsoft Research Report, Redmond, WA. 8
- Braga, A. P., A. C. P. L. F. Carvalho, & T. B. Ludermir (2000). *Redes Neurais Artificiais: Teoria e Aplicações*. Rio de Janeiro, Brasil: LTC Press. 34, 35
- Braga, A. P., A. C. P. L. F. Carvalho, & T. B. Ludermir (2003). *Redes Neurais Artificiais* (1 ed.), Capítulo 6, pp. 141–168. In Rezende (2003). 35
- Brazdil, P. & L. Torgo (1990). Knowledge acquisition via knowledge integration. In B. J. Wielinga, J. Boose, B. Gaines, G. Schreiber, & S. M. Van (Eds.), *Current Trends in Knowledge Acquisition*, pp. 90–104. IOS Press. 132, 133
- Brazdil, P. & L. Torgo (1991). Knowledge integration and learning. Relatório Técnico 91-1, LIACC, Machine Learning Group. 132, 133
- Breiman, L. (1996a). Bagging predictors. *Machine Learning* 24(2), 123–140. 104
- Breiman, L. (1996b). Stacked regressions. *Machine Learning* 24(1), 49–64. 115, 117
- Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics* 26(3), 801–849. 111, 112

- Breiman, L. (2000a). Randomizing outputs to increase prediction accuracy. *Machine Learning* 40(3), 229–242. 101, 111
- Breiman, L. (2000b). Some infinity theory for predictor ensemble. Relatório Técnico 577, University of California, Berkeley. Disponível em: <http://citeseer.nj.nec.com/breiman00some.html> [10/01/2001]. 18
- Breiman, L., J. H. Friedman, C. J. Stone, & R. A. Olshen (1984). *Classification and Regression Trees*. New York: Chapman & Hall / CRC Press. 51
- Carvalho, A. C. P. L. F., A. P. Braga, & T. B. Ludermir (2003). *Computação Evolutiva* (1 ed.), Capítulo 9, pp. 225–248. In Rezende (2003). 121
- Caulkins, C. W. (2000). Aquisição de conhecimento utilizando aprendizado de máquina relacional. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Chan, P. & S. Stolfo (1995). A comparative evaluation of voting and meta-learning on partitioned data. In A. Prieditis & S. Russel (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 90–98. Morgan Kaufmann. 116
- Chan, P. & S. Stolfo (1997). The accuracy of meta-learning for scalable data mining. *Journal of Intelligent System* 8, 5–28. 116
- Cheng, S. (1998). Statistical approaches to predictive modeling in large databases. Dissertação de Mestrado, Simon Fraser University, British Columbia, Canada. 76
- Clark, P. & T. Niblett (1989). The CN2 induction algorithm. *Machine Learning* 3(4), 261–283. 20, 89
- Delve (2003). Data for evaluating learning in valid experiments (DELVE). Disponível em: <http://www.cs.toronto.edu/~delve/> [28/01/2004]. 51
- Dietterich, T. G. (2000a). Ensemble methods in machine learning. *Multiple Classifier Systems*, 1–15. 18, 101, 102
- Dietterich, T. G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* 40(2), 139–157. 104, 111
- Dietterich, T. G. & G. Bakiri (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence* 2, 263–286. 113

Dietterich, T. G. & E. B. Kong (1995). Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Relatório técnico, Department of Computer Science, Oregon State University, Corvallis, Oregon. Disponível em: <ftp://ftp.cs.orst.edu/pub/tgd/papers/tr-bias.ps.gz> [7/11/2000]. 113

Domingos, P. (1997). Why does bagging work? A bayesian account and its implications. In D. Heckerman, H. Mannila, D. Pregibon, & R. Uthurusamy (Eds.), *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 155–158. AAAI Press. 105

Domingues, M. A. (2003). Uso de taxonomias na avaliação de regras de associação. Minidissertação para Qualificação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48

Dong, G. & J. Li (1998). Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. In *Proceedings of the 2nd Pacific-Asia Conf. Knowledge Discovery and Data Mining, PAKDD, Lecture Notes in Artificial Intelligence, LNAI*, Volume 1394, Melbourne, Australia, pp. 72–86. 19, 84

Dosualdo, D. G. (2003). Investigação de regressão no processo de mineração de dados. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 47, 90

Dosualdo, D. G. & S. O. Rezende (2003a). Análise da precisão de métodos de regressão. Relatório Técnico 197, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_197.zip [27/01/2004]. 46, 57

Dosualdo, D. G. & S. O. Rezende (2003b). Descrição do ambiente computacional Dipper. Relatório Técnico 204, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_204.zip [27/01/2004]. 47

Faceli, K., A. Carvalho, & S. Rezende (2002). Combining intelligent techniques for sensor fusion. In *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02)*, Volume 4, Singapore, pp. 1998–2002. 35

Fan, D., P. Chan, & S. Stolfo (1996). A comparative evaluation of combiner and stacked generalization. In *Proceedings of AAAI-96 Workshop on Integrating Multiple Learned Models*, pp. 40–46. 115

Fayyad, U., D. Haussler, & P. Stolorz (1996). KDD for science data analysis: Issues and examples. In E. Simoudis, J. W. Han, & U. Fayyad (Eds.), *Proceedings of the*

Second International Conference on Knowledge Discovery and Data Mining (KDD'96), pp. 50–56. AAAI Press. 2

Fayyad, U., G. Piatetsky-Shapiro, & P. Smith (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of ACM* 39(11), 27–34. 1

Fayyad, U., G. Piatetsky-Shapiro, & P. Smyth (1996a). From data mining to knowledge discovery: an overview. In *Advances in Knowledge Discovery and Data Mining*, pp. 1–34. 8

Fayyad, U., G. Piatetsky-Shapiro, & P. Smyth (1996b). Knowledge discovery and data mining: Towards a unifying framework. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pp. 82–88. AAAI Press. 8, 11

Fayyad, U. M. (1996). Data mining and knowledge discovery: Making sense out of data. *IEEE Expert-Intelligent & Their Applications* 11(5), 20–25. 9, 13, 16, 17

Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (1996). *Advances in Knowledge Discovery and Data Mining*. AAAI Press. 1

Félix, L. C. M., S. O. Rezende, M. C. Monard, & C. W. Caulkins (2000). Transforming a regression problem into a classification problem using hybrid discretization. *Computación y Sistemas* 4, 44–52. 14

Fertig, C. S., A. A. Freitas, L. V. R. Arruda, & C. Kaestner (1999). A fuzzy beam-search rule induction algorithm. In *Proceedings of the Third European Conference (PKDD-99)*, Volume 1704 of *Lecture Notes in Artificial Intelligence*, pp. 341–347. 19

Félix, L. C. M. (1998, Agosto). Data mining no processo de extração de conhecimento de bases de dados. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48

Freitas, A. A. (1998a). A multi-criteria approach for the evaluation of rule interestingness. In *Proceedings of International Conference on Data Mining*, pp. 7–20. WIT Press. 19, 84, 133

Freitas, A. A. (1998b). On objective measures of rule surprisingness. *Principles of Data mining & Knowledge Discovery: Proceedings of the 2nd European Symposium. Lecture Notes in Artificial Intelligence* 1510, 1–9. 8, 84

Freund, Y. & R. E. 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. 106

- Freund, Y. & R. E. Shapire (1996). Experiments with a new boosting algorithm. In *13 th. International Conference on Machine Learning*, pp. 148–156. 106
- Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics* 19(1), 1–141. 30
- Fu, Y. & J. Han (1995). Meta-rule-guided mining of association rules in relational databases. In *Proceedings 1995 International Workshop on Knowledge Discovery and Deductive and Object-Oriented Databases (KDOOD'95)*, Singapore, pp. 39–46. 85
- Fürnkranz, J. (1998). Integrative windowing. *Journal of Artificial Intelligence Research* 8, 129–164. 108, 109, 110
- Gama, J. M. P. (1999). *Combining Classification Algorithms*. Tese de Doutorado, Faculdade de Ciências da Universidade do Porto, Departamento de Ciências de Computadores. 106, 114, 117
- Gardner, S. R. (1998). Building the data warehouse. *Communications of the ACM* 41(9), 52–60. 7
- Geromini, M. R. (2002). Projeto e desenvolvimento da interface gráfica do sistema DISCOVER. Monografia de Qualificação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 43, 48
- Glymour, C., D. Madigan, D. Pregibon, & P. Smyth (1997). Statistical themes and lessons for data mining. *Data Mining and Knowledge Discovery* 1(1), 11–28. 3, 13
- Gomes, A. K. (2002). Análise do conhecimento extraído de classificadores simbólicos utilizando medidas de avaliação e de interessabilidade. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Gonçalves, L. S. M. (2002). Técnicas para extração de conhecimento de textos. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Habn, U. & I. Mani (2000). The challenges of automatic summarization. *IEEE Computer* 33(11), 29–36. 21
- Hastie, T., R. Tibshirani, & J. Friedman (2001). *The Elements of Statistical Learning – Data Mining, Inference and Prediction*. New York: Springer-Verlag. 30, 33
- Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2 ed.). New Jersey: Prentice-Hall. 35

- Hilderman, R. & H. Hamilton (2001). Evaluation of interestingness measures for ranking discovered knowledge. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'01)*, Hong Kong, pp. 247–259. 11, 133
- Horst, P. S. (1999). Avaliação do conhecimento adquirido por algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48, 76, 84
- Hsu, C.-N. & C. A. Knoblock (1996). Discovering robust knowledge from dynamic closed-world data. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pp. 820-827. 134
- Imamura, C. Y. (2001). Pré-processamento para extração de conhecimento de bases textuais. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Karalic, A. (1995). Retis -- a machine learning system. Disponível em: <http://www-ai.ijc.sj/AramKaralic/retis/index.html> [12/10/2002]. 51
- Kearns, M. J. & U. V. Vazirani (1994). *An introduction to computational learning theory*. Ellis Horwood. 17
- Klemettinen, M., H. Mannila, P. Ronkainen, H. Toivonen, & A. I. Verkamo (1994). Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third International Conference on Information and Knowledge Management*, Gaithersburg, Maryland, pp. 401-407. 85
- Kohavi, R., D. Sommerfield, & J. Dougherty (1996). Data mining using *MLC++*: A machine learning library in *C++*. In *Tools with Artificial Intelligence*, pp. 234–245. IEEE Computer Society Press. 17
- Lavrac, N., P. Flach, & R. Zupan (1999). Rule evaluation measures: A unifying view. In S. Dzeroski & P. Flach (Eds.), *Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP'99)*, Volume 1634 of *Lecture Notes in Artificial Intelligence*, pp. 174–185. Springer-Verlag. 19, 69, 76
- LeBlanc, M. & R. Tibshirani (1996). Combining estimates in regression and classification. *Journal of the American Statistical Association* 91(436), 1641–1650. 101
- Lee, H. D. (2000). Seleção e construção de features relevantes para o aprendizado de máquina. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 14, 48

Liu, B. & W. Hsu (1996). Post-analysis of learned rules. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI '96)*, pp. 828–834. 2, 11, 18, 75, 82, 86

Liu, B., W. Hsu, & S. Chen (1997). Using general impressions to analyze discovered classification rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 31–36. 75

Liu, B., W. Jsu, Y. Ma, & S. Chen (1999). Mining interesting knowledge using DM-II. In S. Chaudhuri & D. Madigan (Eds.), *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, pp. 430–434. ACM Press. 86, 133

Magalhães, M. N. & A. C. P. de Lima (2002). *Noções de Probabilidade e Estatística*. Edusp – Editora da Universidade de São Paulo. 92

Mannila, H. (1997). Methods and problems in data mining. *Database Theory ICDT'97, 6th International Conference, Lecture Notes in Computer Science 1186*, 41–55. 2

Martins, C. A. (2003). *Uma abordagem para pré-processamento de dados textuais em algoritmos de aprendizado*. Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 47

Matsubara, E. T., C. A. Martins, & M. C. Monard (2003). PreTexT: uma ferramenta para pré-processamentos de textos utilizando a abordagem bag-of-words. Relatório Técnico 209, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. http://www.icmc.usp.br/~biblio/BIBLIOTECA/rel_tec/RT_209.zip [27/01/2004]. 47

Melandia, E. A. (2002). Pós-processamento de conhecimento de regras de associação. Monografia do Exame de Qualificação de Doutorado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48

Melandia, E. A. & S. O. Rezende (2003). Sintaxe padrão para representar regras de associação. Relatório Técnico 206, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos, São Carlos – SP. 47, 48

Merz, C. J. (1998). *Classification and Regression by Combining Models*. Tese de Doutorado, University of California, Irvine. Disponível em: <http://www.ics.uci.edu:80/~cmerz/thesis.ps> [28/01/2004]. 81, 114

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Series in Computer Science. 17, 27, 30, 82

- Murray, K. S. (1996). KI: A tool for knowledge integration. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI/IAAI*, Volume 1, pp. 835–842. 131
- Nagai, W. A. (2000). Avaliação do conhecimento extraído de problemas de regressão. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Navathe, S. & M. Donahoo (1995). Towards intelligent integration of heterogeneous information sources. In *Proceedings of the 6th International Workshop on Database Re-engineering and Interoperability*, pp. 275–282. 132
- Netz, A., S. Chaudhuri, J. Bernhardt, & U. M. Fayyad (2000). Integration of data mining with database technology. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, & K.-Y. Whang (Eds.), *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000), Cairo, Egypt*, pp. 719–722. Morgan Kaufmann. 1
- Padilha, T. P. P. (1999, Março). Investigação de algoritmos de aprendizado de máquina pertencentes ao paradigma estatístico para aquisição de conhecimento. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Padmanabhan, B. & A. Tuzhilin (1999). Unexpectedness as a measure of interestingness in knowledge discovery. *Decision Support Systems* 27, 303–318. 86
- Paula, M. F. (2003). Ambiente para exploração de regras. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Pazzani, M., S. Mani, & W. Shankle (1997). Comprehensible knowledge discovery in databases. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, pp. 596–601. 19
- Pazzani, M. J. (2000a). Knowledge discovery from data? *IEEE Intelligent Systems* 15(2), 10–13. 19
- Pazzani, M. J. (2000b). Knowledge discovery from data? *IEEE Intelligent Systems* 15(2), 10–13. 83
- Piatetsky-Shapiro, G. (1991). Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro & W. J. Frawley (Eds.), *Knowledge Discovery in Databases*, pp. 229–248. AAAI/MIT Press. 76, 84
- Piatetsky-Shapiro, G. & C. J. Matheus (1994). The interestingness of deviations. In *Proceedings of the Knowledge Discovery in Databases (KDD'94)*, pp. 23–36. 11, 19, 86

- Pila, A. D. (2003, Outubro). Algoritmos genéticos para construção de regras de conhecimento com propriedades específicas. Exame de Qualificação de Doutorado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Prati, R. C. (2003). O framework de integração do sistema DISCOVER. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 43, 48
- Prati, R. C., J. A. Baranauskas, & M. C. Monard (2001). Uma proposta de unificação da linguagem de representação de conceitos de algoritmos de aprendizado de máquina simbólicos. Relatório Técnico 137, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos, São Carlos. Disponível em: ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip [28/01/2004]. 47
- Prati, R. C., M. R. Geromini, & M. C. Monard (2003a). A framework to integrate data mining components. In R. C. A. E. C. Cardenas (Ed.), *29 Conferencia Latinoamericana de Informática*, La Paz, Bolívia, pp. 10. Centro Latinoamericano de Estudios en Informática (CLEI). Publicado em CD. 48
- Prati, R. C., M. R. Geromini, & M. C. Monard (2003b). An integrated environment for data mining. In G. L. Torres, J. M. Abe, M. L. Mucheroni, & P. E. Cruvinel (Eds.), *Proceedings of the Fourth Congress of Logic Applied to Technology*, Marilia/SP, pp. 55–63. Editora Plêiade. 48
- Prati, R. C., M. R. Geromini, & M. C. Monard (2003c). Um framework baseado em patterns para integração de componentes de mineração de dados. In *IV Workshop on Advances & Trends in AI for Problem Solving (ATAI 2003)*, Chilán, Chile. Sociedad Chilena de Ciencia de la Computación. Publicado em CD. 48
- Pugliesi, J. B., D. G. Dosualdo, & S. O. Rezende (2003). Sintaxe padrão para representar regras de regressão no DISCOVER. Relatório Técnico 193, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos, São Carlos. Disponível em: ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_193.zip [28/01/2004]. 47, 57, 87, 121, 129
- Pugliesi, J. B. & S. O. Rezende (2003). Medidas de erro para regras de regressão. In R. de Oliveira Anido & P. C. Masiero (Eds.), *Anais do XXIII Congresso da Sociedade Brasileira de Computação – IV Encontro Nacional de Inteligência Artificial (ENIA 2003)*, Volume VII, pp. 517–526. SBC – Campinas. 72, 121, 129
- Pugliesi, J. B., R. A. Sinoara, & S. O. Rezende (2003). Combinação de regressores homogêneos e heterogêneos: Precisão e comprehensibilidade. Relatório Técnico 203, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos, São Carlos. Disponível em:

ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_203.zip [28/01/2004]. 118, 123

- Quinlan, J. (1996). Bagging, boosting and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 725–730. AAAI/MIT Press. 107
- Quinlan, J. R. (1992). Learning with continuous classes. In *Proceedings Australian Joint Conference on Artificial Intelligence*, pp. 343–348. World Scientific. 50
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann. 20, 89
- Ram, A. (1990). Knowledge goals: A theory of interestingness. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, Cambridge, MA,, pp. 206–214. 83
- Rathjens, D. (1996). *MineSetTM User's Guide*. Silicon Graphics, Inc. 91
- Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e Aplicações* (1 ed.). Barueri, SP: Manole. 37, 158, 159, 167
- Rezende, S. O., M. C. Monard, G. E. A. P. A. Batista, R. C. Prati, J. B. Pugliesi, & E. A. Melanda (2004). Ambiente DISCOVER: Sintaxes padrão de dados e regras. Relatório técnico, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos, São Carlos. Prelo. 44
- Rezende, S. O., J. B. Pugliesi, E. A. Melanda, & M. F. Paula (2003). *Mineração de Dados* (1 ed.), Capítulo 12, pp. 307–335. In Rezende (2003). 2, 9, 10
- Rocha, C. A. J. (1999). Redes bayesianas para extração de conhecimento de bases de dados, considerando a incorporação de conhecimento de fundo e o tratamento de dados incompletos. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos. 48
- Rulequest-Research (2001). Data mining with cubist. Disponível em: <http://www.rulequest.com/cubist-info.html> [28/01/2004]. 49, 91
- Sahar, S. (1999). Interestingness via what is not interesting. In S. Chaudhuri & D. Madigan (Eds.), *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, pp. 332–336. ACM Press. 84, 133
- Sahar, S. & Y. Mansour (1999). An empirical evaluation of objective interestingness criteria. In *SPIE Conference on Data Mining And Knowledge Discovery*, pp. 63–74. 84
- Schapire, R. (1999). A brief introduction to boosting. In *Proceedings of the 16 th International Joint Conference on Artificial Intelligence*, 1401–1405. 108

- Silberschatz, A. & A. Tuzhilin (1995). On subjective measures of interestingness in knowledge discovery. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining – KDD'95* 1, 275–281. 19, 20, 86
- Silberschatz, A. & A. Tuzhilin (1996). What makes patterns interesting in knowledge discovery patterns. *IEEE Transactions on Knowledge and Data Engineering* 8(6), 970–974. 86
- Sinoara, R. A., J. B. Pugliesi, & S. O. Rezende (2003). Combinação de classificadores homogêneos e heterogêneos. Relatório Técnico 184, Instituto de Ciências Matemáticas e de Computação – USP – São Carlos, São Carlos. Disponível em: ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_184.pdf [28/01/2004]. 118
- Skalak, D. B. (1997, May). *Prototype Selection for Composite Nearest Neighbor Classifiers*. Tese de Doutorado, University of Massachusetts Amherst. 116, 118
- Thuraisingham, B. (1999). *Data Mining: Technologies, Techniques, Tools, and Trends*. CRC Press LLC. 21
- Till, D. (1996). *Teach Yourself Perl 5 in 21 Days*. Sams Publishing. 43
- Ting, K. M. & I. H. Witten (1997). Stacked generalization: when does it work? In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, San Francisco, pp. 866–871. Morgan Kaufmann Publishers. 114, 115
- Toivonen, H., M. Klemettinen, P. Ronkainen, K. Hätkönen, & H. Mannila (1995). Prunning and grouping discovered association rules. In *MLNet Workshop on Statistics, Machine Learning and Discovery in Databases*, Crete, Greece, pp. 47–52. 86
- Torgo, L. (1995). Data fitting with rule-based regression. In *Proceedings of the 2nd International Workshop on Artificial Intelligence Techniques (AIT'95)*, Brno, Czech Republic, pp. 223–232. J. Zizka & P. Brazdil. 81, 84, 85
- Torgo, L. (1997). Functional models for regression tree leaves. In D. Fisher (Ed.), *Proceedings of the International Machine Learning Conference (ICML'97)*, pp. 385–393. Morgan Kaufmann Publishers. Disponível em: <http://www.ncc.up.pt/~ltorgo> [26/10/1999]. 33
- Torgo, L. (2001). Rt 4.1 user's manual. Disponível em: http://www.ncc.up.pt/~ltorgo/RT/rt_manual.pdf [25/07/2001]. 50
- Torgo, L. F. R. A. (1999). *Inductive Learning of Tree-Based Regression Models*. Tese de Doutorado, Faculdade de Ciências da Universidade do Porto. Disponível em: <http://www.liacc.up.pt/~ltorgo/PhD/thesis.ps.gz> [28/01/2004]. 25, 29, 33, 81

- Uysal, I. & H. A. Güvenir (1999). An overview of regression techniques for knowledge discovery. *The Knowledge Engineering Review* 14(4), 319–340. 23, 25, 28, 29, 30
- Vapnik, V. N. (1998). *Statistical Learning Theory*. Chichester, GB: Wiley. 35
- Vasileios, H., L. Gravano, & A. Maganti (2000). An investigation of linguistic features and clustering algorithms for topical document clustering. In *23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 224–231. 21
- Wang, Y. & I. H. Witten (1997). Inducing model trees for continuous classes. In *Proceedings of the Poster Papers, 9th European Conference on Machine Learning*, Department of Computer Science, University of Waikato, New Zealand, pp. 128–137. 50
- Weiss, S. M. & N. Indurkhya (1995). Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research* 3, 383–403. 4, 23, 24, 26, 33, 53, 89, 90, 153
- Weiss, S. M. & N. Indurkhya (1998). *Predictive Data Mining: A Practical Guide*. San Francisco, California: Morgan Kaufmann Publishers Inc. 9, 13, 14, 24, 83
- Witten, I. H. & E. Frank (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. São Francisco, Califórnia: Morgan Kaufmann. 49, 50, 118
- Wolpert, D. H. (1992). Stacked generalization. *Neural Networks* 5, 241–259. 114