UNIVERSIDADE FEDERAL FLUMINENSE

LUIZ HENRIQUE DE CAMPOS MERSCHMANN

CLASSIFICAÇÃO PROBABILÍSTICA BASEADA EM ANÁLISE DE PADRÕES

NITERÓI

2007

UNIVERSIDADE FEDERAL FLUMINENSE

LUIZ HENRIQUE DE CAMPOS MERSCHMANN

CLASSIFICAÇÃO PROBABILÍSTICA BASEADA EM ANÁLISE DE PADRÕES

Tese de Doutorado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Doutor. Área de concentração: Otimização Combinatória.

Orientador:
Alexandre Plastino de Carvalho

NITERÓI

2007

CLASSIFICAÇÃO PROBABILÍSTICA BASEADA EM ANÁLISE DE PADRÕES

Luiz Henrique de Campos Merschmann

Tese de Doutorado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Doutor.

Aprovada por:

Alexandre Plastino de Carvalho, D.Sc. / IC-UFF
(Presidente)

Nelson Francisco Favilla Ebecken, D.Sc. / COPPE-UFRJ

Andre Carlos Ponce de Leon Ferreira de Carvalho, Ph.D. / ICMC-USP

Ana Cristina Bicharra Garcia, Ph.D. / IC-UFF

Bianca Zadrozny, Ph.D. / IC-UFF

Niterói, 28 de agosto de 2007.

Dedico este trabalho à minha mãe,
que perdeu noites de sono para que eu dormisse tranqüilo e sonhasse,
que acreditou em mim, dando-me confiança para que eu lutasse pelos meus sonhos,
e que chorou de saudade quando a realização desses sonhos implicou em ficar longe de mim.

Agradecimentos

Muitas pessoas contribuíram para que eu concluísse com êxito esta tese de doutorado. Por isso, seguem os meus sinceros agradecimentos a todos que, direta ou indiretamente, me auxiliaram nessa conquista.

Inicialmente, agradeço aos meus pais, que não pouparam esforços para que eu pudesse alcançar os meus objetivos. Agradeço especialmente por terem me ensinado que, mesmo nos momentos mais difíceis, não devemos desistir de nossos sonhos.

À minha amada Michelle, que com amor, carinho e paciência, compartilhou comigo todas as alegrias e tristezas vividas durante o doutorado. A sua presença iluminou cada instante de nossas vidas durante essa jornada.

Ao professor Alexandre Plastino, orientador deste trabalho, por ter aceitado me orientar, pela amizade, paciência e apoio em todos os momentos necessários. Certamente, o seu entusiasmo como pesquisador e a sua postura profissional servirão como exemplo para mim por toda a vida.

Aos professores e funcionários da UFF, que muito contribuíram para a minha formação acadêmica. Agradeço de forma especial ao professor Luiz Satoru Ochi, pelo auxílio nos primeiros anos do doutorado, e aos professores que participaram da banca de avaliação desta tese.

A todos os amigos que conviveram comigo no Laboratório de Computação da Pós-Graduação, pelo incentivo e companheirismo.

Aos amigos Haroldo Gambini Santos e Euler Horta Marinho, que com sábias palavras, muito me ajudaram na vida acadêmica e pessoal.

Aos meus familiares, pelo inestimável apoio, incentivo e carinho.

Resumo

Classificação é uma das tarefas de Mineração de Dados que tem se mostrado útil em diversas áreas de aplicação, em particular, na área de Bioinformática.

A revolução genômica resultou em um crescimento exponencial da quantidade de dados biológicos gerados pela comunidade científica. Com a finalidade de armazenar toda essa informação biológica gerada, foram criados os bancos de dados biológicos. A necessidade por ferramentas computacionais capazes de realizar análises nesses dados tornou-se cada vez mais evidente, fazendo com que técnicas de mineração de dados começassem a ser empregadas.

O trabalho desta tese concentra-se na tarefa de classificação e, inicialmente, na sua aplicação em bioinformática. O objetivo inicial é apresentar um método de classificação de proteínas computacionalmente eficiente e capaz de alcançar altas taxas de acurácia, superando resultados apresentados anteriormente na literatura.

Os bons resultados, em termos de acurácia preditiva e tempo computacional, obtidos a partir do método proposto nesta tese, demonstram o seu potencial para o problema de classificação de proteínas.

Além disso, visando a construção de um classificador adequado para diversos tipos de aplicação, o método proposto inicialmente para o problema de classificação de proteínas foi estendido e mostrou-se eficiente também quando utilizado com diferentes tipos de bases de dados pertencentes a aplicações distintas.

Palavras-chave: mineração de dados, classificação, bioinformática.

Abstract

Classification is a data mining task that has been useful in several application areas, particularly, in bioinformatics.

The genomic revolution has resulted in an explosive growth of biological data generated by the scientific community. With the aim of storing all of these biological information, biological databases were created. The need for computational tools for analysing biological data becomes evident, resulting in the application of data mining methods in this field.

The work developed in this thesis is related to classification task and, initially, to its application to bioinformatics. The initial goal is to present a computationally efficient method for protein classification capable of yielding highly accurate results, outperforming the results obtained by previous works.

The good results in terms of accuracy and time performance obtained by the proposed method show its potential for the protein classification problem.

In addition, aiming to construct a suitable classifier for several kinds of applications, the method proposed for the protein classification problem was extended, becoming appropriate and efficient for several databases associated with different applications.

Keywords: data mining, classification, bioinformatics.

Glossário

ARCS	:	*Association Rule Clustering System*;
CAEP	:	*Classification by Aggregating Emerging Patterns*;
CARs	:	*Class Association Rules*;
CATH	:	*Class/Architecture/Topology/Homology*;
CBA	:	*Classification Based on Association*;
CFS	:	*Correlation-based Feature Selection*;
GI	:	Ganho de Informação;
EM	:	*Expectation Maximization*;
EPs	:	*Emerging Patterns*;
GPCR	:	*G-Protein Coupled Receptors*;
HiSP	:	***Hi**ghest **S**ubset **P**robability*;
HiSP-Prot	:	***Hi**ghest **S**ubset **P**robability - for **Prot**ein Classification Problem*;
IC	:	Instituto de Computação;
JEP	:	*Jumping Emerging Pattern*;
KDD	:	*Knowledge Discovery in Database*;
k-NN	:	*k-Nearest Neighbor*;
MEME	:	*Multiple Expectation Maximization for Motif Elicitation*;
MS SQL	:	*Microsoft Structured Query Language*;
NB	:	*NaiveBayes*;
PCP	:	*Protein Classification Problem*;
PDB	:	*Protein Data Bank*;
PIR	:	*Protein Identification Resource*;
RNA	:	Redes Neural Artificial;
SCOP	:	*Structural Classification of Proteins*;
SVM	:	*Support Vector Machines*;
UCI	:	*University of California, Irvine*;
UFF	:	Universidade Federal Fluminense;

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A quantidade de dados disponível no mundo, em ambientes computacionais, tem aumentado consideravelmente a cada dia. A necessidade por ferramentas computacionais capazes de analisar esses dados motivou o surgimento da área de pesquisa e aplicação em ciência da computação conhecida como mineração de dados [29]. Processos de mineração de dados permitem a transformação de dados, uma matéria bruta, em informação e conhecimento úteis em diversas áreas de aplicação, tais como administração, finanças, saúde, educação, *marketing*, entre outras.

O progresso verificado na tecnologia de *hardware* de computadores e nas técnicas de processamento, armazenamento e transmissão de informações digitais propiciou o acúmulo de grandes quantidades de dados históricos nas últimas décadas, motivando, a partir de 1990, o desenvolvimento de técnicas e ferramentas específicas de mineração de dados. Tais ferramentas realizam análises em dados com o objetivo de extrair informações novas e úteis.

De forma simples, tarefas em mineração de dados podem ser definidas como processos automatizados de descoberta de novas informações a partir de grandes massas de dados armazenadas em bancos de dados, arquivos de texto, *data warehouses*, ou em algum outro repositório de dados. Sendo uma área de estudo extensa e interdisciplinar, a mineração de dados envolve a integração de conceitos e técnicas de diversas áreas, tais como: banco de dados, estatística, inteligência artificial, visualização de dados e otimização.

Apesar de alguns autores utilizarem o termo mineração de dados como sinônimo de *KDD* (*Knowledge Discovery in Database*) [29] – processo de descoberta de conhecimento em bases de dados –, outros consideram que a mineração de dados representa a etapa central desse processo maior denominado *KDD*. As outras etapas tratam, basicamente,

do pré-processamento dos dados (seleção, limpeza e transformação) e pós-processamento da informação minerada (visualização e análise).

Os problemas tratados em mineração de dados são resolvidos por dois grandes grupos de soluções ou tarefas:

- Tarefas descritivas: têm como objetivo encontrar padrões que descrevam os dados, permitindo sua análise. As principais tarefas descritivas são: Extração de Regras de Associação e Agrupamento (*Clustering*).
- Tarefas preditivas: realizam inferências sobre os dados existentes para predizer o comportamento de novos dados. As principais tarefas preditivas são: Classificação e Regressão.

Classificação é uma das tarefas mais importantes em mineração de dados, sendo portanto objeto de inúmeras pesquisas. Os resultados apresentados nesta tese concentram-se nessa tarefa e na sua aplicação na área de bioinformática.

Dentre as diversas áreas de aplicação das técnicas de mineração de dados, a bioinformática vem se destacando nas últimas décadas. Essa nova área de estudo, também surgida recentemente, trata do armazenamento, organização, análise, interpretação e utilização de informações provenientes de dados biológicos [92].

A revolução genômica resultou num crescimento exponencial da quantidade de dados biológicos gerados pela comunidade científica. Com a finalidade de armazenar toda essa informação biológica gerada, foram criados os bancos de dados biológicos [10]. A necessidade por ferramentas capazes de realizar análises nesses dados tornou-se evidente, fazendo com que técnicas de mineração de dados começassem a ser empregadas nesse contexto.

Desse modo, diversas técnicas de mineração de dados vêm sendo propostas por pesquisadores para solucionar problemas biológicos [92], tais como: descobrir associações e correlações entre genes, localizar genes específicos em uma seqüência, predizer a estrutura ou a função de uma proteína, agrupar proteínas em famílias de seqüências relacionadas entre si, entre outros.

Motivação e Objetivos do Trabalho

A construção de classificadores precisos e computacionalmente eficientes para bases de dados grandes, em termos de volume e dimensão, é um importante desafio da área de

mineração de dados. O intenso interesse por esse tema fez com que diversas técnicas para construção de classificadores fossem propostas, tais como: árvores de decisão [71], k-NN (*k-Nearest Neighbor*) [20], classificadores Bayesianos [26], redes neurais [39], máquinas de vetor de suporte (*Support Vector Machines* – SVM) [89], entre outras.

O problema de classificação de proteínas (*Protein Classification Problem* – PCP) vem sendo objeto de vários trabalhos de pesquisa na área de bioinformática. Trata-se de uma importante tarefa para a biologia molecular, uma vez que, através da identificação da classe de uma proteína, é possível inferir quais são as suas funções. Algumas técnicas de classificação têm sido utilizadas com o intuito de predizer a função de proteínas a partir de suas composições de motivos (*motifs*) [38, 70, 91, 93]. Uma ferramenta computacional de pré-processamento e análise de dados genéticos, denominada *GenMiner*, foi apresentada em [38]. Nesse trabalho, árvores de decisão foram utilizadas para a classificação das proteínas. No trabalho proposto em [70], desenvolveu-se um algoritmo de indução de regras de classificação a partir de autômatos finitos. As propostas apresentadas em [91, 93] também exploraram a técnica de extração de árvores de decisão com o objetivo de extrair regras para classificar as proteínas. Apesar de esses trabalhos apresentarem resultados promissores para a tarefa de classificação de proteínas, evidenciando que as técnicas utilizadas foram adequadas para resolução do problema em questão, somente em [70] realizaram-se experimentos englobando um grande conjunto de proteínas (aproximadamente 40000) armazenado no banco de dados biológico adotado – o Prosite [45]. E para esse caso, o método proposto atingiu uma acurácia de somente 41,4%.

Esse fato motivou a proposta inicial deste trabalho, cujo objetivo é apresentar um método de classificação de proteínas computacionalmente eficiente e capaz de alcançar altas taxas de acurácia, superando resultados apresentados anteriormente, mesmo para bases de dados contendo um grande número de proteínas.

Por realizar a classificação tomando como base probabilidades associadas a subconjuntos de valores de atributos que caracterizam as proteínas, o primeiro método proposto nesta tese foi denominado HiSP-Prot (***Hi****ghest* ***S****ubset* ***P****robability - for* ***Prot****ein Classification Problem*).

No HiSP-Prot, a classificação de instâncias é baseada em análise de padrões. Considerando que cada instância é descrita por um conjunto de atributos, um padrão corresponde a uma combinação de valores desses atributos que caracteriza uma determinada classe. A idéia central do método é classificar uma instância descobrindo-se quais subconjuntos dos seus valores de atributos melhor caracterizam alguma classe. Desse modo, o classificador

atribui a uma nova instância a classe que melhor é descrita pelos seus subconjuntos de valores de atributos.

As técnicas apresentadas em [38] e [70] para resolver o PCP foram escolhidas para uma avaliação comparativa com o HiSP-Prot, uma vez que essas abordagens apresentaram os melhores resultados experimentais. Os bons resultados obtidos na avaliação realizada nesta tese mostraram o potencial do método proposto para a resolução do PCP e motivaram a proposta de uma generalização do mesmo.

Portanto, o segundo objetivo desta tese é estender o método proposto inicialmente para o PCP (HiSP-Prot), tornando-o adequado e eficiente para diferentes tipos de bases de dados pertencentes a aplicações distintas. A generalização do HiSP-Prot foi denominada simplesmente HiSP (***Hi****ghest* ***S****ubset* ***P****robability*).

Para avaliar e validar o método HiSP a partir de bases de dados com características distintas e pertencentes a diferentes aplicações, 40 bases de dados de domínio público, freqüentemente utilizadas em experimentos de mineração de dados, foram selecionadas no repositório de dados *UCI Machine Learning Repository* [12]. A partir dessas bases, resultados que confirmam a eficiência do HiSP são obtidos ao compará-lo com importantes técnicas de classificação, tais como: árvores de decisão, k-NN, classificação Bayesiana simples e classificação associativa.

O restante desta tese está organizado como especificado a seguir. O Capítulo 2 contém uma definição do processo de classificação e uma revisão bibliográfica sobre o tema. O Capítulo 3 apresenta o problema de classificação de proteínas, os trabalhos relacionados, o método proposto para solucionar esse problema, os experimentos computacionais realizados e a análise dos resultados obtidos. A generalização do método proposto para o problema de classificação de proteínas, com o objetivo de torná-lo adequado e eficiente para outras aplicações, é descrita no Capítulo 4. O Capítulo 5 relata os resultados obtidos com o método generalizado para experimentos realizados com as 40 bases de dados utilizadas. Por fim, o Capítulo 6 apresenta as conclusões deste trabalho e propostas de trabalhos futuros.

Capítulo 2

Técnicas e Algoritmos de Classificação

2.1 Introdução

Tradicionalmente, a literatura da área de mineração de dados apresenta o processo de *KDD (Knowledge Discovery in Database)* dividido em seis etapas, as quais englobam duas grandes fases: a preparação dos dados e a sua mineração.

O processo de *KDD* começa com a compreensão do domínio da aplicação e o estabelecimento de objetivos a serem alcançados. A próxima etapa corresponde a uma preparação dos dados. Nessa fase, os dados são pré-processados, ou seja, passam por transformações com o objetivo de ficarem adequados ao uso pelas técnicas de mineração de dados. Posteriormente, chega-se à fase de mineração dos dados.

Uma das tarefas mais importantes em mineração de dados, que tem se mostrado útil em diversas áreas de aplicação, é a classificação. Ela corresponde a uma forma de análise de dados cujo objetivo é construir modelos, a partir de um conjunto de instâncias com características e classes conhecidas, capazes de classificar novas instâncias a partir de suas características. Essa tarefa é considerada preditiva, uma vez que são realizadas inferências sobre dados para se predizer a classe de uma nova instância. Vários métodos de classificação vêm sendo propostos por pesquisadores das áreas de aprendizado de máquina, estatística e mineração de dados.

Neste capítulo, uma visão geral do processo de classificação é apresentada na Seção 2.2. As Seções 2.3 e 2.4 discutem técnicas relacionadas com a avaliação dos métodos de classificação. Em seguida, algumas questões relacionadas com o pré-processamento dos dados são abordadas na Seção 2.5. Por fim, na Seção 2.6, são apresentados métodos e algoritmos comumente utilizados pela comunidade científica para a tarefa de classificação.

2.2 O Processo de Classificação

O processo de classificação pode ser dividido em duas etapas. Na primeira etapa, conhecida como treinamento, o objetivo é construir modelos que caracterizem as classes a partir dos valores dos atributos das instâncias da base de dados, ou seja, modelos capazes de realizar o mapeamento entre valores de atributos de uma instância e uma determinada classe. A construção desse modelo é realizada por meio de análise de instâncias contidas numa base de dados, as quais são representadas pelos seus atributos. Cada instância pertence a uma classe, definida por um dos atributos, denominado atributo classe. As instâncias analisadas para a construção do modelo constituem uma base de dados conhecida como base de dados de treinamento.

Na segunda etapa, o modelo construído é avaliado quanto à sua acurácia preditiva. Se esse teste de acurácia produzir resultados aceitáveis, o modelo poderá ser utilizado na classificação de novas instâncias, ou seja, instâncias cujas classes são desconhecidas. A estimativa da acurácia preditiva do modelo é realizada utilizando-se um conjunto de instâncias cujas classes são conhecidas. Desse modo, por meio de análise comparativa, pode-se calcular o percentual de instâncias corretamente classificadas, ou seja, a precisão (ou acurácia) do modelo para o conjunto de instâncias em questão. Se a acurácia do modelo for estimada a partir das mesmas instâncias utilizadas para sua construção, provavelmente, uma estimativa otimista será obtida. Sendo assim, o mais indicado é utilizar instâncias que não fizeram parte do conjunto de treinamento. Essas instâncias constituem a base de dados de teste. Portanto, um procedimento comum antes do início do processo de classificação é a divisão da base de dados inicial (que contém instâncias cujas classes são conhecidas) em base de dados de treinamento e de teste. A Seção 2.3 descreve dois métodos comumente utilizados na estimativa da acurácia de modelos de classificação.

2.3 Estimativa da Acurácia dos Modelos de Classificação

A avaliação da acurácia dos modelos de classificação é importante por permitir uma estimativa da precisão do modelo ao classificar dados futuros, ou seja, dados que não foram utilizados no processo de construção do modelo. Além disso, essa estimativa permite a comparação de desempenho entre diferentes classificadores sobre um mesma base de dados de teste.

Segundo [37], *holdout* e validação cruzada (*cross validation*) são dois métodos comumente utilizados na avaliação da acurácia de um classificador. Nos dois métodos, a base de dados inicial é particionada para gerar as bases de dados de treinamento e de teste.

No método *holdout* a base de dados inicial é aleatoriamente dividida nas bases de treinamento e de teste. Geralmente, dois terços dos dados da base inicial são utilizados como base de dados de treinamento, e o restante, como base de dados de teste [51]. Para evitar que a estimativa da acurácia seja influenciada por uma possível particularidade da partição que compõe a base de dados de teste, na prática, é comum aplicar o método *holdout* k vezes, gerando assim k pares (treinamento e teste) de bases de dados distintos. Desse modo, a acurácia é obtida a partir da média dos percentuais de acerto obtidos a partir de cada um dos k testes.

Na k-validação cruzada (k-*fold cross validation*) [51], a base de dados inicial é aleatoriamente dividida em k partições de mesmo tamanho (ou aproximadamente do mesmo tamanho). A acurácia corresponde à média dos percentuais de acerto de k iterações. Em cada iteração, cada uma das k partições geradas forma a base de dados de teste e as $k-1$ partições restantes, a base de dados de treinamento.

Segundo [95], uma grande quantidade de testes sobre diversas bases de dados têm mostrado que o valor de k igual a dez é o mais adequado para se obter uma boa estimativa da precisão do classificador. Portanto, na prática, a validação cruzada com k igual a dez vem sendo o método mais utilizado. No entanto, uma única realização da 10-validação cruzada pode não ser suficiente para obtenção de uma boa estimativa da acurácia do modelo. Isso pode acontecer devido à aleatoriedade existente na escolha das k partições. Sendo assim, um procedimento geralmente realizado para tentar melhorar a estimativa da acurácia é a repetição da 10-validação cruzada dez vezes. A acurácia será o resultado médio dessas dez execuções.

Apesar de o método 10-validação cruzada ser o mais comumente utilizado, outros dois métodos são freqüentemente citados na literatura, o *leave-one-out* [85] e o *bootstrap* [28]. O *leave-one-out* é simplesmente uma n-validação cruzada, onde n é o número de instâncias da base de dados. Já o *bootstrap* corresponde a um procedimento estatístico de amostragem com reposição. Nesse caso, a base de dados de treinamento é formada por meio de amostragem com reposição na base de dados inicial, e a base de teste, pelos elementos não selecionados para a composição da base de dados de treinamento.

2.4 Avaliação dos Métodos de Classificação

Tanto na avaliação de um método de classificação específico quanto na comparação entre diferentes métodos, os seguintes critérios podem ser levados em consideração [37]:

- Acurácia preditiva: é a habilidade que o modelo possui de predizer corretamente a classe de uma instância desconhecida.
- Desempenho: corresponde aos custos computacionais envolvidos na geração e aplicação do modelo.
- Robustez: é a habilidade do modelo de classificar corretamente instâncias a partir de dados com ruídos e com valores de atributos desconhecidos.
- Escalabilidade: refere-se à construção eficiente de modelos a partir de grandes quantidades de dados.
- Interpretabilidade: diz respeito ao grau de compreensibilidade proporcionado pelo modelo ao usuário da aplicação.

2.5 Pré-Processamento dos Dados

Atualmente, é muito comum existirem bases de dados contendo ruídos, dados inconsistentes e instâncias com valores de atributos desconhecidos. Desse modo, uma fase de preparação dos dados pode ser utilizada com o intuito de melhorar a qualidade dos mesmos. Além disso, transformações podem ser necessárias para adequar os dados ao uso por alguma técnica de mineração de dados específica.

Diversas técnicas de pré-processamento podem ser utilizadas na fase de preparação dos dados. Na Seção 2.5.1, são apresentados alguns métodos de discretização de atributos. Em seguida, na Seção 2.5.2, são mostradas algumas alternativas propostas para lidar com os valores desconhecidos de atributos. Por fim, o problema de seleção de atributos é discutido na Seção 2.5.3.

2.5.1 Discretização de Atributos

Diversos algoritmos de mineração de dados trabalham com bases de dados contendo somente atributos discretos. No entanto, na prática, algumas bases de dados possuem

também atributos contínuos. Nesses casos, para que tais algoritmos possam ser aplicados, faz-se necessária uma etapa de pré-processamento dos dados para a discretização dos atributos contínuos.

Os métodos de discretização de atributos podem ser divididos em: supervisionados e não supervisionados [25]. Enquanto os métodos supervisionados utilizam informações referentes às classes das instâncias da base de dados durante o processo de discretização de um atributo, os não supervisionados consideram somente os valores do atributo a ser discretizado.

O método de discretização mais simples, denominado Partição em Intervalos Iguais (*Equal Interval Width*) [25], divide a faixa de valores de um atributo em k intervalos iguais (de mesma amplitude), atribuindo a cada intervalo um rótulo. Outro método de discretização semelhante, conhecido como Partição em Intervalos com Freqüências Iguais (*Equal Frequency Interval*) [25], divide os valores de um atributo contínuo em k partições, de modo que, considerando m instâncias na base de dados, cada partição deve conter m/k valores adjacentes (possivelmente duplicados). O parâmetro k deve ser informado pelo usuário dos métodos. Esses dois métodos de discretização são não supervisionados. Desse modo, as fronteiras escolhidas para particionar os dados em intervalos podem colocar juntas muitas instâncias pertencentes a diferentes classes, afetando a precisão do classificador.

Na classe dos métodos supervisionados, várias propostas são encontradas na literatura. Em [43], é apresentado um algoritmo de discretização denominado 1R, cujo objetivo é dividir o domínio de cada atributo contínuo em partições "puras", ou seja, partições que sejam caracterizadas predominantemente por uma das classes da base de dados.

Alguns outros métodos supervisionados, tais como *ChiMerge* [49] e *StatDisc* [77], utilizam testes estatísticos para determinar as partições no processo de discretização.

Há também um grupo de métodos supervisionados que realiza a discretização utilizando o conceito de entropia [25]. Entre esses, tem-se uma heurística recursiva de minimização de entropia, apresentada em [30] e [18]. Nessa heurística, a proposta é determinar um ponto de corte que divida um intervalo de valores em dois subintervalos. Esse ponto de corte, definido a partir da medida de Ganho de Informação [37], será aquele que resultar em subintervalos mais "puros" possíveis (com maior ganho de informação). Enquanto um determinado critério de parada não for alcançado, o procedimento é recursivamente aplicado aos subintervalos resultantes das iterações anteriores.

Dado um conjunto de instâncias S, a discretização de um atributo A, pela heurística recursiva de minimização de entropia, é realizada da seguinte maneira:

1. Cada valor do atributo A é considerado como um possível limite de intervalo, ou seja, como um ponto de corte T. Por exemplo, um valor $v \in A$ pode particionar as instâncias de S em dois subconjuntos satisfazendo as condições $A < v$ e $A \geq v$.

2. Dado S, o ponto de corte escolhido será aquele que maximiza o ganho de informação resultante daquele particionamento. O ganho de informação (GI) é dado por:

$$GI(A, T; S) = E(S) - E(A, T; S), \tag{2.1}$$

onde $E(S)$ é a entropia do conjunto S e $E(A, T; S)$ é a entropia do conjunto S depois de o atributo A ter sido particionado em T. A entropia de um determinado conjunto de instâncias é calculada considerando-se a distribuição do atributo classe entre as instâncias desse conjunto. Ou seja, dado um conjunto de instâncias S associadas a m classes, a entropia $E(S)$ é calculada da seguinte forma:

$$E(S) = -\sum_{i=1}^{m} p_i log_2(p_i), \tag{2.2}$$

onde p_i é a probabilidade da classe i ocorrer em S, calculada dividindo-se o número de instâncias da classe i em S pelo número total de instâncias de S.

A entropia $E(A, T; S)$ é dada por:

$$E(A, T; S) = \frac{|S_1|}{|S|} E(S_1) + \frac{|S_2|}{|S|} E(S_2), \tag{2.3}$$

onde S_1 e S_2 são os subintervalos de S gerados pelo ponto de corte T, e $E(S_1)$ e $E(S_2)$ são as entropias dos conjuntos S_1 e S_2, respectivamente, cujos cálculos são semelhantes ao apresentado na Equação 2.2.

3. O particionamento recursivo do conjunto S continuará enquanto o critério de parada não for alcançado. Na heurística proposta em [30] e [18], o critério de parada adotado foi o *Minimal Description Length Principle* [78], no qual o particionamento recursivo do conjunto de instâncias S é finalizado se e somente se

$$GI(A, T; S) < \frac{log_2(N-1)}{N} + \frac{\Delta(A, T; S)}{N}, \tag{2.4}$$

onde N é o número de instâncias do conjunto S e

$$\Delta(A, T; S) = log_2(3^k - 2) - [kE(S) - k_1 E(S_1) - k_2 E(S_2)], \tag{2.5}$$

sendo k_i o número de classes presentes no conjunto de instâncias S_i.

2.5.2 Valores Desconhecidos de Atributos

Bases de dados reais, por razões diversas, contêm instâncias que possuem valores desconhecidos de atributos. Portanto, simplesmente considerar que os valores desconhecidos de atributos não correspondem a uma informação importante para a etapa de mineração de dados, pode não ser a melhor alternativa. A decisão de como tratar os valores desconhecidos de atributos numa tarefa de mineração de dados deve levar em consideração o significado dessa ausência de valores.

A seguir, são apresentadas algumas alternativas propostas em [37] para lidar com os valores desconhecidos de atributos:

- Ignorar a instância: essa alternativa é comumente adotada quando o atributo classe possui valor desconhecido (assumindo que a tarefa de mineração envolvida seja a classificação). Fora isso, essa estratégia só é recomendada se a instância possuir muitos atributos cujos valores sejam desconhecidos.
- Preencher os valores desconhecidos manualmente: geralmente, essa abordagem é muito trabalhosa e inviável para grandes bases de dados.
- Adotar uma constante global para preencher os valores desconhecidos: nesse caso, todos os valores desconhecidos são substituidos pela mesma constante, diferente dos valores dos domínios dos atributos, que indicará o desconhecimento do valor.
- Utilizar o valor médio (ou a moda) do atributo para preencher os valores desconhecidos.
- Para cada classe da base de dados, utilizar o valor médio (ou a moda) do atributo para preencher todos os valores desconhecidos das instâncias da classe em questão.
- Usar o valor mais provável para preencher os valores desconhecidos: o valor mais provável pode ser determinado por meio de regressão, inferência Bayesiana ou árvore de decisão.

2.5.3 Seleção de Atributos

A qualidade dos dados está entre os diversos fatores que afetam o desempenho dos algoritmos de mineração de dados. Ela está relacionada com os graus de relevância e redundância

dos dados, com a quantidade de ruídos existentes e com a veracidade dos dados.

A seleção de atributos é um processo que visa identificar e remover a maior quantidade possível de informações irrelevantes e redundantes. Em alguns casos, o próprio algoritmo de mineração de dados, como é o caso dos algoritmos de indução de árvores de decisão, já realiza algum processamento envolvendo seleção de atributos. Já para vários outros métodos, a seleção de atributos não faz parte de seu processamento.

Como uma etapa de pré-processamento dos dados, uma abordagem comumente utilizada para a seleção de atributos considera cada atributo individualmente, ordenando-os de acordo com as suas capacidades preditivas e selecionando os melhores atributos para compor o subconjunto que será utilizado pelo algoritmo de mineração de dados. A técnica *Information Gain Attribute Ranking* é um exemplo desse tipo de abordagem.

Information Gain Attribute Ranking é uma das técnicas de seleção de atributos mais simples, freqüentemente utilizada em aplicações onde a dimensionalidade dos dados proíbe a utilização de técnicas mais sofisticadas. Seja A um atributo de uma base de dados e C o seu conjunto de classes. O cálculo da entropia do atributo classe antes e depois de observado o atributo A é dado pelas equações 2.6 e 2.7, respectivamente.

$$E(C) = -\sum_{c \in C} p(c) log_2 p(c), \tag{2.6}$$

onde $p(c)$ é a probabilidade da classe c ocorrer na base de dados.

$$E(C|A) = -\sum_{a \in A} p(a) \sum_{c \in C} p(c|a) log_2 p(c|a), \tag{2.7}$$

onde $p(a)$ é a probabilidade do valor a ocorrer na base de dados e $p(c|a)$ a probabilidade da classe c ocorrer, dado que o valor de atributo a ocorreu.

A redução causada na entropia de C devido à informação adicional fornecida pelo atributo A é denominada Ganho de Informação (*Information Gain*). Na técnica *Information Gain Attribute Ranking* cada atributo A_i da base de dados é associado a um valor correspondente ao ganho de informação (GI), calculado da seguinte maneira:

$$GI_i = E(C) - E(C|A_i). \tag{2.8}$$

Desse modo, os atributos associados aos maiores ganhos de informação serão os selecionados.

Independentemente do método que esteja sendo utilizado, a seleção de atributos,

como uma etapa de pré-processamento dos dados, pode ser muito benéfica. A redução da dimensionalidade dos dados permite que os algoritmos de mineração de dados trabalhem com mais rapidez e maior eficiência. Em alguns casos, devido à redução de ruídos, de dados redundantes ou de dados irrelevantes, a seleção de atributos proporciona também uma melhoria na qualidade do classificador. Além disso, ganhos podem ser obtidos com o aumento da compreensibilidade do modelo construído.

Além das técnicas de seleção de atributos cuja abordagem considera cada atributo individualmente, existem outras que avaliam os subconjuntos de atributos com o objetivo de encontrar aquele que maximiza a acurácia preditiva do classificador. No entanto, para uma base de dados com n atributos, tem-se 2^n subconjuntos de atributos possíveis. Portanto, realizar uma busca exaustiva nesse espaço de soluções pode ser computacionalmente intratável. Desse modo, métodos heurísticos, que exploram um espaço de soluções reduzido, são comumente utilizados na tarefa de selecionar um subconjunto de atributos. Tais métodos são tipicamente gulosos no sentido de que a busca pelo espaço de soluções é sempre conduzida pela melhor escolha a cada momento. Na prática, esses métodos são geralmente muito eficientes.

A Figura 2.1 mostra um grafo que representa todos os subconjuntos de atributos possíveis para um problema contendo quatro atributos. Os nós desse grafo são rotulados com seqüências de zeros e uns, que codificam os subconjuntos de atributos. Como o problema considerado nesse exemplo possui quatro atributos, os nós serão rotulados com quatro *bits*. O valor zero significa a ausência de um atributo e o valor 1, a presença do mesmo. Além disso, cada nó está conectado a outros nós que possuem um atributo a mais ou um atributo a menos do que ele.

Segundo os autores do trabalho proposto em [14], a natureza dos métodos heurísticos de seleção de atributos é determinada por quatro itens básicos:

1. Ponto de partida.
2. A estratégia que será utilizada para percorrer o espaço de soluções.
3. A forma de avaliação dos subconjuntos de atributos.
4. Critério de parada.

Primeiramente, deve-se determinar o ponto (ou pontos) de partida no espaço de soluções. O ponto de partida influencia diretamente na escolha dos operadores que serão utilizados para percorrer o espaço de soluções. Pode-se, por exemplo, começar com o

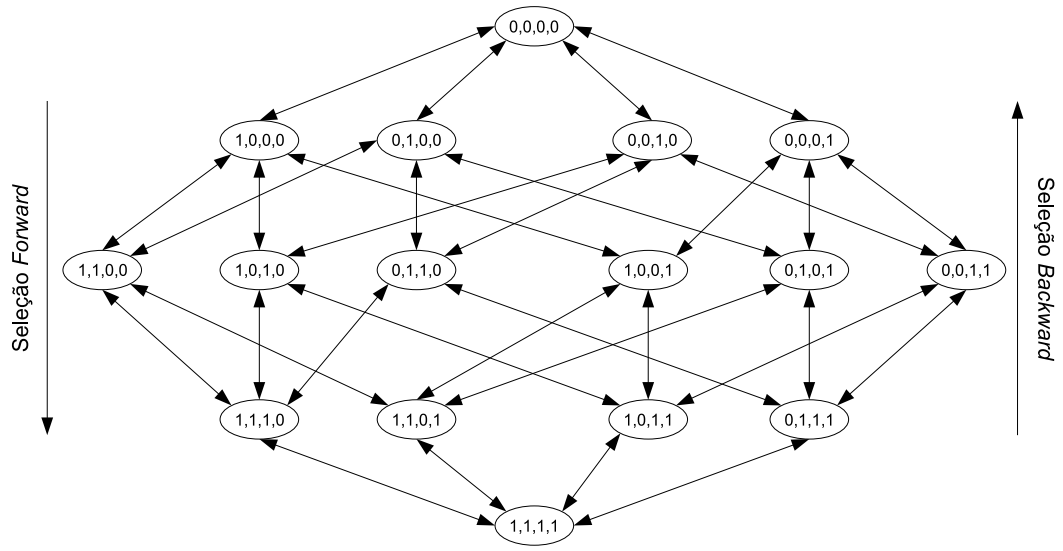


Figura 2.1: Espaço de soluções para um problema contendo quatro atributos [52].

conjunto vazio de atributos e ir sucessivamente adicionando atributos, ou começar com o conjunto completo de atributos para depois realizar uma sucessão de remoções dos mesmos. A primeira abordagem é conhecida como seleção *forward* e a última, como seleção *backward*. A Figura 2.1 apresenta a direção percorrida no espaço de soluções para cada uma das abordagens citadas anteriormente.

O próximo passo consiste em determinar qual estratégia será utilizada para se percorrer o espaço de soluções. Duas estratégias comumente utilizadas nos trabalhos de seleção de atributos são a *hill-climbing* e a *best-first* [52].

Na estratégia *hill-climbing*, também conhecida como seleção *stepwise*, em cada iteração, expande-se a solução corrente gerando-se várias soluções vizinhas, e move-se para a melhor solução gerada. O processo é interrompido quando nenhuma das soluções geradas na expansão for melhor do que a solução corrente.

A estratégia *best-first* é mais robusta do que a *hill-climbing*. Nessa estratégia, em cada iteração, expande-se a solução corrente gerando-se várias soluções vizinhas, e move-se para a melhor solução gerada até o momento, que ainda não tenha sido expandida. Nesse caso, o critério de parada pode ser dado pelo número de iterações seguidas sem alcançar melhorias. A principal diferença dessa estratégia com relação à *hill-climbing* está no fato de ela permitir que soluções geradas em iterações anteriores, que ainda não tenham sido expandidas, se tornem a solução corrente do processo de busca.

De acordo com a abordagem utilizada na avaliação dos atributos, as técnicas de seleção

de atributos podem ser classificadas como dos tipos filtro ou *wrapper* [68]. No caso dos *wrappers*, a avaliação dos atributos é realizada por um algoritmo de mineração de dados específico, ou seja, a precisão alcançada pelo algoritmo é que indica a qualidade de cada subconjunto de atributos. Por outro lado, os filtros operam independentemente de qualquer algoritmo de mineração de dados, selecionando atributos apenas utilizando as características gerais dos dados, tal como a capacidade de discriminação dos atributos com relação às classes.

Enquanto as técnicas *Information Gain Attribute Ranking* [27, 96] e *Relief* [50, 53] avaliam os atributos individualmente, os métodos *Correlation-based Feature Selection* (CFS) [34, 35], *Consistency-based Feature Selection* [59] e *Wrapper Subset Selection* [52] avaliam os subconjuntos de atributos [36].

No método CFS (*Correlation-based Feature Selection*), a avaliação dos subconjuntos leva em consideração a capacidade de discriminação dos atributos com relação às classes e o grau de correlação entre eles.

Quanto maior for a correlação dos atributos de um subconjunto com o atributo classe, e menor a correlação entre eles, maior é a pontuação atribuída ao subconjunto, a qual é calculada da seguinte forma:

$$R_S = \frac{k\overline{r_{ac}}}{\sqrt{k + k(k-1)\overline{r_{aa}}}}, \tag{2.9}$$

onde R_S é denominada medida da "recompensa" de um subconjunto S contendo k atributos, $\overline{r_{ac}}$ é a correlação média entre os atributos e a classe, e $\overline{r_{aa}}$ a correlação média entre os atributos. O numerador da Equação 2.9 indica a capacidade preditiva do subconjunto de atributos, enquanto o denominador mede o grau de redundância entre os atributos.

Através de alguma das estratégias de busca mencionadas anteriormente, o CFS percorre o espaço de soluções tentando encontrar um bom subconjunto de atributos, avaliando-os segundo a Equação 2.9.

A técnica *Consistency-based Feature Selection* utiliza a consistência como medida de avaliação dos subconjuntos de atributos. Essa técnica busca por combinações de atributos cujos valores dividam os dados em subconjuntos associados a uma classe majoritária. Normalmente, a busca privilegia subconjuntos de atributos menores com alta consistência. A medida de consistência, proposta em [59], é dada pela Equação 2.10.

$$Consistência_S = 1 - \frac{\sum_{i=0}^{J} |D_i| - |M_i|}{N}, \tag{2.10}$$

onde S é um subconjunto de atributos, J é o número de combinações distintas de valores dos atributos de S, $|D_i|$ é o número de ocorrências da i-ésima combinação de valores de atributos, $|M_i|$ é a cardinalidade da classe majoritária para a i-ésima combinação de valores de atributos e N é o número total de instâncias da base de dados.

Adotando alguma das estratégias de busca mencionadas anteriormente, a técnica *Consistency-based Feature Selection* percorre o espaço de soluções em busca de um bom subconjunto de atributos, cuja avaliação é feita segundo a Equação 2.10.

2.6 Métodos e Algoritmos de Classificação

Conforme mencionado na Seção 2.2, o processo de classificação envolve a geração de um modelo. A construção desse modelo é realizada por um algoritmo de aprendizado, ou indutor, a partir de uma análise sobre um conjunto de dados de treinamento.

Existem dois tipos de algoritmos de aprendizado, denominados *lazy* e *eager*. Nos algoritmos *lazy*, o processamento dos dados só acontece quando existe uma instância para ser classificada. O termo *lazy*, expressando a idéia de preguiçoso, deve-se ao fato de esses algoritmos adiarem o processamento dos dados até que uma requisição seja realizada. Contrariamente, os algoritmos *eager* (gulosos) utilizam as instâncias de treinamento para induzir um modelo de classificação e descartam-nas logo que o processamento dos dados se encerra. Nesse caso, a classificação de novas instâncias é realizada por meio do modelo de classificação previamente construído.

Vários métodos e algoritmos de classificação vêm sendo propostos por pesquisadores das áreas de aprendizado de máquina, estatística e mineração de dados. Nas seções a seguir, métodos comumente utilizados pela comunidade científica para a tarefa de classificação, tais como, árvores de decisão, classificadores Bayesianos, classificadores baseados em conceitos de regras de associação, k-NN, máquinas de vetor de suporte e redes neurais, serão apresentados.

2.6.1 Classificação por Indução de Árvores de Decisão

Árvore de decisão é um modelo simbólico cuja estrutura apresenta-se no formato de uma árvore, onde cada nó interno indica um teste sobre um atributo, cada ramo representa um resultado do teste, e os nós terminais correspondem às classes ou distribuições de classes. O nó localizado no topo da árvore é denominado nó raiz. A Figura 2.2 ilustra uma árvore

de decisão para avaliação de comportamento de consumidores com relação à compra de um determinado produto. Os nós internos e o nó raiz são representados por retângulos. Cada retângulo corresponde a um atributo teste. Já os nós terminais são representados por elipses, cada um correspondendo a uma classe.

Para classificar uma instância desconhecida, os valores de seus atributos são testados nos nós ao longo da árvore, formando um caminho do nó raiz até um nó terminal, o qual conterá a predição da classe dessa instância.

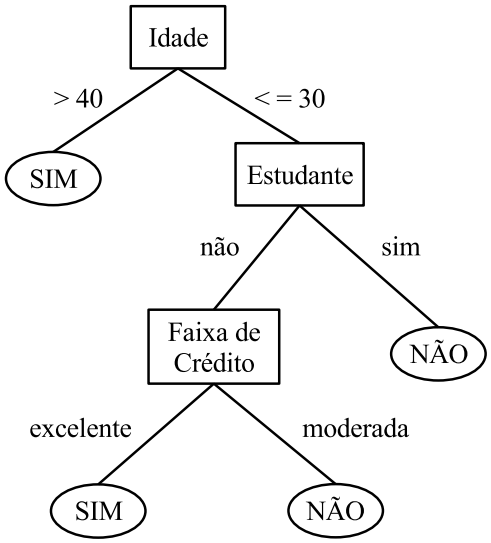


Figura 2.2: Árvore de decisão simples para avaliação de comportamento de consumidores.

2.6.1.1 Indução de Árvores de Decisão

A tarefa de construção de uma árvore de decisão é chamada de indução. A maioria dos algoritmos de indução de árvores de decisão corresponde a um procedimento guloso que recursivamente constrói a árvore de cima para baixo, ou seja, do nó raiz em direção aos nós terminais. Em cada iteração, a partir da base de dados de treinamento, os algoritmos procuram pelo atributo que melhor separa as classes para realizarem a ramificação da árvore, e recursivamente processam os subproblemas resultantes das ramificações.

Essa abordagem de divisão e conquista adotada pelos algoritmos de indução de árvores de decisão foi desenvolvida e refinada ao longo de vários anos por John Ross Quinlan. Sua contribuição inicial foi o algoritmo ID3 [71]. Várias melhorias foram realizadas nesse algoritmo, culminando no surgimento do algoritmo C4.5 [73]. Posteriormente surgiu o C5.0, uma versão comercial do C4.5.

A Figura 2.3 apresenta o pseudocódigo do algoritmo ID3. Como entrada de dados têm-se um conjunto de instâncias de treinamento (*instâncias*) e um conjunto de atributos candidatos (*lista de atributos*) dessas instâncias. A árvore de decisão começa com um único nó representando todas as instâncias de treinamento (linha 1). Se todas as instâncias forem da mesma classe, esse nó, denominado nó terminal, será rotulado com a classe dessas instâncias (linhas 2 a 4). Senão, se a *lista de atributos* estiver vazia, o nó terminal será rotulado com a classe mais freqüente em *instâncias* (linhas 5 a 7). Caso as duas condições anteriores sejam falsas, o algoritmo utiliza a medida denominada Ganho de Informação para selecionar, entre os atributos de *lista de atributos*, aquele que melhor separa as instâncias em classes individuais (linha 8). Esse atributo se torna o atributo teste do nó corrente (linha 9). A partir dele, para cada um de seus valores, uma ramificação é criada e as instâncias são particionadas (linhas 11 e 12). Todo esse procedimento é recursivamente aplicado para formar uma árvore de decisão a partir das instâncias de cada ramificação. A base do procedimento recursivo é atingida quando uma das seguintes condições for verdadeira: (a) todas as instâncias em um determinado nó pertecem à mesma classe (linhas 2 a 4), (b) não existem mais atributos para dar prosseguimento ao processo de particionamento (linhas 5 a 7), ou (c) não existem mais instâncias em uma ramificação de um atributo teste (linhas 13 e 14).

Nos algoritmos de indução de árvores de decisão, para cada nó a ser inserido na árvore, um atributo teste precisa ser escolhido. Se essa escolha fosse aleatória, dificilmente, a árvore formada possuiria nós terminais contendo instâncias sempre da mesma classe, ou com a maioria das instâncias pertencendo à mesma classe. Desse modo, a escolha do melhor atributo teste em cada nó é geralmente feita por meio de uma medida de avaliação. Várias medidas foram propostas por pesquisadores, sendo que, a maioria delas baseia-se no princípio da minimização da entropia. O objetivo é escolher o atributo que gere partições (resultantes das ramificações) com o menor número possível de classes distintas em cada uma delas.

No algoritmo ID3, a escolha do atributo teste é feita utilizando-se uma medida conhecida como Ganho de Informação. O atributo que proporcionar o maior ganho de informação (ou maior redução de entropia) é selecionado como atributo teste do nó corrente. Essa medida de seleção de atributo garante a construção de uma árvore mais simples e minimiza o número de testes necessários para classificar uma instância.

O exemplo a seguir ilustra o processo de escolha de um atributo teste. Seja a base de dados de treinamento representada na Tabela 2.1. No caso desse exemplo, o atributo

procedimento CRIA-ÁRVORE(*instâncias*, *lista de atributos*)
1: Crie o nó N;
2: **se** (*instâncias* são todos da mesma classe, C) **então**
3: **Retorne** N como sendo um nó terminal rotulado com a classe C;
4: **fim se**
5: **se** (*lista de atributos* está vazia) **então**
6: **Retorne** N como sendo um nó terminal rotulado com a classe mais freqüente em *instâncias*;
7: **fim se**
8: Selecione, como *atributo teste*, o atributo da *lista de atributos* que possui o maior ganho de informação;
9: Rotule o nó N com o *atributo teste*;
10: **para** cada valor a_i do *atributo teste* **faça**
11: Crie uma ramificação a partir do nó N para *atributo teste* $= a_i$;
12: Seja s_i o conjunto de instâncias de *instâncias* para as quais *atributo teste* $= a_i$;
13: **se** (s_i está vazio) **então**
14: Coloque um nó terminal com a classe mais freqüente em *instâncias*;
15: **senão**
16: Coloque o nó retornado por CRIA-ÁRVORE(s_i, *lista de atributos* de s_i)
17: **fim se**
18: **fim para**
fim.

Figura 2.3: Pseudocódigo do ID3.

dependente, *Comprou*, tem dois valores possíveis, *sim* e *não*. Sendo assim, temos duas classes distintas ($m = 2$), C_1 correspondendo ao valor *sim* e C_2, ao valor *não*.

O procedimento a seguir mostra como é realizada a escolha do atributo teste para um determinado nó da árvore. Dado um conjunto de treinamento S, inicialmente deve-se calcular a informação necessária para se classificar uma determinada instância de S, a qual é dada por:

$$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} p_i \ log_2(p_i) \tag{2.11}$$

onde, p_i é a probabilidade de uma instância pertencer à classe C_i. Sendo S o conjunto de treinamento em consideração e s o número de instâncias contidas em S, a probabilidade p_i pode ser estimada por s_i/s, onde s_i é o número de instâncias de treinamento que pertencem à classe C_i. Para o caso do exemplo em questão, onde está sendo feita a escolha do atributo teste para o nó raiz, todas as instâncias da base de dados de treinamento são levadas em consideração, ou seja, $s = 14$. Além disso, pode-se verificar que dessas 14 instâncias, 9 (s_1) pertencem à classe *sim* e 5 (s_2) pertencem à classe *não*. Desse modo, a partir da

Tabela 2.1: Base de dados de treinamento (*)

Idade	Salário	Estudante	Faixa de Crédito	Comprou
<=30	alto	não	moderada	não
<=30	alto	não	excelente	não
31...40	alto	não	moderada	sim
> 40	médio	não	moderada	sim
> 40	baixo	sim	moderada	sim
> 40	baixo	sim	excelente	não
31...40	baixo	sim	excelente	sim
<=30	médio	não	moderada	não
<=30	baixo	sim	moderada	sim
> 40	médio	sim	moderada	sim
<=30	médio	sim	excelente	sim
31...40	médio	não	excelente	sim
31...40	alto	sim	moderada	sim
> 40	médio	não	excelente	não

(*) Dados extraídos de [37].

Equação 2.11 tem-se:

$$I(s_1, s_2) = -\frac{9}{14}\ log_2\frac{9}{14} - \frac{5}{14}\ log_2\frac{5}{14} = 0,940.$$

.

O próximo passo é calcular os valores de entropia de S, considerando o particionamento das instâncias a partir de cada um dos possíveis atributos. Para tal, considere que um atributo A possui v valores distintos, $\{a_1, a_2, \ldots, a_v\}$. Sendo assim, o atributo A pode ser utilizado para particionar as instâncias de treinamento em v subconjuntos, $\{S_1, S_2, \ldots, S_v\}$, onde S_j contém as instâncias de S cujo atributo A possui valor a_j. Sendo s_{ij} o número de instâncias da classe C_i no subconjunto S_j, a entropia de S considerando o particionamento a partir do atributo A é dada por:

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s}\ I(s_{1j}, \ldots, s_{mj}). \tag{2.12}$$

A Tabela 2.2 apresenta o cálculo de entropia considerando os quatro possíveis atributos do exemplo que está sendo resolvido.

Uma vez calculadas as entropias de S considerando os particionamentos a partir de cada um dos atributos, o cálculo do ganho de informação (GI) para o particionamento a

Tabela 2.2: Cálculo de entropia

Atributo	Valor	Classe	s_{ij}	$I(s_{1j}, \ldots, s_{mj})$	Entropia
Idade	<=30	sim	$s_{11} = 2$	$I(s_{11}, s_{21}) = 0,971$	0,694
		não	$s_{21} = 3$		
	31...40	sim	$s_{12} = 4$	$I(s_{12}, s_{22}) = 0$	
		não	$s_{22} = 0$		
	> 40	sim	$s_{13} = 3$	$I(s_{13}, s_{23}) = 0,971$	
		não	$s_{23} = 2$		
Salário	baixo	sim	$s_{11} = 3$	$I(s_{11}, s_{21}) = 0,811$	0,911
		não	$s_{21} = 1$		
	médio	sim	$s_{12} = 4$	$I(s_{12}, s_{22}) = 0,918$	
		não	$s_{22} = 2$		
	alto	sim	$s_{13} = 2$	$I(s_{13}, s_{23}) = 1$	
		não	$s_{23} = 2$		
Estudante	sim	sim	$s_{11} = 6$	$I(s_{11}, s_{21}) = 0,592$	0,789
		não	$s_{21} = 1$		
	não	sim	$s_{12} = 3$	$I(s_{12}, s_{22}) = 0,985$	
		não	$s_{22} = 4$		
Faixa de Crédito	moderada	sim	$s_{11} = 6$	$I(s_{11}, s_{21}) = 0,811$	0,892
		não	$s_{21} = 2$		
	excelente	sim	$s_{12} = 3$	$I(s_{12}, s_{22}) = 1$	
		não	$s_{22} = 3$		

partir de um atributo A é feito por meio da seguinte equação:

$$GI(A) = I(s_1, s_2, \ldots, s_m) - E(A). \tag{2.13}$$

Para os atributos apresentados na Tabela 2.2, têm-se os seguintes valores para a medida de ganho de informação:

$$GI(Idade) = 0,940 - 0,694 = 0,246,$$

$$GI(Salário) = 0,940 - 0,911 = 0,029,$$

$$GI(Estudante) = 0,940 - 0,789 = 0,151,$$

$$GI(Faixa\ de\ Crédito) = 0,940 - 0,892 = 0,048.$$

Como o atributo *Idade* proporciona o maior ganho de informação, ele é selecionado como atributo teste. Desse modo, a árvore de decisão do exemplo começa com o nó raiz rotulado com o atributo *Idade*. A partir desse nó, surgem três ramificações, uma para cada um dos possíveis valores do atributo teste, e as instâncias de treinamento são particionadas de acordo esses valores. A Figura 2.4 ilustra essa ramificação e os três subproblemas que

surgem a partir da mesma. A continuação da construção da árvore de decisão dá-se por

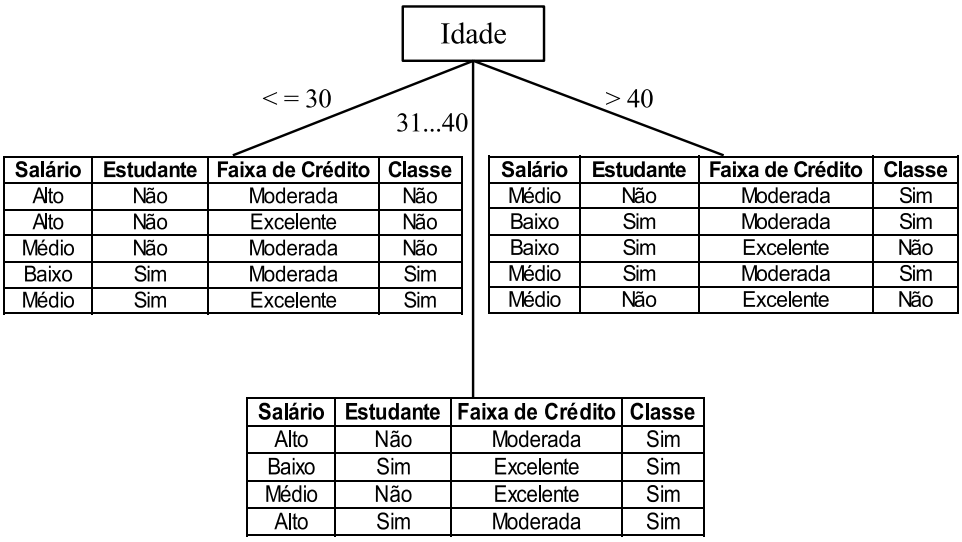


Figura 2.4: Etapa inicial da construção de uma árvore de decisão.

meio da resolução de cada um dos subproblemas que surgem a partir das ramificações, conforme o procedimento recursivo apresentado na Figura 2.3.

2.6.1.2 Extração de Regras de Classificação

O conhecimento obtido por meio das árvores de decisão pode ser representado na forma de regras de classificação. Cada caminho a partir do nó raiz até um nó terminal da árvore de decisão forma uma regra de classificação. Os pares ⟨atributo, valor do atributo⟩ ao longo de um determinado caminho formam a conjunção que dará origem ao antecedente da regra. O nó terminal, contendo o valor do atributo classe, formará o conseqüente da regra. Por exemplo, percorrendo a árvore de decisão ilustrada na Figura 2.2, tem-se a seguinte regra de classificação:

$$\textbf{SE } (Idade \leq 30) \textit{ e } (Estudante = \text{sim}) \textbf{ ENTÃO } (Compra = \text{não}).$$

Cada regra extraída da árvore de decisão pode ser podada removendo-se as condições presentes no antecedente que não contribuem no valor de acurácia estimado para a regra. No entanto, para cada condição candidata à remoção, o efeito produzido pela mesma deve ser avaliado perante todas as instâncias de treinamento. Portanto, por melhor que seja o método responsável pela análise e remoção das condições nas regras, esse procedimento geralmente é computacionalmente caro. Por esse motivo, outros métodos mais eficientes

foram propostos para gerar regras de maneira direta, ou seja, sem a construção prévia de uma árvore de decisão [95].

2.6.1.3 Considerações Adicionais sobre Árvores de Decisão

O algoritmo ID3 trabalha apenas com atributos categóricos, ou seja, atributos cujos valores estão discretizados. Uma melhoria incorporada ao algoritmo C4.5 (sucessor do ID3) foi a capacidade de trabalhar com atributos contínuos. Para tanto, os valores contínuos são divididos em intervalos. O C4.5 também é capaz de trabalhar com ruídos e instâncias que possuem atributos cujos valores são desconhecidos.

Conforme visto na Subseção 2.5.2, vários métodos têm sido propostos por pesquisadores para tratar a ausência de valores de atributos nas bases de dados de treinamento e de teste [72]. Dentre as alternativas para lidar com esse problema durante a construção da árvore de decisão, tem-se: substituir os valores desconhecidos por algum valor que caracterize o atributo (média, valor mais comum); ignorar as instâncias que possuem atributos com valores desconhecidos; ou criar uma ramificação na árvore especificamente para as instâncias que possuem valores desconhecidos para o atributo em questão.

Com relação à seleção dos atributos teste ao longo do processo de construção da árvore de decisão, observa-se que a medida de ganho de informação pode ser tendenciosa no sentido de favorecer atributos que possuem uma grande quantidade de valores. No entanto, existem outras medidas de seleção de atributos teste, tais como o índice Gini e o teste do qui-quadrado em tabelas de contingência [94].

Uma questão crucial que surge com os métodos de particionamento recursivos de indução de árvores de decisão é o momento de parada na construção da árvore, ou seja, identificar o tamanho ideal da árvore de decisão. Quando uma árvore de decisão é construída, alguns de seus ramos podem ser resultado de anomalias presentes na base de dados de treinamento, prejudicando a capacidade de generalização do modelo. Esse problema é conhecido como sobre-ajustamento aos dados de treinamento (*overfitting*). Na tentativa de se evitar esse problema, técnicas de poda foram propostas.

Duas abordagens são utilizadas pelas técnicas de poda. Na primeira, denominada pré-poda, se durante a construção da árvore um determinado critério de parada for atingido, o seu crescimento é interrompido. Uma das dificuldades dessa abordagem é a definição de um critério de parada que seja adequado para a construção de uma árvore com boa acurácia preditiva. No caso da segunda abordagem, denominada pós-poda, alguns ramos

da árvore são removidos após a etapa de construção da mesma. Nesse caso, os nós contidos nos ramos removidos deixam de fazer parte da árvore e os nós terminais passam a ser aqueles que não possuem ramificações.

Algumas das vantagens da utilização de árvores de decisão estão relacionadas à agilidade na construção do modelo, ao fato de serem de fácil interpretação e de terem capacidade de trabalhar com grande quantidade de atributos.

Aplicações reais de mineração de dados, em geral, envolvem bases de dados contendo vários milhões de instâncias. O fato de alguns algoritmos de indução de árvores de decisão, tais como o ID3 e o C4.5, trabalharem com as instâncias de treinamento armazenadas em memória principal, limita a aplicação dos mesmos a grandes bases de dados. Desse modo, algoritmos de indução de árvores de decisão capazes de trabalhar com grandes bases de dados foram propostos, dentre eles, o SLIQ [60], o SPRINT [84] e o RainForest [33].

2.6.2 Classificação Bayesiana

Os classificadores Bayesianos, pertencentes à classe dos métodos estatísticos, utilizam o teorema de Bayes para realizar a tarefa de classificação. Os classificadores Bayesianos simples assumem que o efeito do valor de um atributo sobre uma determinada classe independe dos valores dos demais atributos. Essa consideração, conhecida como independência condicional, tem como objetivo simplificar o processamento envolvido na tarefa de classificação. Diferentemente desses, as redes Bayesianas permitem considerar dependências entre subconjuntos de atributos.

A Seção 2.6.2.1 mostra como o Teorema de Bayes é aplicado na tarefa classificação. O classificador Bayesiano simples e as redes Bayesianas são respectivamente descritos nas seções 2.6.2.2 e 2.6.2.3.

2.6.2.1 Teorema de Bayes

Seja $C = \{C_1, C_2, \ldots, C_m\}$ um conjunto de classes e X uma instância cuja classe é desconhecida. Considerando-se que X pertence a uma das classes do conjunto C, em problemas de classificação, deseja-se determinar $P(C_i|X)$, $1 \leq i \leq m$, ou seja, a probabilidade da classe C_i dada a instância X. O teorema de Bayes permite o cálculo da probabilidade *a posteriori* da classe C_i condicionada a X, $P(C_i|X)$, da seguinte forma:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}, \tag{2.14}$$

onde $P(C_i)$ é a probabilidade *a priori* de C_i, $P(X)$ é a probabilidade *a priori* de X e $P(X|C_i)$ é a probabilidade *a posteriori* de X condicionada a C_i. As probabilidades $P(C_i)$, $P(X)$, e $P(X|C_i)$ podem ser estimadas a partir das instâncias de treinamento.

2.6.2.2 Classificação Bayesiana Simples

O classificador Bayesiano simples [26] considera que cada instância é representada por um vetor de atributos n-dimensional, $B = \{b_1, b_2, \ldots, b_n\}$, o qual contém os valores dos seus n atributos, $A_1, A_2, \ldots, A_n$. Supondo que existam m classes, $C_1, C_2, ..., C_m$, a uma instância X cuja classe é desconhecida, esse classificador atribuirá a classe que tem a maior probabilidade *a posteriori* $P(C_i|X)$. Desse modo, X é associada à classe C_k, se e somente se

$$P(C_k|X) > P(C_j|X) \quad \forall j,\ 1 \leq j \leq m, \quad j \neq k. \tag{2.15}$$

Portanto, para se classificar uma instância desconhecida X, basta encontrar a maior probabilidade $P(C_i|X)$. O cálculo de $P(C_i|X)$ é realizado utilizando-se o teorema de Bayes (Equação 2.14). Como $P(X)$ é constante para todas as classes, $P(C_i|X)$ pode ser expresso como

$$P(C_i|X) \propto P(X|C_i)P(C_i). \tag{2.16}$$

$P(C_i)$ pode ser estimado a partir da base de dados de treinamento da seguinte maneira:

$$P(C_i) = \frac{|C_i|}{N}, \tag{2.17}$$

onde $|C_i|$ é o número de instâncias de treinamento que pertencem à classe C_i e N é o número total de instâncias de treinamento.

Visando reduzir o custo computacional envolvido no cálculo da probabilidade $P(X|C_i)$, o classificador Bayesiano simples assume independência condicional entre os atributos. Isso permite expressar $P(X|C_i)$ como

$$P(X|C_i) = \prod_{j=1}^{n} P(x_j|C_i). \tag{2.18}$$

As probabilidades $P(x_j|C_i)$ podem ser estimadas a partir da base de dados de treinamento, da seguinte forma:

1. Se A_j for um atributo categórico, então $P(x_j|C_i) = \frac{s_{ij}}{|C_i|}$, onde s_{ij} é o número de instâncias de treinamento da classe C_i cujo atributo A_j possui o valor x_j, e $|C_i|$ é o

número de instâncias de treinamento que pertencem à classe C_i.

2. Se A_j for um atributo contínuo, então deve-se trabalhar com uma distribuição de probabilidade para o atributo. Geralmente a distribuição Gaussiana é considerada, resultando no seguinte cálculo:

$$P(x_j|C_i) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x_j-\mu_{C_i})^2}{2\sigma_{C_i}^2}} \tag{2.19}$$

onde, μ_{C_i} e σ_{C_i} são, respectivamente, a média e o desvio padrão, calculados considerando-se os valores do atributo A_j das instâncias que pertencem à classe C_i.

O exemplo a seguir ilustra o funcionamento do classificador Bayesiano simples. Considere a base de dados de treinamento mostrada na Tabela 2.3, onde dois atributos numéricos, A_1 e A_2, descrevem cinco instâncias que pertencem à classe C_1 ou C_2. Deseja-se predizer a classe da nova instância $X = [x_1, x_2] = [1, 1]$. A predição da classe da instância X é realizada comparando-se $P(C_1|X)$ com $P(C_2|X)$ (veja Equação 2.15). Para tais cálculos necessita-se conhecer os valores de $P(X|C_1)$, $P(X|C_2)$, $P(C_1)$ e $P(C_2)$.
As probabilidades *a priori* das classes são estimadas a partir da base de treinamento, resultando em $P(C_1) = \frac{3}{5}$ e $P(C_2) = \frac{2}{5}$.

Tabela 2.3: Base de dados de treinamento

Atributos		Classe
A_1	A_2	
1	0	C_1
0	0	C_1
2	1	C_2
1	2	C_2
0	1	C_1

Conforme mostrado na Equação 2.18, estimando-se $P(x_j|C_i)$ a partir das instâncias de treinamento, tem-se:

$$P(X|C_1) = P(x_1 = 1|C_1).P(x_2 = 1|C_1) = \frac{1}{3} \times \frac{1}{3} = \frac{1}{9}$$

e

$$P(X|C_2) = P(x_1 = 1|C_2).P(x_2 = 1|C_2) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}.$$

Finalmente, de acordo com a Equação 2.16, conclui-se que:

$$P(C_1|X) \propto \frac{1}{9} \times \frac{3}{5} = \frac{1}{15}$$

e

$$P(C_2|X) \propto \frac{1}{4} \times \frac{2}{5} = \frac{1}{10}.$$

Como $P(C_2|X)$ é a maior probabilidade, a classe C_2 é atribuída à instância X.

2.6.2.3 Redes Bayesianas

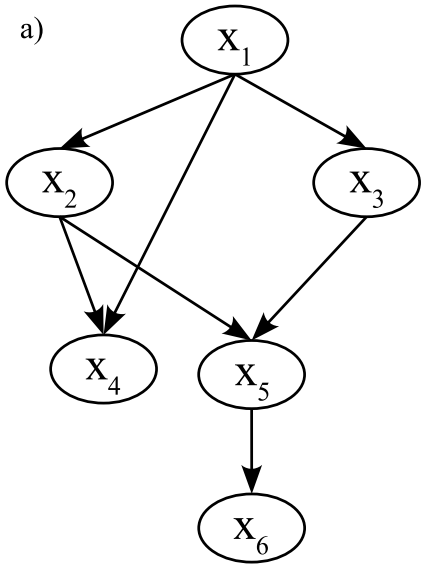
O classificador Bayesiano simples assume que o efeito do valor de um atributo sobre uma determinada classe independe dos valores dos demais atributos. Apesar dessa consideração simplificar o processo de classificação, na prática, podem existir dependências entre os atributos. As redes Bayesianas permitem incluir no modelo dependências entre subconjuntos de atributos.

Uma rede Bayesiana é definida por dois componentes. O primeiro corresponde a um grafo orientado acíclico, cujos nós representam variáveis aleatórias e os arcos definem as relações de dependências entre as variáveis. Se existir um arco de um nó N para um nó M, então M é descendente de N, e N é pai de M. Cada variável representada no grafo é condicionalmente independente de todos os seus nós não descendentes, dados os seus nós pais. Essas variáveis podem corresponder a atributos presentes nos dados, ou a variáveis "externas" acreditadas como sendo formadoras de algum relacionamento.

O segundo componente é formado por um conjunto de probabilidades condicionais associadas às variáveis. Sendo assim, cada nó do grafo está associado a uma tabela de probabilidades condicionais (TPC). Essas probabilidades condicionais quantificam a força da influência entre as variáveis conectadas no grafo.

A Figura 2.5a apresenta uma rede Bayesiana (retirada de [48]), onde pode-se observar a relação causal direta existente entre suas seis variáveis binárias. Nessa figura, verifica-se, por exemplo, que a variável X_4 é influenciada pelas variáveis X_1 e X_2. Além disso, observa-se que a variável X_4 é condicionalmente independente de X_3, dados os seus nós pais, X_1 e X_2. Isso significa que, conhecidos os valores de X_1 e X_2, a variável X_3 não fornece qualquer informação adicional com relação à variável X_4.

A quantificação da relação causal é dada por uma distribuição condicional $P(Z|$ $Pais(Z))$, onde $Pais(Z)$ correspondem aos nós pais de Z. A Figura 2.5b, apresenta a TPC da variável X_4. Nessa tabela, a probabilidade condicional para cada valor de X_4



b)

	x_1, x_2	$x_1, \sim x_2$	$\sim x_1, x_2$	$\sim x_1, \sim x_2$
x_4	0,8	0,5	0,7	0,1
$\sim x_4$	0,2	0,5	0,3	0,9

Figura 2.5: (a) Exemplo de uma rede Bayesiana e (b) Tabela de probabilidades condicionais da variável X_4.

é dada para cada possível combinação de valores de seus pais. Por exemplo,

$$P(X_4 = \text{sim}|X_1 = \text{não}, X_2 = \text{sim}) = 0,7.$$

A probabilidade conjunta de qualquer tupla $(x_1, x_2, \ldots, x_n)$ correspondente às variáveis (ou atributos) $X_1, X_2, \ldots, X_n$ é dada por

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|Pais(X_i)).$$

Por exemplo, utilizando-se a Figura 2.5 como referência, tem-se:

$$P(x_1, x_2, x_3, x_4, x_5, x_6) = P(x_6|x_5).P(x_5|x_2, x_3).P(x_4|x_1, x_2).P(x_3|x_1).P(x_2|x_1).P(x_1).$$

Qualquer nó da rede Bayesiana pode representar o atributo dependente de um problema de classificação, sendo que, pode existir ao mesmo tempo mais de um nó representando atributos dependentes. Além disso, o processo de classificação em vez de retornar unicamente uma classe, pode fornecer uma distribuição de probabilidade, ou seja, a probabilidade de cada uma das possíveis classes ocorrer.

O processo de treinamento de uma rede Bayesiana varia de acordo com o cenário, pois a estrutura da rede pode ser previamente conhecida ou inferida a partir dos dados. Além disso, as variáveis da rede também podem ser conhecidas ou possuir valores desconhecidos.

Se a estrutura da rede é dada e as variáveis são conhecidas, então o processo de treinamento resume-se ao cálculo de probabilidades condicionais (elaboração da TPC), de maneira semelhante aos cálculos executados na classificação Bayesiana simples.

Quando a estrutura é dada e as variáveis possuem valores desconhecidos, métodos como gradiente descendente [81] e EM (*Expectation Maximization*) [54] podem ser utilizados no treinamento da rede Bayesiana.

O caso em que a estrutura é desconhecida e as variáveis são conhecidas corresponde a um problema de otimização discreta. Para se resolver esse tipo de problema, algoritmos foram propostos em [19] e [40].

2.6.2.4 Considerações Adicionais sobre Classificadores Bayesianos

Os classificadores Bayesianos são considerados simples, e geralmente apresentam alta acurácia preditiva e boa escalabilidade para grandes bases de dados. Segundo [64], seu desempenho é comparável ao das árvores de decisão e redes neurais. No entanto, o desempenho do classificador pode ser afetado se as suposições feitas na sua utilização não se verificarem na prática, ou se a quantidade de dados disponíveis não for suficiente para uma boa estimativa das probabilidades.

2.6.3 Classificação Baseada em Conceitos de Regras de Associação

A mineração de regras de associação é uma importante área de pesquisa em mineração de dados, cujo objetivo é encontrar regras interessantes a partir de um conjunto de dados históricos [1]. Tais regras correspondem a padrões que ocorrem com uma determinada freqüência numa dada base de dados.

Recentemente, métodos de classificação que aplicam conceitos relacionados a regras de associação têm sido propostos. Para essas técnicas, cada par ⟨atributo, valor do atributo⟩ da base de dados é conhecido como um item. Desse modo, as instâncias são descritas por conjuntos de itens. O foco dessas técnicas está na extração de padrões (subconjuntos de itens) que aparecem com freqüência significativa na base de dados. Os padrões extraídos a partir de uma base de dados dão origem às regras que são utilizadas na tarefa de classificação. Essas regras apresentam-se na forma $X \Rightarrow Y$, onde X é um conjunto de itens (ou pares ⟨atributo, valor do atributo⟩) e Y é o atributo classe com seu respectivo valor.

Os métodos ARCS (*Association Rule Clustering System*) [55] e classificação associa-

tiva (*Associative Classification*) [58] utilizam regras de associação no processo de classificação. Já o método CAEP (*Classification by Aggregating Emerging Patterns*) [24] não trabalha diretamente com as regras, mas sim com "padrões emergentes", que também consideram o conceito de suporte utilizado por técnicas de mineração de regras de associação.

O suporte e a confiança são medidas de interesse freqüentemente utilizadas por técnicas de mineração de regras de associação, sendo suas interpretações dadas a seguir. Um suporte de 2% para uma regra $A \Rightarrow B$, onde A e B são os conjuntos de itens, significa que $(A \cup B)$ aparece em 2% do total de instâncias da base de dados. Já uma confiança de 70% para essa mesma regra, indica que dentre as instâncias que contêm A, 70% também contêm B.

Nesta seção, os métodos ARCS, classificação associativa e CAEP serão abordados.

2.6.3.1 ARCS

Em [55], um método denominado ARCS (*Association Rule Clustering System*) é proposto para extrair regras de associação a partir de um conjunto de dados pré-processados e utilizá-las na tarefa de classificação. As regras possuem o formato $A \wedge B \Rightarrow C$, onde A e B são atributos quantitativos que caracterizam a instância e C é um atributo categórico que corresponde à classe da instância. As regras extraídas são representadas numa matriz bidimensional, a partir da qual um algoritmo realiza a busca por agrupamentos retangulares de regras. Desse modo, regras diferentes, mas com valores adjacentes de atributos, podem ser agrupadas. Por fim, as regras de associação agrupadas são utilizadas na classificação.

O processo inicial do método consiste na leitura da base de dados e no particionamento dos valores dos atributos em intervalos. Desse modo, um atributo inteiro que varia entre 0 e 99, poderia ser particionado, por exemplo, em dois intervalos: $[0, 50)$ e $[50, 100]$. O objetivo do particionamento é viabilizar a construção de uma matriz bidimensional com tamanho não proibitivo para as etapas seguintes.

O passo seguinte utiliza um algoritmo específico para lidar com os dados particionados com o objetivo de gerar regras de associação de acordo com um suporte e confiança mínimos, previamente estabelecidos.

Uma vez extraídas as regras de associação segundo determinados valores de suporte e confiança, uma matriz bidimensional é construída a partir das regras que estão relacionadas com um determinado valor do atributo classe C. A partir disso, uma busca

por agrupamentos retangulares de regras é realizada na matriz. As regras de associação resultantes dos agrupamentos encontrados são utilizadas no processo de classificação. A Figura 2.6 mostra um exemplo de matriz bidimensional, onde os eixos representam os atributos *Salário* e *Idade*, e as regras consideradas na construção dessa matriz predizem a condição "compra(computador)".

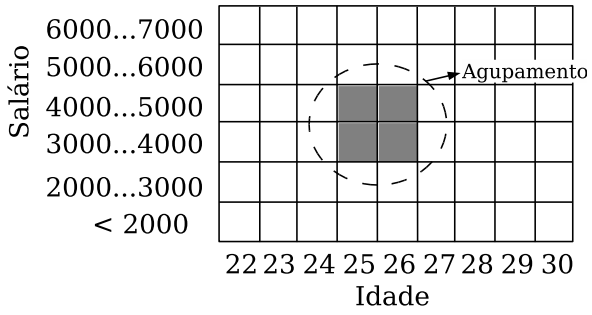


Figura 2.6: Matriz bidimensional representando as características dos consumidores que compram computadores.

As quatro regras representadas no exemplo da Figura 2.6 são:

idade(25) $\wedge$ salário(3000...4000) $\Rightarrow$ compra(computador)

idade(26) $\wedge$ salário(3000...4000) $\Rightarrow$ compra(computador)

idade(25) $\wedge$ salário(4000...5000) $\Rightarrow$ compra(computador)

idade(26) $\wedge$ salário(3000...4000) $\Rightarrow$ compra(computador)

A regra resultante do agrupamento é:

idade(25...26) $\wedge$ salário(3000...5000) $\Rightarrow$ compra(computador)

A acurácia do modelo, estimada a partir de um conjunto de instâncias de teste, é utilizada como parâmetro para que o sistema (ARCS) ajuste os valores de suporte e/ou confiança e reinicie o processo de extração de regras de associação. Esse procedimento iterativo de ajustes continua enquanto existir melhoria nas regras de associação geradas, ou o tempo limite de processamento não for atingido.

De acordo com os experimentos realizados em [55], para bases de dados contendo ruídos, o método ARCS apresentou acurácia ligeiramente maior do que o C4.5. Em termos de escalabilidade, o tempo de execução do ARCS cresce no máximo linearmente com o aumento da base de dados. Além disso, para um determinado tamanho de matriz, a quantidade de memória necessária independe do tamanho da base de dados. O C4.5 apresentou tempos de execução maiores do que o ARCS.

2.6.3.2 Classificação Associativa

A classificação associativa integra as tarefas de extração de regras de associação e classificação. O objetivo dessa integração é realizar a classificação a partir de regras de associação especiais $X \Rightarrow Y$, denominadas regras de associação de classe (*Class Association Rules – CARs*), onde X corresponde à conjunção de condições definidas por pares ⟨atributo, valor do atributo⟩ e Y é o valor do atributo classe.

A extração das CARs é realizada por meio de uma adaptação do algoritmo *Apriori* [2]. Essa adaptação envolve a discretização dos atributos contínuos e a redução do número de regras geradas.

O método de classificação associativa, proposto em [58], consiste, basicamente, em três etapas. Na primeira, realiza-se a discretização dos atributos contínuos. A segunda etapa envolve a extração das regras de associação (CARs). Por fim, na terceira etapa, o classificador é construído a partir das regras extraídas.

O algoritmo que executa todas essas etapas é denominado CBA (*Classification Based on Association*). Ele possui duas partes, o gerador de regras (CBA-RG), que utiliza os parâmetros suporte e confiança mínimos definidos pelo usuário, e o construtor do classificador (CBA-CB), onde as regras extraídas são ordenadas de acordo com seus valores de suporte e confiança.

A classificação de uma nova instância é dada pela primeira regra que se adequar à mesma, ou seja, pela primeira regra cujos pares ⟨atributo, valor do atributo⟩ forem compatíveis com os pares que caracterizam a instância. Caso nenhuma das regras seja adequada, o classificador utiliza uma regra padrão para classificá-la.

Os experimentos realizados em [58] mostraram que o CBA apresenta uma acurácia superior à do C4.5 para várias bases de dados.

2.6.3.3 CAEP

O CAEP utiliza um conceito proposto em [23], denominado padrões emergentes (*Emerging Patterns - EPs*). Padrões emergentes são conjuntos de itens (pares atributo-valor) cujos suportes se alteram significativamente entre instâncias da base pertencentes a diferentes classes. Por exemplo, suponha um conjunto de dados de consumidores contendo dois valores para o atributo classe *compra*: *sim* (C_1) e *não* (C_2). O conjunto de itens s= {idade < 40, salário > 1500} é um típico EP, uma vez que seu suporte aumenta de 0,1% na classe

C_1 para 65% na classe C_2, ou seja, uma taxa de crescimento igual a $\frac{65}{0,1} = 650$. Cada EP representa uma grande diferenciação entre classes, permitindo a identificação das classes das instâncias. Isso pode ser observado no exemplo anterior, onde a probabilidade de uma instância com as características de s pertencer à classe C_2 é muito alta. Geralmente o poder de diferenciação de um EP está relacionado com a taxa de crescimento de seu suporte de uma classe para outra e com seu suporte numa determinada classe.

Como um EP sozinho geralmente possui uma capacidade limitada para predizer as classes das instâncias, a utilização de um único EP pode resultar numa acurácia preditiva muito baixa para o classificador. Visando a construção de um classificador com um bom desempenho, o CAEP encontra, para cada classe C, todos os EPs que ultrapassam um suporte e uma taxa de crescimento mínimos, sendo a taxa de crescimento computada por meio da relação entre todas as instâncias que não pertencem à classe C e as que são dessa classe. Em seguida, ele agrega o poder de diferenciação dos EPs encontrados para permitir a classificação de novas instâncias.

Para classificar uma nova instância X, para cada classe C, os valores do poder de diferenciação dos EPs da classe C que ocorrem em X são agregados para formar uma pontuação para a classe C. A pontuação obtida por cada classe é normalizada com o intuito de reduzir o efeito do desbalanceamento da distribuição dos EPs entre as classe. Por fim, a classe com a maior pontuação normalizada é atribuída à instância X.

Os resultados experimentais apresentados em [24] mostraram que o CAEP apresenta maior acurácia do que o C4.5 e o CBA para várias bases de dados. Além disso, o seu desempenho mantém-se constante para todas as classes, mesmo para bases de dados de treinamento onde existem classes dominantes (que englobam a maioria das instâncias) e classes minoritárias. Verificou-se também que o CAEP é escalável com relação ao tamanho da base de dados de treinamento e quantidade de atributos de suas instâncias.

2.6.3.4 Outros Classificadores Baseados em Conceitos de Regras de Associação

Assim como na classificação associativa, outros trabalhos propuseram a utilização de regras de associação para a construção de um classificador. A diferença entre as propostas geralmente concentra-se na questão da seleção das regras para a construção do modelo de classificação. No entanto, em [5], apresenta-se uma proposta que utiliza diferentes tipos de regras de associação na construção do modelo, ou seja, além das regras de associação positivas, são consideradas também as regras de associação negativas. Considerando-se regras no formato $X \Rightarrow Y$, onde X representa o conjunto de valores dos atributos e Y

corresponde à classe, as regras de associação negativas consideradas em [5] são do tipo $\neg X \Rightarrow Y$ e $X \Rightarrow \neg Y$. Para medir o desempenho desse classificador, seis bases de dados da *UCI Machine Learning Repository* [12] foram utilizadas. Para três bases de dados os resultados de acurácia foram melhores do que os apresentados pelos métodos árvore de decisão (C4.5) e classificação associativa.

O método CAEP foi o primeiro a utilizar os EPs, mas outras propostas surgiram depois dele. Influenciado pelo CAEP, um classificador denominado JEP foi proposto em [56]. Sua principal característica é a utilização de um tipo especial de EP, denominados JEP (*Jumping Emerging Pattern*). O JEP é um padrão emergente cuja taxa de crescimento de suporte é infinita, ou seja, um EP cuja freqüência é diferente de zero para uma determinada classe da base de dados e nula para as demais. Segundo [24], o CAEP é melhor para bases de dados com poucos ou nenhum JEP, enquanto o classificador JEP apresenta melhores resultados quando vários JEPs estão presentes. Em termos de desempenho, os dois classificadores são considerados competitivos quando comparados com árvores de decisão (C4.5) e classificação associativa.

Outro classificador que utiliza EPs foi proposto recentemente em [57]. Esse classificador, denominado DeEPs, utilizando uma abordagem *lazy*, procura por padrões que tenham capacidade de diferenciar as classes presentes na base de dados de treinamento. Uma vez que os padrões correspondem a subconjuntos de itens, o número de padrões possíveis cresce exponencialmente com o aumento da dimensão das instâncias. Para lidar com esse problema, o DeEPs utiliza um método de redução de dados. Para avaliar a importância de cada padrão encontrado, eles são ordenados de acordo com algumas medidas de interesse, tais como freqüência do padrão, tamanho, e taxa de crescimento da freqüência. A predição das classes das instâncias é realizada por meio de comparação das freqüências agregadas dos padrões para cada classe. De acordo com os experimentos realizados para 40 bases de dados da *UCI Machine Learning Repository* [12], o DeEPs apresentou desempenho comparável ou melhor do que classificadores como o k-NN (k-*Nearest Neighbor*) [20] e C5.0 (versão comercial do C4.5).

2.6.4 Outros Métodos de Classificação

Além dos métodos e algoritmos de classificação citados nas seções anteriores, alguns outros métodos, freqüentemente encontrados na literatura de mineração de dados, são abordados nesta seção. Esses métodos incluem o k-vizinhos mais próximos (k-*Nearest Neighbor* – k-NN) [3], Máquinas de Vetor de Suporte (*Support Vector Machines* – SVM) [89] e Redes

Neurais [39]. A seguir, tem-se uma breve descrição de cada um deles.

2.6.4.1 *k*-NN

A técnica denominada k-vizinhos mais próximos (k-*Nearest Neighbor* – k-NN) foi inicialmente analisada em [31], sendo sua aplicação em problemas de classificação pioneiramente realizada em [47]. No entanto, foi somente a partir dos resultados apresentados em [3] que essa abordagem ganhou popularidade como método de classificação nas áreas de aprendizado de máquina e mineração de dados.

A idéia do classificador k-NN é bastante simples. Cada instância B é descrita por um vetor n-dimensional $B = \{b_1, b_2, \ldots, b_n\}$, onde $b_1, b_2, \ldots, b_n$ correspondem aos valores dos seus n atributos, $A_1, A_2, \ldots, A_n$. Para classificar uma nova instância (uma instância cuja classe é desconhecida), o classificador utiliza alguma métrica de distância para determinar as k instâncias de treinamento mais similares (próximas) à instância que se deseja classificar. Em seguida, a classe mais freqüente entre essas k instâncias mais próximas é atribuída à nova instância. No k-NN, o valor de k é um parâmetro de entrada.

A definição da similaridade entre as instâncias é dada por uma função de distância. Apesar de existirem diversas funções de distância, a mais usual para esse tipo de classificador é a Euclidiana, onde a distância entre duas instâncias, $X = \{x_1, x_2, \ldots, x_n\}$ e $Y = \{y_1, y_2, \ldots, y_n\}$, é dada por

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.20}$$

Geralmente, os atributos de uma instância possuem valores em diferentes escalas. Desse modo, a aplicação direta da função Euclidiana pode reduzir drasticamente a contribuição de um atributo no cálculo da distância quando existirem outros atributos com escalas de medidas maiores. Um processo de normalização dos valores dos atributos pode evitar esse desbalanceamento na contribuição de cada atributo no cálculo da distância. Uma possível normalização é colocar todos os atributos variando numa escala entre 0 e 1. Para tal, utiliza-se a a seguinte equação:

$$a_i = \frac{v_i - min\ v_i}{max\ v_i - min\ v_i}, \tag{2.21}$$

onde a_i é a normalização do valor original v_i do atributo i, e $min\ v_i$ e $max\ v_i$ são, respectivamente, o menor e o maior valores do atributo i na base de dados de treinamento.

A distância Euclidiana, como foi apresentada, é aplicável a atributos numéricos. Uma alternativa comumente utilizada quando os atributos são nominais é considerar cada diferença $(x_i - y_i)$ existente na Equação (2.20) sendo igual a 0 se os valores x_i e y_i forem iguais, e 1 em caso contrário.

2.6.4.2 Máquinas de Vetor de Suporte

Métodos como árvores de decisão e regras de decisão trabalham com maior naturalidade quando os atributos são nominais. No entanto, por meio de um processo de discretização ou utilizando-se os próprios valores numéricos nos testes realizados nas árvores ou regras de decisão, esses métodos podem ser facilmente estendidos para trabalharem com atributos numéricos. Observa-se que métodos de regressão linear trabalham de maneira mais natural quando os atributos são numéricos.

A regressão linear pode ser facilmente utilizada na tarefa de classificação quando o domínio da aplicação for constituído por atributos numéricos. A idéia central é expressar uma classe como uma combinação linear de atributos, da seguinte maneira:

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_n a_n, \tag{2.22}$$

onde x é a classe, $a_1, a_2, \ldots, a_n$ são os valores dos atributos, e $w_1, w_2, \ldots, w_n$ são os pesos. Esses pesos são calculados a partir das instâncias de treinamento.

Como pode ser observado na Equação (2.22), a regressão linear é adequada quando o atributo classe é numérico. Uma vez que nos problemas de classificação freqüentemente o atributo classe não é numérico, o seguinte artifício pode ser utilizado para viabilizar a utilização da regressão linear na tarefa de classificação. Para cada classe C, a saída x da Equação (2.22) é igual a 1 para as instâncias de treinamento que pertencem à C, e 0 para todas as outras. O resultado desse procedimento é a obtenção de um modelo linear, ou seja, uma equação linear para cada classe. Então, a classificação de uma instância desconhecida é dada calculando-se o valor das equações lineares de cada classe e escolhendo-se a classe daquela que retornar o maior valor.

Apesar da simplicidade e da grande aplicabilidade da regressão linear na área de estatística, a grande desvantagem dos modelos lineares para a tarefa de classificação é a limitação de representarem somente limites lineares entre as classes. O método proposto em [89], denominado máquinas de vetor de suporte (*Support Vector Machines* – SVM), utiliza, de forma mais adequada, modelos lineares mesmo para domínios contendo classes não linearmente separáveis.

Para explicar como SVM funciona, talvez seja mais didático começar pelo caso onde as classes são linearmente separáveis. A idéia principal de uma máquina de vetor de suporte é encontrar um tipo especial de modelo linear, o hiperplano com máxima margem de separação. Para explicar o conceito de hiperplano com máxima margem de separação, o seguinte exemplo será considerado. Seja uma base de dados contendo duas classes linearmente separáveis, ou seja, onde existe um hiperplano no espaço amostral que permite uma classificação correta de todas as instâncias de treinamento. O hiperplano com máxima margem de separação, denominado hiperplano ótimo, é aquele que proporciona a maior separação possível entre as classes. A Figura 2.7 ilustra a construção geométrica de um hiperplano ótimo, para um espaço amostral bidimensional, que separa instâncias das classes representadas pelos símbolos "+" e "∘".

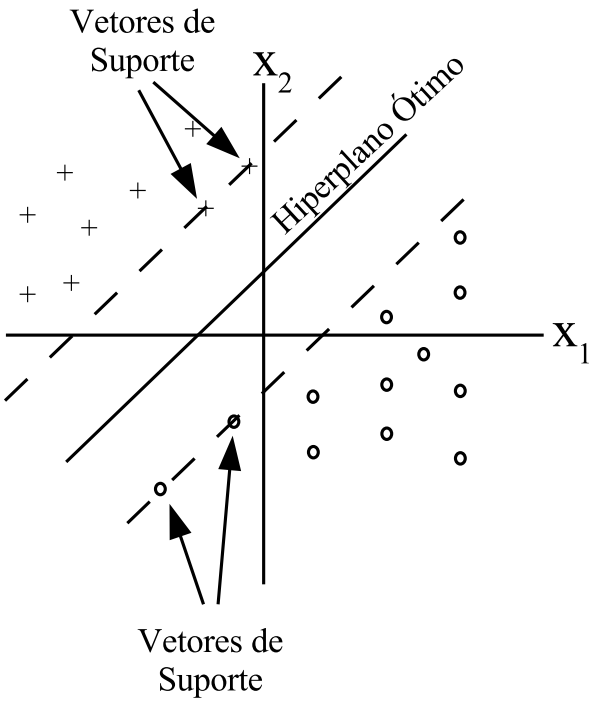


Figura 2.7: Hiperplano ótimo para classes linearmente separáveis.

As instâncias que possuem distância mínima até o hiperplano são denominadas vetores de suporte. Na Figura 2.7 pode-se observar quatro vetores de suporte.

Considere que cada instância seja representada pelo par $\{\mathbf{x}_i, C_i\}$, onde $\mathbf{x}_i$ é o vetor de atributos e C_i a classe da instância i. Assumindo que existem duas classes linearmente separáveis, $C_i = +1$ e $C_i = -1$, a equação do hiperplano que realiza essa separação é

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.23}$$

A etapa de treinamento do modelo de classificação consiste em encontrar os parâmetros **w** e **b**, os quais determinam o hiperplano ótimo. Uma vez treinado o modelo, a classificação de uma instância desconhecida é feita determinando-se a posição da mesma em relação ao hiperplano de separação.

Caso as instâncias de treinamento não sejam linearmente separáveis, o seguinte artifício é utilizado:

1. Um mapeamento não linear transforma o espaço amostral inicial num novo espaço, denominado espaço de características, onde as instâncias são linearmente separáveis.
2. Assim como no caso separável, o hiperplano ótimo é encontrado no espaço de características.

O problema de encontrar os vetores de suporte para uma dada base de dados de treinamento e determinar os parâmetros **w** e **b** pode ser formulado como um problema clássico de otimização, conhecido como programação quadrática com restrições lineares. Esse problema de otimização pode ser resolvido analiticamente somente quando a base de dados de treinamento é muito pequena.

Contudo, para a maioria dos casos reais, esse problema de otimização deve ser resolvido numericamente. Para problemas pequenos, qualquer pacote computacional que resolva problemas de programação quadrática com restrições lineares pode ser utilizado. Para problemas maiores, diversas técnicas vêm sendo propostas. Um tutorial abrangendo os mais variados aspectos de SVM foi publicado em [17].

2.6.4.3 Classificação por Redes Neurais

Redes Neurais Artificiais (RNAs) são modelos matemáticos que se assemelham às estruturas neurais biológicas e que têm capacidade computacional adquirida por meio de aprendizado e generalização [76, 39]. As RNAs são capazes de resolver problemas de predição, classificação e otimização.

Uma RNA é formada por um conjunto de unidades de entrada conectadas a unidades de saída, onde cada conexão possui um peso associado. Algumas RNAs admitem unidades e conexões intermediárias. A fase de treinamento de uma RNA envolve o ajuste de seus pesos de modo a capacitá-la a predizer corretamente a classe de instâncias desconhecidas.

Um exemplo de uma RNA do tipo *feed-forward* é mostrado na Figura 2.8. As entradas correspondem aos valores dos atributos de cada instância de treinamento. A alimentação

da RNA é realizada por meio das unidades que formam a camada de entrada. As saídas dessas unidades alimentam as unidades da camada intermediária, que por sua vez, alimentam a camada de saída. O resultado da classificação em uma RNA é obtido a partir das unidades que compõem a camada de saída. O número de unidades de entrada e saída depende da dimensionalidade dos dados, enquanto o número de unidades nas camadas intermediárias depende da complexidade do problema. Uma rede neural pode, por exemplo, ser formada por uma quantidade de unidades de entrada igual ao número de atributos que caracterizam as instâncias e uma única unidade de saída, que indicará a classe da instância. O número de camadas intermediárias é arbitrário, embora na prática seja usual uma única camada.

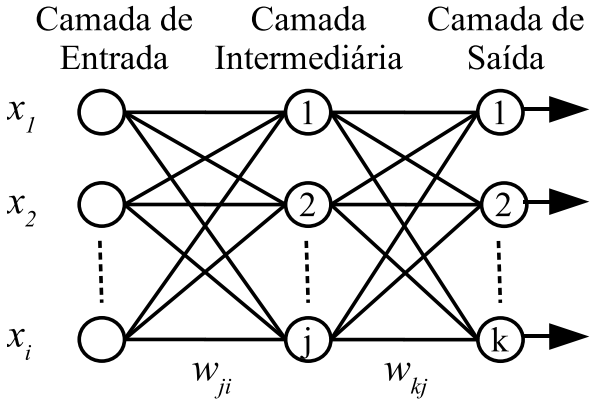


Figura 2.8: Exemplo de uma rede neural. A instância de treinamento $X = \{x_1, x_2, \ldots, x_i\}$ está alimentando a rede. Os pesos estão representados por w_{ji} e w_{kj}.

Antes de treinar uma RNA deve-se decidir sobre sua estrutura, ou seja, o número de unidades da camada de entrada, o número de camadas intermediárias e a quantidade de unidades em cada uma dessas camadas, e por fim, o número de unidades da camada de saída. Não existem regras que determinem o número ideal de camadas intermediárias. Sendo assim, o projeto de uma rede é um processo de tentativa e erro. Uma vez que a estrutura da rede determina diretamente a qualidade do modelo obtido, se uma RNA é treinada e sua acurácia preditiva não é satisfatória, é comum repetir-se o processo de treinamento em uma rede topologicamente diferente e com pesos iniciais modificados.

As RNAs caracterizam-se pelo aprendizado por meio de exemplos. Dado um conjunto de instâncias de treinamento, o algoritmo de aprendizado realiza a adaptação dos parâmetros (pesos) da rede. O primeiro modelo de RNA que envolveu o conceito de aprendizado foi o Perceptron Simples [79], descrito no final da década de 1950. Esse modelo possuía uma única camada. O treinamento de redes de múltiplas camadas surgiu com a neces-

sidade de se resolver problemas de maior complexidade. No entanto, ele só foi possível em meados da década de 1980, com o surgimento do algoritmo de treinamento *back-propagation* [80]. O termo *backpropagation* deve-se ao fato do algoritmo de treinamento se basear na retropropagação dos erros para realizar os ajustes dos pesos das camadas intermediárias.

Na fase de treinamento de uma RNA, os dados de entrada são propagados para frente (*forward*) até a saída da rede. Como o valor da saída desejado para cada entrada é conhecido, o erro para a camada de saída pode ser calculado, permitindo então que os pesos da camada de saída sejam ajustados. Como se conhece o valor desejado somente para a camada de saída, o ajuste dos pesos das camadas intermediárias é realizado por meio de propagação para trás do erro calculado na saída, o que caracteriza o treinamento por *backpropagation*.

O treinamento de uma RNA geralmente é um processo lento. Sendo assim, sua utilização torna-se vantajosa somente para determinados tipos de aplicações. Além disso, alguns parâmetros da rede só podem ser ajustados empiricamente, tal como sua estrutura. Uma crítica freqüentemente feita por pesquisadores da área de mineração de dados é a baixa interpretabilidade das RNAs, devido à dificuldade humana de interpretação dos pesos relacionados ao modelo. Em contrapartida, as RNAs são apreciadas por sua grande capacidade de generalização, bem como por sua alta tolerância a ruídos. A questão da baixa interpretabilidade tem motivado pesquisas voltadas para a representação simbólica do conhecimento extraído a partir das RNAs. Desse modo, vários algoritmos de extração de regras a partir de RNAs têm sido propostos [82, 86, 7].

Capítulo 3

Classificação de Proteínas

3.1 Considerações Iniciais

Nos últimos anos, vários projetos de seqüenciamento de genomas surgiram com o objetivo de revelar a informação contida no DNA das células. Após o seqüenciamento do genoma de um organismo, os pesquisadores passam para a etapa de estudo dos seus produtos, as proteínas. Elas são as responsáveis por grande parte da estrutura e do funcionamento das células, definindo portanto, as características e propriedades de um determinado organismo.

Proteínas são macromoléculas complexas, formadas por aminoácidos (moléculas que contêm simultaneamente grupos funcionais amina e ácido carboxílico), necessárias para os processos químicos que ocorrem nos organismos vivos. As seqüências de aminoácidos que as compõem são responsáveis por suas estruturas e funções. As proteínas realizam diversas funções em um organismo, desde o transporte de nutrientes até a catálise de reações biológicas. Desse modo, conhecer suas funções é fundamental.

Até pouco tempo, a identificação das funções das proteínas só era possível por meio de procedimentos laboratoriais caros e demorados. Entretanto, os avanços na área de biologia molecular em conjunto com a evolução tecnológica computacional resultaram na formação de grandes bases de dados biológicos. A crescente quantidade de dados disponíveis fez surgir novos desafios com relação ao armazenamento, organização, visualização e análise desses dados. Portanto, pesquisas voltadas para criação de métodos computacionais que auxiliem no processo de análise dos dados vêm se tornando cada vez mais importantes.

Neste capítulo, o problema de classificação de proteínas e uma proposta para a sua solução são apresentados. Na Seção 3.2, descreve-se o problema. A Seção 3.3 contém uma

revisão bibliográfica sobre bancos de dados de proteínas e algumas características do banco de dados utilizado neste trabalho. Em seguida, na Seção 3.4, são apresentados alguns trabalhos relacionados com o problema de classificação de proteínas. O método proposto nesta tese é descrito na Seção 3.5 e a sua avaliação é apresentada na Seção 3.6. Por fim, na Seção 3.7, propõe-se incorporar seleção de atributos no método descrito na Seção 3.5. Parte dos resultados apresentados neste capítulo foram publicados em [62, 61, 63].

3.2 O Problema de Classificação de Proteínas

A classificação de proteínas é uma importante tarefa para a biologia molecular, uma vez que, através da identificação de suas classes é possível inferir quais são as funções das proteínas. Apesar dos avanços e da intensa pesquisa nessa área, o problema de classificação de proteínas (*Protein Classification Problem* – PCP) continua sendo um grande desafio, já que sua solução envolve um grande número de variáveis (por exemplo, as características inerentes aos componentes das proteínas – os aminoácidos), combinadas em um ambiente não totalmente conhecido (ainda não se tem um conhecimento completo sobre o mecanismo de síntese das proteínas) [87].

Na busca por soluções para o PCP, várias abordagens envolvendo técnicas de mineração de dados vêm sendo propostas em trabalhos da área de bioinformática, tais como [13, 22, 38, 70, 91, 93] e outros. Geralmente, a predição das funções das proteínas é realizada de duas maneiras: predizendo a estrutura da proteína e, a partir dessa, inferindo a sua função, ou classificando as proteínas em famílias funcionais, supondo que, proteínas de uma mesma família possuam funções semelhantes [88]. Esta última corresponde à abordagem utilizada neste trabalho.

Dada a grande quantidade de fontes de informações sobre as seqüências das proteínas, surgiram métodos que trabalham diretamente com essas informações para predizerem suas funções. Um grupo de métodos utiliza a abordagem do vizinho mais próximo, ou seja, avalia similaridades entre seqüências para classificar uma nova proteína. A mais simples abordagem de vizinho mais próximo corresponde à classificação de uma nova proteína a partir da seleção da seqüência (com função conhecida) mais semelhante àquela que se deseja classificar. Ferramentas computacionais tais como Blast [4] ou Fasta [67] fornecem medidas que indicam o grau de similaridade entre as seqüências, possibilitando que a função da seqüência mais semelhante seja atribuída à nova proteína.

Algumas técnicas que também utilizam a abordagem do vizinho mais próximo cons-

troem um modelo para cada família de proteínas a partir de um conjunto representativo de seqüências selecionado para cada família. A classificação de uma nova proteína é realizada selecionando-se a família cujo modelo mais se adequa à seqüência da proteína que se deseja classificar.

A partir da sugestão de que pequenas seqüências de aminoácidos, chamadas motivos (*motifs*), poderiam se conservar em proteínas da mesma família [21], surgiu um segundo grupo de métodos de classificação de proteínas. Dado que as funções das proteínas estão diretamente relacionadas com os motivos que ocorrem nas suas seqüências, esse grupo de métodos utiliza a composição de motivos das proteínas na tarefa de predizer as funções de novas proteínas [93]. A dificuldade nessa tarefa surge do fato de que proteínas com um ou mais motivos iguais podem pertencer a diferentes famílias.

Vários bancos de dados de motivos estão disponíveis e podem ser utilizados na obtenção da composição de motivos das proteínas que serão utilizadas na tarefa de classificação. Uma outra maneira de se obter essa composição de motivos é por meio de ferramentas computacionais, tal como a MEME (*Multiple Expectation Maximization for Motif Elicitation*) [8]. Essas ferramentas capturam motivos, ou seja, seqüências de aminoácidos que se conservam em um dado conjunto de proteínas.

Por se tratar de um problema de classificação, a proposta deste trabalho pode ser vista como uma técnica de mineração de dados que utiliza informações sobre a composição de motivos de proteínas previamente estudadas (isto é, proteínas agrupadas em famílias de acordo com suas funções) para predizer a família funcional de novas proteínas. As informações sobre a composição de motivos das proteínas, bem como suas famílias funcionais, foram obtidas a partir de um banco de dados de proteínas denominado PROSITE [45].

O problema aqui abordado está definido a seguir. Dada uma base de dados de proteínas, chamada base de dados de treinamento, onde cada proteína está caracterizada por seus motivos e famílias funcionais (também denominadas classes), o objetivo é atribuir novas proteínas a classes que se encontram representadas na base de dados de treinamento.

3.3 Bancos de Dados de Proteínas

As proteínas são formadas basicamente por 20 aminoácidos que se repetem numa seqüência específica em cada uma delas. Essa seqüência, conhecida como estrutura primária, corresponde ao nível mais simples entre aqueles que compõem a organização estrutural das proteínas.

Enquanto a estrutura primária corresponde somente à seqüência dos aminoácidos, sem considerar o arranjo tridimensional da molécula, a estrutura secundária é dada pelo arranjo espacial dos aminoácidos próximos entre si na seqüência primária da proteína.

Já a estrutura terciária é formada pelo arranjo espacial de aminoácidos distantes entre si na seqüência da proteína. Ela corresponde à conformação espacial da proteína como um todo, e não somente de determinados segmentos particulares da cadeia protéica.

Os bancos de dados de proteínas podem ser divididos em três grandes grupos de acordo com a organização estrutural das proteínas que armazenam: os bancos de dados primários, os bancos de dados secundários e os bancos de dados de estruturas.

De maneira geral, os bancos de dados primários contêm informações sobre as seqüências de aminoácidos das proteínas. Como exemplos desses bancos têm-se: PIR (*Protein Identification Resource*) [9] e SWISS-PROT [15]. Já os bancos de dados secundários contêm informações derivadas dos bancos de dados primários acrescidas de outras informações, tais como: seqüências conservadas (motivos) e assinaturas. O PROSITE [45], o PRINTS [6] e o BLOCKS [41] são exemplos de bancos de dados secundários. Os bancos de dados de estruturas foram criados para auxiliar na compreensão do relacionamento entre a seqüência e a estrutura das proteínas e dos processos evolutivos que dão origem às diferentes famílias. São exemplos de bancos de dados de dados de estruturas: o SCOP (*Structural Classification of Proteins*) [65], o CATH (*Class/Architecture/Topology/Homology*) [66] e o PDB (*Protein Data Bank*) [11].

Como pode ser observado, vários bancos de dados de proteínas estão disponíveis e podem ser utilizados na tarefa de classificação das mesmas. Com o objetivo de classificar proteínas a partir da sua composição de motivos, diversos trabalhos propostos na literatura, tais como [13, 22, 38, 70, 91, 93], utilizaram bases de dados extraídas de um banco de dados secundário, o PROSITE.

Segundo [42], o PROSITE é o banco de dados de proteínas melhor documentado, isto é, cada família de proteína está detalhadamente descrita e associada a um conjunto atualizado de referências. Além disso, por ser um banco de dados onde a manutenção e a inserção de novas informações acontecem com supervisão humana, erros e redundâncias são evitados. Devido a essas características e para que fossem possíveis comparações com técnicas de classificação utilizadas em outros trabalhos apresentados na literatura, o método proposto neste trabalho utilizou bases de dados extraídas do banco de dados PROSITE. A seguir, são apresentadas as principais características do banco de dados PROSITE e das bases de dados obtidas a partir do mesmo.

O PROSITE e as Características das Bases de Dados

O PROSITE foi criado em 1989 e sua última versão é a 20.0, a qual contém 1449 entradas (classes) com 1331 padrões e 671 perfis. As bases de dados utilizadas nesta tese foram obtidas a partir da versão 18.0 (julho/2003), contendo 1183 entradas.

O banco de dados PROSITE contém proteínas (obtidas a partir do banco de dados primário SWISS-PROT) que são agrupadas em famílias de acordo com suas funções. As famílias das proteínas são representadas por perfis ou padrões. Os perfis são tabelas de escores que armazenam informações de alinhamentos de seqüências, definindo, basicamente, quais regiões das seqüências são muito ou pouco conservadas. Diferentemente dos perfis, que tentam caracterizar as famílias das proteínas sobre toda a extensão das seqüências, padrões são pequenas seqüências de aminoácidos (codificadas como expressões regulares) que se conservam nas proteínas. Tanto os perfis como os padrões são importantes na determinação das funções biológicas de uma proteína. Eles auxiliam na atribuição de funcionalidades ao relacionar uma proteína com uma família já conhecida. Neste trabalho, perfis e padrões caracterizarão os motivos (*motifs*).

O método de classificação proposto neste trabalho utiliza bases de dados extraídas do PROSITE para predizer a família funcional de novas proteínas. Essas bases de dados armazenam informações sobre proteínas previamente estudadas (isto é, proteínas agrupadas em famílias de acordo com suas funções).

No PROSITE, cada entrada (codificada por uma expressão da forma PDOCxxxxx) representa uma classe (família funcional) de proteínas e está associada a uma função. Por exemplo, a entrada codificada como PDOC00343 corresponde à classe "*kinesin motor domain signature and profile*". Cada classe está associada a um ou mais padrões e, em alguns casos, a perfis também. Tanto os padrões como os perfis são representados por códigos da forma PSxxxxx.

Além dos dados referentes às classes e aos motivos das proteínas, o PROSITE contém outras informações que não possuem utilidade para a tarefa de classificação aqui realizada. Desse modo, uma etapa de pré-processamento foi necessária para gerar as bases de dados de proteínas com o conteúdo e formato adequados para o método proposto neste trabalho. A Figura 3.1 ilustra o formato das bases de dados geradas a partir do PROSITE. A segunda linha da figura indica, por exemplo, que a proteína O00139 possui os motivos PS00017, PS50007, PS50067 e PS50079, e pertence à classe PDOC00343.

Proteína	Motivos	Classe
O00060	PS50072	PDOC00154
O00139	PS00017 PS50007 PS50067 PS50079	PDOC00343
O02741	PS50053	PDOC00271
O08424	PS00066 PS00318 PS50065	PDOC00064
O12984	PS01031	PDOC00791
O14343	PS00017 PS50067 PS50079 PS50101	PDOC00343
O14788	PS50049 PS50099	PDOC00224
O14936	PS00856 PS50002 PS50011 PS50052 PS50106	PDOC00670

Figura 3.1: Exemplo de base de dados de proteína.

3.4 Trabalhos Relacionados

O PCP tem sido explorado por diversos grupos de pesquisa. A seguir, serão descritas algumas das principais propostas apresentadas para esse problema.

O trabalho proposto em [91] adotou árvore de decisão como método para classificar proteínas a partir da composição de motivos das suas seqüências. A partir do banco de dados PROSITE [45], bases de dados de treinamento e teste foram construídas para a realização dos experimentos computacionais. As bases de dados de treinamento foram utilizadas na construção dos modelos de classificação (árvores de decisão), enquanto as bases de dados de teste foram utilizadas na validação dos mesmos. Das 1100 classes presentes na versão do PROSITE utilizada nesse trabalho, somente 15 foram consideradas nas bases de dados utilizadas nos experimentos. Num primeiro grupo, 585 proteínas pertencentes a 10 classes foram selecionadas para a composição das bases de dados. Um segundo grupo continha 73 proteínas relacionadas com cinco classes. Uma ferramenta computacional denominada *ProfileScan* foi utilizada para determinar a composição de motivos das proteínas. Cada proteína era representada por um vetor binário, onde 1 indica a presença de um determinado motivo e 0 a ausência do mesmo. Num primeiro grupo contendo 585 proteínas, cada uma delas foi representada por um vetor binário de 82 posições (atributos), enquanto que, num segundo grupo, o vetor binário tinha 10 atributos. As proteínas selecionadas foram divididas para formar bases de dados de treinamento e teste de diferentes tamanhos. Tais bases de dados foram utilizadas na realização dos experimentos computacionais. Os resultados evidenciaram que as árvores de decisão obtidas apresentaram um desempenho satisfatório em termos de acurácia preditiva.

Uma ferramenta computacional de pré-processamento e análise de dados genéticos, denominada *GenMiner*, foi apresentada em [38]. Assim como no trabalho proposto em [91],

o objetivo era criar uma ferramenta capaz de classificar proteínas com funções desconhecidas em alguma família (classe) presente numa base de dados de treinamento (contendo proteínas com funções conhecidas). O *GenMiner* é capaz de processar dados contidos em três importantes bancos de dados de proteínas, colocando-os num formato adequado para o processamento com vários algoritmos de mineração de dados disponíveis nas ferramentas Weka (Waikato Environment for Knowledge Analysis) [95] e *MS SQL Analysis Manager 2000* [83]. O *GenMiner* foi desenvolvido em Java para facilitar sua comunicação com a ferramenta Weka, a qual foi adotada pelo fato de ser um *software* de código aberto e por fornecer uma grande variedade de algoritmos de mineração de dados. A representação computacional das proteínas na forma de vetores binários foi a mesma adotada em [91]. Bases de dados extraídas do PROSITE [45], contendo 1379, 2174 e 2599 proteínas pertencentes a 10, 18 e 28 classes, respectivamente, foram utilizadas para a construção e avaliação de árvores de decisão. Segundo os autores, apesar de apresentar boa acurácia preditiva para todas as bases de dados utilizadas nos experimentos, a ferramenta Weka mostrou-se ineficiente para o caso da base contendo 28 classes. Experimentos também foram realizados com a ferramenta *MS SQL Analysis Manager 2000*, evidenciando sua baixa capacidade para processar dados contendo um grande número de atributos.

O trabalho proposto em [70] também tratou o PCP tomando como base sua composição de motivos. Nesse trabalho, desenvolveu-se um algoritmo de indução de regras de classificação a partir de autômatos finitos. Com essa abordagem foram extraídas regras de classificação do tipo "$Motivo \rightarrow Classe$". As regras extraídas eram ordenadas de acordo a medida de interesse *interest*. A classificação dava-se por meio de comparação da proteína a ser classificada com as regras extraídas. A primeira regra a satisfazer a comparação era utilizada para classificar a proteína. O método proposto foi avaliado por meio de dois conjuntos de testes comparativos contra os resultados apresentados pelos trabalhos [91] e [38]. Além disso, foi realizado um experimento envolvendo todas as proteínas e classes de uma versão do banco de dados PROSITE [45]. Nos testes comparativos, em termos de acurácia preditiva, o método proposto apresentou desempenho superior ao dos métodos utilizados por [91] e [38]. No experimento contendo todas as proteínas e classes armazenadas numa versão do banco de dados PROSITE, aproximadamente 40000 proteínas e 1100 classes diferentes foram consideradas. Segundo os autores, para esse caso, o percentual de sucesso da classificação ficou em 41,4%. Além disso, esse experimento consumiu 7 horas e 34 minutos quando executado num Pentium IV, 1,51GHz, com 512 Mbytes de RAM.

A proposta apresentada em [93] também adotou a técnica de árvore de decisão com o objetivo de extrair regras para classificar proteínas a partir dos motivos que as compõem.

Na base de dados de treinamento, cada proteína é representada por sua composição de motivos e está rotulada com uma família funcional de acordo com o banco de dados Merops [75]. Os experimentos computacionais realizados tinham como objetivo comparar as regras construídas a partir de motivos gerados pela ferramenta computacional denominada MEME [8] com as regras obtidas a partir de motivos extraídos do banco de dados PROSITE [45]. Os resultados experimentais mostraram que as regras extraídas a partir dos motivos obtidos com a ferramenta MEME apresentaram maior acurácia preditiva do que aquelas obtidas a partir dos motivos extraídos no banco de dados PROSITE.

3.5 O Método Proposto

Como pode ser observado na Seção 3.4, a maioria dos trabalhos envolvendo mineração de dados para o PCP apresentou estratégias de classificação *eager*, nas quais um modelo é construído a partir das instâncias da base de dados de treinamento antes mesmo de se ter qualquer instância para ser classificada. Nesses casos, as novas instâncias são classificadas a partir do modelo previamente construído.

Diferentemente do que acontece na abordagem *eager*, nas estratégias de classificação *lazy*, o processamento dos dados de treinamento só acontece a partir do momento em que se tem uma instância para ser classificada, ou seja, nenhum modelo é previamente construído. Desse modo, para cada nova instância a ser classificada, toda a base de dados de treinamento é processada para que a classificação possa ser realizada. Devido a essa característica, as estratégias *lazy* são especialmente apropriadas para as situações práticas onde a base de dados de treinamento é freqüentemente atualizada [74].

Como as bases de dados biológicas têm um crescimento muito rápido, a alta taxa de atualização dessas bases de dados indica o uso de técnicas *lazy* para a tarefa de classificação. Portanto, o método proposto neste trabalho utiliza uma abordagem *lazy* para classificar proteínas a partir da análise de seus motivos.

A experiência obtida com o estudo e compreensão detalhados dos dados desse problema levou-nos a propor um método de classificação mais direcionado para a aplicação em questão. A idéia central do método é classificar uma proteína descobrindo-se qual subconjunto de seus motivos melhor caracteriza alguma classe. Desse modo, o classificador atribui à proteína a classe que melhor é descrita por um dos subconjuntos de seus motivos. Por exemplo, considere a proteína P22216 contida numa base de dados extraída do PROSITE, a qual pertence à classe PDOC50006 e é constituída pelo seguinte conjunto

de motivos $M = \{PS50006, PS50011, PS50321, PS50322\}$. Observando-se a base de dados, verifica-se que todas as proteínas que contêm os motivos $S = \{PS50006, PS50321\}$ estão rotuladas com a classe PDOC50006, ou seja, 100% dos casos estão associados com PDOC50006. Esse tipo de constatação nos leva a acreditar que a busca por subconjuntos de motivos com altas probabilidades de pertencer a uma determinada classe seja uma estratégia de classificação apropriada para esse problema.

O método proposto neste trabalho possui alguma relação com classificadores Bayesianos por também realizar a classificação tomando como base um conjunto de probabilidades a *posteriori*. Mas a principal característica do método é a classificação das proteínas com base na avaliação dos seus subconjuntos de motivos. O método proposto, denominado HiSP-Prot (***Hi****ghest* ***S****ubset* ***P****robability - for* ***Prot****ein Classification Problem*), encontra-se descrito a seguir.

Seja M_X o conjunto de motivos presentes na proteína X. De forma análoga aos classificadores Bayesianos, supondo que existam m classes, $\{C_1, C_2, \ldots, C_m\}$, X é atribuída à classe C_k, $1 \leq k \leq m$, se e somente se

$$P(C_k|X) > P(C_j|X) \quad \forall j,\ 1 \leq j \leq m, \quad j \neq k. \tag{3.1}$$

Diferentemente do classificador Bayesiano simples, calculamos $P(C_i|X)$, $1 \leq i \leq m$, como apresentado a seguir:

$$P(C_i|X) = max\{P(C_i|t)\} \quad \forall t \subseteq M_X,\ t \neq \emptyset, \tag{3.2}$$

onde

$$P(C_i|t) = \frac{P(C_i \wedge t)}{P(t)}. \tag{3.3}$$

$P(C_i \wedge t)$ corresponde à probabilidade de uma proteína pertencer à classe C_i e possuir o subconjunto de motivos t. $P(t)$ é a probabilidade de uma proteína conter o subconjunto de motivos t. Essas probabilidades são estimadas a partir da base de dados de treinamento da seguinte maneira:

$$P(C_i \wedge t) = \frac{F_{C_i t}}{N}, \tag{3.4}$$

onde $F_{C_i t}$ é o número de instâncias (proteínas) que contêm o subconjunto de motivos t e são rotuladas com a classe C_i , e N o número total de instâncias de treinamento. E

$$P(t) = \frac{F_t}{N}, \tag{3.5}$$

onde F_t é o número de instâncias que contêm o subconjunto de motivos t.

Dada a necessidade de se analisar todos os subconjuntos de motivos presentes nas proteínas, o HiSP-Prot pode apresentar um alto custo computacional se o número de motivos por proteína for elevado. Contudo, para o caso do problema abordado, pode-se concluir que a aplicação desse método é viável, uma vez que o número de motivos por proteína é reduzido. Por exemplo, na versão 18.0 do banco de dados PROSITE [45], 98% das proteínas contêm no máximo cinco motivos.

O pseudocódigo que implementa o método HiSP-Prot é apresentado na Figura 3.2. Sejam $C = \{C_1, C_2, \ldots, C_m\}$ o conjunto das classes existentes na base de dados de treinamento e $BaseDadosTreinamento = \{a_1, a_2, \ldots, a_z\}$ o conjunto de proteínas que a constituem. Cada proteína da base de dados de treinamento pertence a uma classe $C_i \in C$. M_x é o conjunto de motivos presentes na proteína x. Dada uma proteína b a ser classificada, ela será atribuída a uma classe $C_k \in C$ por meio do procedimento `CLASSIFICADOR`. Nas linhas 1, 2 e 3, as variáveis $melhorClasse$, $maiorProbabilidade$ e $melhorSubconjunto$ são inicializadas. Para cada subconjunto de motivos presentes na proteína b (linha 4), os vetores $F[t]$ (freqüência do subconjunto t) e $F[t][i]$ (freqüência do subconjunto t quando associado com classe C_i) são inicializados (linhas 5 e 6). Essas freqüências são calculadas percorrendo-se a base de dados de treinamento (linha 7 até 12). Entre as linhas 14 e 30, $P(C_i|t)$ é calculado para cada classe $C_i \in C$ (conforme mostrado na Equação 3.3), e comparações são realizadas para se determinar qual classe será atribuída à proteína b. Caso $P(C_i|t) = maiorProbabilidade$ (linha 22), ou seja, mais de um subconjunto esteja associado à maior probabilidade, o critério de desempate adotado é baseado na freqüência dos subconjuntos em suas respectivas classes (linha 23). Por fim, a classificação da proteína b é retornada na linha 31.

3.6 Avaliação do Método Proposto

Dois conjuntos de experimentos computacionais foram realizados para avaliar o desempenho do método proposto. Primeiramente, foram escolhidos os trabalhos [70] e [38] para uma avaliação comparativa, uma vez que, de acordo com nosso conhecimento, esses trabalhos apresentavam os melhores resultados experimentais para o problema de classificação de proteínas aqui abordado. A medida de desempenho utilizada na avaliação comparativa foi a acurácia preditiva, dado que somente essa medida de desempenho foi utilizada na avaliação dos classificadores propostos em [70] e [38]. Num segundo momento, com o objetivo de avaliar como o desbalanceamento de classes nas bases de dados de proteínas afeta o método proposto neste trabalho, outras medidas de desempenho foram consideradas.

```
procedimento CLASSIFICADOR(C, BaseDadosTreinamento, b)
 1: melhorClasse ← SEM CLASSE;
 2: maiorProbabilidade ← 0;
 3: melhorSubconjunto ← SEM SUBCONJUNTO;
 4: para cada subconjunto t ⊆ M_b (tal que t ≠ ∅) faça
 5:    F[t] ← 0;
 6:    F[t][i] ← 0; ∀i = 1, ..., m
 7:    para cada proteína a_r ∈ BaseDadosTreinamento faça
 8:       se (t ⊆ M_{a_r}) então
 9:          F[t][s] ← F[t][s] + 1, onde C_s é a classe da proteína a_r;
10:          F[t] ← F[t] + 1;
11:       fim se
12:    fim para
13: fim para
14: para cada classe C_i ∈ C faça
15:    para cada subconjunto t ⊆ M_b (tal que t ≠ ∅) faça
16:       P(C_i|t) ← F[t][i]/F[t];
17:       se (P(C_i|t) > maiorProbabilidade) então
18:          melhorClasse ← C_i;
19:          melhorSubconjunto ← t;
20:          maiorProbabilidade ← P(C_i|t);
21:       senão
22:          se (P(C_i|t) = maiorProbabilidade) então
23:             se (F[t][i] > F[melhorSubconjunto][melhorClasse]) então
24:                melhorClasse ← C_i;
25:                melhorSubconjunto ← t;
26:             fim se
27:          fim se
28:       fim se
29:    fim para
30: fim para
31: Retorne (melhorClasse);
fim.
```

Figura 3.2: Pseudocódigo do HiSP-Prot.

Tais medidas permitem avaliar o desempenho do classificador separadamente para cada classe de proteína.

3.6.1 Avaliação Comparativa

A avaliação do método proposto foi realizada comparando-se os resultados obtidos por meio da nossa abordagem com aqueles apresentados por outros dois classificadores: método baseado em autômatos finitos [70] e árvore de decisão [38].

As bases de dados utilizadas neste estudo, denominadas *genbase10*, *genbase18*, *gen-*

base28 e *genprosite*, contêm proteínas extraídas do banco de dados PROSITE [45]. Nessas bases de dados, cada instância corresponde a uma proteína, cada proteína é composta por um conjunto de motivos e está associada a uma classe que indica a sua família funcional. As três primeiras bases, contendo 662, 2233 e 2934 proteínas pertencentes a 10, 18 e 28 classes, respectivamente, foram cedidas pelos autores do trabalho apresentado em [70]. Com o objetivo de avaliar o método proposto num cenário mais realista, testes também foram realizados utilizando-se a base de dados *genprosite*, que contém todas as proteínas e classes armazenadas na versão 18.0 do banco de dados PROSITE, ou seja, 75384 proteínas e 1183 classes.

Apesar de a k-validação cruzada (*k-fold cross validation*) ser o método mais utilizado na estimação da acurácia de classificadores em pesquisas nas áreas de mineração de dados e aprendizado de máquina, para efeito de comparação, adotou-se o mesmo método utilizado em [70], denominado *holdout* [37]. Nessa abordagem, a base de dados é aleatoriamente dividida em dois conjuntos de dados disjuntos, denominados base de dados de treinamento e base de dados de teste. Desse modo, a base de dados de treinamento é utilizada na construção do classificador, enquanto a base de dados de teste contém as instâncias que serão utilizadas na estimativa da acurácia do classificador.

Assim como em [70], a acurácia (AC) foi utilizada como medida de desempenho do classificador. Seu cálculo é definido por

$$AC = \frac{\text{número de proteínas corretamente classificadas}}{\text{número total de proteínas na base de dados}}. \tag{3.6}$$

A estimativa da acurácia foi realizada repetindo-se dez vezes o método *holdout*, ou seja, dez testes foram aleatoriamente gerados, cada qual contendo uma base de dados de treinamento e uma de teste. Cada base de dados de teste possuía 10% da quantidade total de proteínas. O valor geral da acurácia corresponde à média dos valores obtidos individualmente nos dez testes.

O banco de dados PROSITE contém algumas proteínas que estão associadas a mais de uma família funcional. Portanto, as bases de dados utilizadas nos experimentos computacionais têm proteínas pertencendo a mais de uma classe. Nesses casos, a proteína é repetidamente representada na base de dados, já que cada representação só pode estar associada a uma classe. Essa forma de representação das proteínas apresenta uma desvantagem na avaliação do desempenho do classificador. Para ilustrar, considere uma base de dados de teste onde uma determinada proteína aparece representada k vezes, cada qual com uma classe diferente. Uma vez que a classificação dessa proteína será sempre a

mesma, independentemente do número de vezes que ela aparece representada na base de dados, considerando-se somente as k repetições dessa proteína, o valor da acurácia será no máximo igual a $1/k$. Sendo assim, para as bases de dados de teste que contêm proteínas pertencendo a mais de uma classe, a acurácia preditiva do classificador não poderá alcançar 100%.

A Tabela 3.1 apresenta os resultados da avaliação comparativa entre o HiSP-Prot e os classificadores propostos em [38] e [70]. Nessa tabela, as bases de dados de teste estão listadas na primeira coluna e os valores de acurácia (média de dez testes) para cada método são apresentados nas demais colunas. Na coluna HiSP-Prot, cada valor contido nos parênteses corresponde ao desvio-padrão da média em questão. Devido ao fato de algumas bases de dados de teste conterem proteínas que pertencem a mais de uma classe, o valor máximo de acurácia que pode ser obtido para cada base de teste é apresentado na coluna intitulada *Limite Superior*.

Tabela 3.1: Resultados da avaliação comparativa.

Base de dados	Ref. [70]	Ref. [38]	HiSP-Prot	Limite Superior
genbase10	93,71	80,13	**99,85** (0,48)	100
genbase18	94,31	84,27	**97,57** (1,12)	99,91
genbase28	75,98	74,49	**83,24** (2,18)	99,45
genprosite	—	—	**85,71** (0,49)	98,55

Os resultados mostrados na Tabela 3.1 evidenciam o bom desempenho do método proposto neste trabalho. Em relação à acurácia preditiva, o HiSP-Prot superou os métodos utilizados em [38] e [70] em todas as bases de dados testadas. É importante ressaltar que o HiSP-Prot também alcançou alta acurácia para a base de dados que contém um elevado número de proteínas e classes, a *genprosite*.

Apesar de a base de dados *genprosite* não ter sido utilizada nos experimentos realizados em [70], uma base de dados contendo todas as proteínas e classes de uma versão mais antiga do PROSITE foi considerada na avaliação de desempenho conduzida por eles. Tal base de dados era composta por aproximadamente 40000 proteínas e 1100 classes. A acurácia obtida foi de 41,4%. Segundo os autores, esse resultado deve-se principalmente ao fato de somente regras de classificação simples (do tipo "$Motivo \rightarrow Classe$") terem sido utilizadas. O uso de modelos de classificação compactos que permitissem o processamento de bases de dados maiores foi a justificativa dada para utilização de somente um motivo como antecedente das regras. Mesmo considerando somente regras simples, nos

experimentos realizados em [70], utilizando-se um Pentium IV, 1,51GHz, com 512 Mbytes de RAM, aproximadamente 7 horas e 34 minutos foram gastos para processar a base de dados mencionada anteriormente. Segundo eles, regras contendo mais de um motivo no antecedente poderiam melhorar o desempenho do classificador, mas o custo computacional desse processamento seria substancialmente maior. O HiSP-Prot apresentou um desempenho mais do que duas vezes melhor (85,71%) para uma base de dados contendo 75384 proteínas e 1183 classes, consumindo nesse experimento 1 hora e 20 minutos quando executado num AMD XP 2600+, com 512 MBytes de RAM.

Como mencionado na Seção 3.5, o HiSP-Prot pode apresentar altos custos computacionais se as proteínas forem constituídas por um grande número de motivos. Por isso, decidiu-se medir o tempo de CPU médio gasto na classificação de uma proteína. Esses experimentos também foram conduzidos numa máquina AMD XP 2600+, com 512 MBytes de RAM.

Para o caso da base de dados *genprosite*, o tempo médio de CPU gasto pelo HiSP-Prot na classificação de uma proteína foi de 0,6 segundos. Esse bom resultado é devido ao pequeno número de motivos por proteína, em média. No entanto, o desempenho em termos de consumo de tempo de CPU é diretamente influenciado pelo número motivos por proteína na base de dados de teste, uma vez que a quantidade de subconjuntos de motivos avaliados durante o processo de classificação depende do número de motivos de cada proteína.

No banco de dados PROSITE (versão 18.0), cada proteína é constituída por no máximo dez motivos, sendo o número médio de motivos por proteína igual a 1,56 com desvio-padrão de 0,9. No caso da base de dados *genprosite*, que contém todas as proteínas e classes da versão 18.0 do PROSITE, quando a proteína possui dois motivos, o HiSP-Prot gasta, em média, 0,3 segundos para classificá-la. Por outro lado, para proteínas contendo dez motivos (o pior caso na versão 18.0 do PROSITE), o HiSP-Prot gasta, em média, 108 segundos para classificá-las. Contudo, 98% das proteínas da base de dados *genprosite* contêm no máximo cinco motivos, sendo que, para esses casos, o HiSP-Prot gasta, em média, 3,2 segundos para classificar cada proteína.

Tanto a elevada acurácia preditiva quanto o bom desempenho, em termos de consumo de tempo de CPU, obtidos na tarefa de classificação, levam-nos a concluir que o método proposto neste trabalho se mostrou apropriado para o problema de classificar proteínas a partir dos seus motivos.

Recentemente, outros dois trabalhos abordaram o PCP sob diferentes perspectivas.

Em [13], realizou-se a classificação de proteínas a partir de uma rede neural artificial. Diferentemente da abordagem utilizada em nosso trabalho, nessa proposta, os motivos não correspondem diretamente aos atributos utilizados na tarefa de classificação. Nesse caso, os atributos utilizados na classificação são denominados motivos probabilísticos, pois cada proteína da base de dados é mapeada em um vetor com dimensão D, onde cada posição desse vetor contém um valor numérico (escore) relacionado com a probabilidade da proteína possuir um determinado motivo. Os experimentos foram realizados utilizando-se bases de dados extraídas do banco de dados PROSITE [45] e do GPCR (*G-protein coupled receptors*) [44]. Com o objetivo de avaliar o desempenho do classificador em diferentes cenários, diferentes tipos de bases de dados foram considerados. Em algumas bases, características estruturais das proteínas também foram utilizadas como atributos. No entanto, os experimentos mostraram que essas características estruturais não contribuíram para a melhoria de desempenho do classificador. Experimentos foram realizados para comparar o desempenho da rede neural com os resultados obtidos por outros dois métodos que também classificam proteínas a partir de motivos probabilísticos. Os resultados experimentais mostraram que a rede neural apresentou o melhor desempenho.

Já em [22], realizou-se uma avaliação comparativa entre diferentes algoritmos de classificação para o problema de classificar proteínas a partir da suas composições de motivos. Além disso, técnicas de combinações de algoritmos de classificação foram utilizadas com o objetivo de tentar melhorar o desempenho (em termos de acurácia) da classificação. Foram selecionadas dez famílias de proteínas contidas no banco de dados PROSITE [45] para a formação da base de dados utilizada nos experimentos. A base de dados formada continha 662 proteínas. Para a realização dos experimentos computacionais, foram selecionados nove algoritmos de classificação distintos. Inicialmente, cada um desses algoritmos, implementados na ferramenta Weka [95], foi individualmente avaliado. Posteriormente, cinco diferentes métodos foram utilizados para combinar os algoritmos anteriormente avaliados. A técnica *Selective Fusion*, que combina algoritmos de classificação, foi a que apresentou o melhor desempenho. Apesar de testes terem sido realizados com vários algoritmos de classificação, somente uma base de dados foi considerada nessa avaliação.

3.6.2 Avaliação do Impacto do Desbalanceamento de Classes das Bases de Dados na Classificação

Em mineração de dados, a avaliação de classificadores pode ser realizada por meio de métricas baseadas em quatro resultados possíveis: verdadeiro positivo, verdadeiro negativo,

falso positivo e o falso negativo. Para explicar o significado de cada um desses resultados numa tarefa de classificação, o seguinte exemplo será utilizado. Suponha que um classificador esteja sendo utilizado no diagnóstico de uma doença X, onde dois resultados são possíveis: doente ou saudável. Os seguintes tipos de resultados podem ser produzidos por esse classificador:

- Verdadeiro positivo: quando a pessoa que possui a doença X é classificada como doente.
- Verdadeiro negativo: quando a pessoa que não possui a doença X é classificada como saudável.
- Falso positivo: quando a pessoa que não possui a doença X é classificada como doente.
- Falso negativo: quando a pessoa que possui a doença X é classificada como saudável.

Na avaliação comparativa apresentada na Seção 3.6.1, somente a acurácia preditiva foi utilizada como medida de desempenho do classificador. Quando a acurácia é utilizada como medida de desempenho, assume-se que os diferentes tipos de erros de classificação têm igual importância. Contudo, em várias situações, pode ser importante a distinção entre os diferentes tipos de erros. Por exemplo, considerando o problema de classificação citado anteriormente, dois erros podem acontecer: o falso positivo (quando uma pessoa saudável é equivocadamente classificada como doente) ou o falso negativo (quando uma pessoa doente é erroneamente classificada como saudável). Nesse exemplo, provavelmente o falso negativo é um erro pior do que o falso positivo.

A utilização da acurácia preditiva na avaliação de desempenho de classificadores em bases de dados com distribuições de classes desbalanceadas pode ser insuficiente para a obtenção de conclusões úteis. Isso se deve ao fato de esse tipo de métrica ser tendenciosa em favor das classes majoritárias. Por exemplo, um classificador que sempre atribui a classe majoritária às instâncias pode alcançar altas taxas de acurácia quando a classe majoritária está relacionada com 99% das instâncias da base de dados. Por esse motivo, decidiu-se realizar uma segunda avaliação do HiSP-Prot utilizando-se outras duas medidas de desempenho: precisão e revocação. Tais métricas permitem a desassociação dos erros ocorridos em cada classe.

A precisão indica a capacidade do classificador de reconhecer as instâncias pertencentes à classe de interesse enquanto rejeita todas as demais. Já a revocação indica a

habilidade do classificador para identificar as instâncias pertencentes à classe de interesse [93]. Considere que, para um determinado classificador, $VP(c)$, $VN(c)$, $FP(c)$ e $FN(c)$ representem, para uma determinada classe c, o número de resultados verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, respectivamente. Desse modo, a precisão e a revocação do classificador para a classe c é dada por:

$$\text{Precisão}(c) = \frac{VP(c)}{VP(c) + FP(c)}, \text{ e} \tag{3.7}$$

$$\text{Revocação}(c) = \frac{VP(c)}{VP(c) + FN(c)}. \tag{3.8}$$

As características das bases de dados e o desempenho do HiSP-Prot segundo as medidas de precisão e revocação são mostrados na Tabela 3.2. O número médio de proteínas por classe em cada base de dados é apresentado na segunda coluna. Com o objetivo de medir o grau de desbalanceamento das classes nas bases de dados, utilizou-se o coeficiente de variação. Essa medida, mostrada na terceira coluna, corresponde à relação entre o desvio padrão e a média apresentada na segunda coluna da tabela. Para cada teste experimental, a precisão e a revocação foram obtidas calculando-se a média para todas as classes presentes na base de dados de teste. Os valores das médias de precisão e revocação são apresentados na Tabela 3.2 com seus respectivos valores de desvio padrão entre parênteses. As médias foram obtidas a partir de dez execuções independentes para cada base de dados.

Tabela 3.2: Resultados de desempenho do HiSP-Prot.

Base de dados	μ	CV (%)	Precisão (%)	Revocação (%)
genbase10	66,20	57,67	99,88 (0,37)	99,80 (0,60)
genbase18	124,06	192,01	87,63 (24,77)	86,06 (27,41)
genbase28	104,79	174,91	80,18 (31,00)	77,34 (35,76)
genprosite	63,72	176,44	84,92 (27,31)	85,60 (29,23)

Apesar de existir um considerável desbalanceamento entre as classes da base *genbase10*, os resultados médios de precisão e revocação, bem como seus desvios-padrão, indicam o bom desempenho do HiSP-Prot para todas as classes dessa base de dados. Isso pode ser observado na Tabela 3.3, onde os valores de precisão e revocação são apresentados para cada classe das bases de dados *genbase10*, *genbase18* e *genbase28*. Esses valores correspondem a uma média de dez execuções independentes para cada base de dados, sendo seus respectivos desvios-padrão apresentados entre parênteses.

Para as bases *genbase18* e *genbase28*, mesmo com um alto grau de desbalanceamento entre as classes, em geral, as precisões e as revocações médias permaneceram elevadas. No entanto, alguns valores de desvio padrão aumentaram quando comparados com aqueles obtidos para a base *genbase10*. Num primeiro momento, tais resultados de desvio padrão podem induzir à conclusão de que o desempenho do HiSP-Prot é fortemente influenciado ao trabalhar com bases desbalanceadas. Mas uma análise mais cuidadosa dos resultados mostrados na Tabela 3.3 revela que algumas classes específicas ({PDOC00552} em *genbase18* e {PDOC00010, PDOC00490 e PDOC00728} em *genbase28*) são as principais responsáveis por tais resultados.

As classes PDOC00552 em *genbase18* e PDOC00728 em *genbase28* são representadas por uma única proteína cada uma. Sendo assim, quando essas bases de dados são aleatoriamente divididas em dois conjuntos disjuntos, base de dados de treinamento e base de dados de teste, se tal proteína é alocada na base de dados de teste, torna-se impossível para o classificador predizer corretamente a classe dessa proteína, já que sua classe não está representada por nenhuma proteína na base de dados de treinamento. Por isso, nesses casos, os valores de precisão e revocação são sempre nulos, contribuindo para a degradação do desempenho do HiSP-Prot.

Por outro lado, as classes PDOC00010 e PDOC00490 em *genbase28* são compostas por várias proteínas, mas ainda assim, o HiSP-Prot apresentou baixo desempenho para as mesmas. Enquanto a classe majoritária (PDOC00027) é representada por 800 proteínas, a classe PDOC00490 é composta por apenas sete, ou seja, ela pode ser considerada uma classe minoritária. Novamente, poder-se-ia inferir que o baixo desempenho do HiSP-Prot está relacionado com o desbalanceamento de classes das bases de dados. No entanto, algumas evidências contradizem essa hipótese. Por exemplo, a classe PDOC00298 também é representada por apenas sete proteínas e, mesmo assim, o HiSP-Prot apresenta um bom desempenho, alcançando valores de precisão e revocação superiores a 93%. Além disso, apesar de a classe PDOC00010 (composta por 145 proteínas) estar longe de ser uma classe minoritária, os valores médios de precisão e revocação não alcançam 7,5%. Trabalhos anteriores que avaliaram a degradação de desempenho de diversos classificadores ao utilizarem bases de dados desbalanceadas, concluíram que o problema da queda de desempenho dos classificadores pode não estar relacionado somente com o desbalanceamento de classes das bases de dados, mas também com o grau de sobreposição existente entre as classes [69, 46]. A sobreposição está associada à semelhança entre as instâncias pertencentes às classes. Quanto mais semelhantes forem as instâncias pertencentes a classes distintas, maior será o grau de sobreposição entre essas classes. De fato, em nosso

Tabela 3.3: Distribuição das proteínas entre as classes.

Base de dados	Classe	Número de proteínas	Precisão (%)	Revocação (%)
genbase10	PDOC00064	49	100 (0,00)	100 (0,00)
	PDOC00154	79	100 (0,00)	100 (0,00)
	PDOC00224	51	100 (0,00)	100 (0,00)
	PDOC00271	62	100 (0,00)	100 (0,00)
	PDOC00343	76	98,75 (3,75)	100 (0,00)
	PDOC00561	36	100 (0,00)	100 (0,00)
	PDOC00662	41	100 (0,00)	100 (0,00)
	PDOC00670	66	100 (0,00)	100 (0,00)
	PDOC00791	171	100 (0,00)	100 (0,00)
	PDOC50007	31	100 (0,00)	98,00 (6,00)
genbase18	PDOC00020	48	100 (0,00)	100 (0,00)
	PDOC00023	60	100 (0,00)	100 (0,00)
	PDOC00027	800	98,16 (1,83)	100 (0,00)
	PDOC00335	40	100 (0,00)	96,66 (10,02)
	PDOC00340	6	80,00 (40,00)	70,00 (40,00)
	PDOC00344	8	100 (0,00)	100 (0,00)
	PDOC00552	1	0,00 (0,00)	0,00 (0,00)
	PDOC00561	36	52,83 (19,12)	84,17 (20,57)
	PDOC00564	97	94,62 (7,04)	95,99 (6,14)
	PDOC00567	23	92,67 (11,72)	100 (0,00)
	PDOC00789	14	100 (0,00)	100 (0,00)
	PDOC00790	10	100 (0,00)	100 (0,00)
	PDOC00793	771	100 (0,00)	100 (0,00)
	PDOC00800	19	100 (0,00)	100 (0,00)
	PDOC00803	37	98,00 (6,00)	91,67 (12,91)
	PDOC50001	203	98,11 (2,44)	100 (0,00)
	PDOC50006	33	95,50 (9,07)	85,17 (16,84)
	PDOC50017	27	67,50 (44,79)	25,52 (21,85)
genbase28	PDOC00010	145	7,50 (16,01)	3,48 (8,57)
	PDOC00013	149	84,67 (29,33)	25,95 (12,39)
	PDOC00021	372	68,32 (8,52)	90,52 (5,31)
	PDOC00024	25	100 (0,00)	100 (0,00)
	PDOC00027	800	77,02 (4,30)	100 (0,00)
	PDOC00031	230	87,25 (15,46)	34,16 (9,40)
	PDOC00061	42	100 (0,00)	100 (0,00)
	PDOC00067	41	100 (0,00)	100 (0,00)
	PDOC00072	18	100 (0,00)	87,50 (21,65)
	PDOC00075	21	44,44 (49,69)	34,07 (41,63)
	PDOC00160	85	91,58 (12,52)	89,71 (7,65)
	PDOC00164	25	100 (0,00)	100 (0,00)
	PDOC00166	42	100 (0,00)	100 (0,00)
	PDOC00167	36	100 (0,00)	98,33 (5,00)
	PDOC00171	587	99,16 (1,11)	100 (0,00)
	PDOC00298	7	93,75 (16,54)	100 (0,00)
	PDOC00300	62	84,98 (18,30)	100 (0,00)
	PDOC00304	14	40,00 (48,99)	23,33 (32,66)
	PDOC00305	13	86,90 (15,34)	100 (0,00)
	PDOC00307	13	100 (0,00)	100 (0,00)
	PDOC00310	57	100 (0,00)	100 (0,00)
	PDOC00485	18	95,00 (13,23)	100 (0,00)
	PDOC00486	19	100 (0,00)	100 (0,00)
	PDOC00490	7	0,00 (0,00)	0,00 (0,00)
	PDOC00495	40	98,75 (3,75)	100 (0,00)
	PDOC00499	10	85,71 (34,99)	78,57 (36,42)
	PDOC00722	55	100 (0,00)	100 (0,00)
	PDOC00728	1	0,00 (0,00)	0,00 (0,00)

caso, verificou-se que a sobreposição entre classes parece ser a principal responsável pelo baixo desempenho do HiSP-Prot nas classes PDOC00010 e PDOC00490, uma vez que essas classes possuem proteínas que são caracterizadas basicamente pelo mesmo conjunto

de motivos que compõem as proteínas associadas à classe PDOC00021.

Os resultados de precisão e revocação para cada classe da base de dados *genprosite* não foram apresentados na Tabela 3.3 devido ao grande número de classes dessa base de dados. No entanto, os valores médios de precisão e revocação dessa base são apresentados na Tabela 3.2 e indicam que o HiSP-Prot apresenta um comportamento semelhante àquele obtido para as bases *genbase18* e *genbase28*.

Os resultados apresentados na Tabela 3.3 indicam que o HiSP-Prot não é diretamente afetado pelo desbalanceamento de classes das bases de dados. Sendo assim, acreditamos que a estratégia de classificação adotada pelo HiSP-Prot, isto é, a procura por subconjuntos de motivos que tenham alta probabilidade de pertencer a determinadas classes, contribui para o seu bom desempenho mesmo para classes pouco representadas nas bases de dados, uma vez que a medida de probabilidade pode apresentar valores elevados tanto para classes minoritárias quanto para majoritárias.

3.7 HiSP-Prot com Seleção de Motivos

Na avaliação de desempenho do HiSP-Prot verificou-se a viabilidade do método em termos de desempenho e consumo de tempo de CPU para as bases extraídas do banco de dados PROSITE. Conforme mencionado na Seção 3.6.1, a quantidade de motivos por proteína na base de dados *genprosite* era no máximo igual a dez, permitindo que o HiSP-Prot realizasse a classificação das proteínas consumindo um tempo de CPU aceitável. No entanto, se o cenário fosse diferente, ou seja, se as proteínas fossem caracterizadas por um número maior de motivos, o HiSP-Prot apresentaria custos computacionais mais elevados na classificação das mesmas, podendo, em alguns casos, inviabilizar sua utilização. Por esse motivo, propôs-se uma alteração na implementação do HiSP-Prot para possibilitar sua aplicação mesmo para proteínas caracterizadas por uma quantidade maior de motivos. Essa alteração corresponde a uma etapa de pré-processamento do HiSP-Prot, cujo objetivo é selecionar os k melhores motivos (segundo uma medida de qualidade) da proteína a ser classificada antes de se iniciar o procedimento de classificação proposto no HiSP-Prot .

O parâmetro de entrada k do pré-processador do HiSP-Prot, se adequadamente escolhido, torna viável a aplicação do HiSP-Prot para a classificação de proteínas representadas por qualquer quantidade de motivos. A métrica utilizada para a seleção dos k melhores motivos de uma proteína, denominada capacidade de discriminação (CD), foi proposta em [87] como fórmula para calcular o *fitness* dos indivíduos num algoritmo genético.

A medida da capacidade de um motivo m na discriminação da classe i para todas as demais classes j ($j = 1, \ldots, n$, $j \neq i$) é calculada conforme a Equação 3.9.

$$CD(m,i) = F_{m,i} \times (1 - \frac{\sum_{j=1,j\neq i}^{n} F_{m,j}}{d-1}), \tag{3.9}$$

onde:

- $F_{m,i}$ é a freqüência relativa do motivo m na i-ésima classe;
- $F_{m,j}$ é a freqüência relativa do motivo m na j-ésima classe;
- n é o número total de classes;
- d é o número total de classes que contêm pelo menos uma ocorrência do motivo m.

Uma vez que o valor $CD(m,i)$ tenha sido calculado para todas as classes i ($i = 1, \ldots n$), a capacidade de discriminação do motivo m será dada pelo maior valor de $CD(m,i)$, ou seja,

$$CD(m) = max\{CD(m,i)\}, \forall i = 1, \ldots n \tag{3.10}$$

Dada uma proteína a ser classificada, se o número de motivos dessa proteína for maior do que k, o pré-processador do HiSP-Prot selecionará os k motivos com maiores capacidades de discriminação para serem utilizados na etapa de classificação da proteína. Caso contrário, todos os motivos da proteína serão utilizados pelo HiSP-Prot na sua classificação.

O pseudocódigo do pré-processador do HiSP-Prot é apresentado na Figura 3.3. Sejam $C = \{C_1, C_2, \ldots, C_n\}$ o conjunto das classes existentes na base de dados de treinamento e $BaseDadosTreinamento = \{a_1, a_2, \ldots, a_z\}$ o conjunto de proteínas que a constitui. Cada proteína da base de dados de treinamento pertence a uma classe $C_i \in C$. M_x é o conjunto de motivos presentes na proteína x. Dada uma proteína b a ser classificada, se o número de motivos dessa proteína for maior do que k, o procedimento `PRÉ-PROCESSADOR` será executado e somente os motivos selecionados nesse procedimento serão considerados pelo HiSP-Prot. Entre as linhas 3 e 6, para cada motivo $m \in M_b$ da proteína b, a capacidade de discriminação CD é calculada (segundo as Equações 3.9 e 3.10) e inserida na $listaMotivos$. A $listaMotivos$ é uma lista de pares $(m, CD(m))$ organizada em ordem decrescente de acordo com a capacidade de discriminação dos motivos. Na linha 7, $melhoresMotivos$ recebe os k melhores motivos (segundo o valor de CD) da proteína b.

Por fim, na linha 8, o procedimento retorna os motivos selecionados.

procedimento PRÉ-PROCESSADOR(C, $BaseDadosTreinamento$, b, k)
1: $melhoresMotivos \leftarrow \emptyset$;
2: $listaMotivos \leftarrow \emptyset$
3: **para** cada motivo $m \in M_b$ **faça**
4: $CD(m) = max\{CD(m,i)\}, \forall i = 1, \ldots n$;
5: $listaMotivos \leftarrow (m, CD(m))$
6: **fim para**
7: $melhoresMotivos \leftarrow k$ primeiros motivos de $listaMotivos$
8: Retorne ($melhoresMotivos$);
fim.

Figura 3.3: Pseudocódigo do pré-processador do HiSP-Prot.

Avaliação de Desempenho do HiSP-Prot com Seleção de Motivos

Para avaliar o desempenho do HiSP-Prot com seleção de motivos foram utilizadas as mesmas bases de dados consideradas nas avaliações anteriores, ou seja, *genbase10*, *genbase18*, *genbase28* e *genprosite*. Além disso, as mesmas partições, geradas a partir do método *holdout* para a avaliação comparativa apresentada na Seção 3.6.1, foram utilizadas nos testes aqui realizados.

O número máximo de motivos considerado na classificação de uma proteína foi cinco (k=5), dado que, para essa quantidade de motivos, o HiSP-Prot gasta poucos segundos (em média 3,2) para classificar uma proteína. A Tabela 3.4 apresenta os resultados do HiSP-Prot com e sem a seleção de motivos. O percentual de proteínas nas bases de dados de teste que possuíam mais de cinco motivos é mostrado na segunda coluna. A terceira e quarta coluna apresentam a acurácia preditiva do HiSP-Prot sem o pré-processamento e com o pré-processamento (k=5), respectivamente.

Os resultados apresentados na Tabela 3.4 mostram que a seleção de motivos (k=5) não prejudicou o desempenho do classificador, apresentando, ao contrário disso, uma acurácia preditiva ligeiramente superior à do HiSP-Prot sem o pré-processamento para a maioria das bases de dados. Essa melhoria acontece porque o pré-processamento seleciona sempre os melhores motivos da proteína. Somente para a base de dados *genprosite* a acurácia preditiva manteve-se igual para os casos com e sem a seleção de motivos. Isso se deve principalmente ao fato de essa base de dados possuir uma pequena proporção de proteínas (2,02%) representadas por mais de cinco motivos.

Tabela 3.4: Resultados experimentais do HiSP-Prot com e sem seleção de motivos.

Bases de dados	Percentual de proteínas com mais de 5 motivos	HiSP-Prot sem seleção de motivos Acurácia (%)	HiSP-Prot com seleção de motivos Acurácia (%)
genbase10	9,09	99,85 (0,48)	100 (0,00)
genbase18	6,77	97,57 (1,12)	97,89 (1,06)
genbase28	18,05	83,24 (2,18)	83,82 (2,07)
genprosite	2,02	85,71 (0,49)	85,71 (0,47)

Desse modo, pode-se concluir que, com a utilização do pré-processador apresentado nesta seção, a aplicação do HiSP-Prot torna-se viável mesmo para a classificação de proteínas caracterizadas por quantidades de motivos maiores do que aquelas utilizadas nos estudos experimentais apresentados neste trabalho. Além disso, pelo que indicam os resultados mostrados na Tabela 3.4, a utilização desse pré-processador pode inclusive contribuir para a melhoria da acurácia preditiva apresentada pelo HiSP-Prot.

Capítulo 4

Generalização do HiSP-Prot

4.1 Introdução

O método de classificação HiSP-Prot apresentado no capítulo anterior foi proposto para resolver um problema específico, o problema de classificação de proteínas. No entanto, os bons resultados obtidos com o HiSP-Prot para aquela aplicação, serviram como motivação para a proposta de uma generalização do método objetivando sua utilização com outros tipos de bases de dados e, portanto, em outras aplicações. Para efeito de diferenciação com relação à implementação inicial, a partir deste ponto, a generalização do HiSP-Prot será denominada HiSP (***Hi****ghest* ***S****ubset* ***P****robability*).

Neste capítulo, as características relativas ao tipo e formato dos dados utilizados pelo método HiSP são apresentadas na Seção 4.2. Na Seção 4.3, apresentam-se os detalhes do método e descreve-se o algoritmo que o implementa. Finalmente, na Seção 4.4, são discutidos alguns pontos importantes sobre a abordagem de classificação utilizadada pelo HiSP.

4.2 Características do Método

O método aqui proposto considera que os dados estão armazenados em uma base de dados relacional, mais especificamente em uma tabela, a qual consiste em um conjunto de n instâncias caracterizadas por z atributos distintos. Cada uma dessas n instâncias está associada a uma única classe pertencente a um conjunto de m classes conhecidas.

Os atributos das bases de dados podem ser discretos ou contínuos. Se o atributo for discreto, todos os seus possíveis valores são mapeados em um conjunto de números inteiros

positivos e consecutivos. No caso dos atributos contínuos, sua faixa de valores é discretizada em intervalos, e os intervalos também são mapeados em números inteiros positivos e consecutivos. A discretização dos atributos contínuos foi realizada utilizando-se uma técnica supervisionada proposta em [30], cuja idéia básica é, recursivamente, encontrar um ponto de corte que divida uma faixa de valores contínuos em dois subintervalos. A escolha dos pontos de cortes é feita utilizando-se a medida Ganho de Informação [37]. A cada iteração, o ponto de corte que proporcionar o maior ganho de informação é selecionado para dividir o intervalo em questão em dois subintervalos. Esse processo de divisão é repetido até que uma condição de parada seja satisfeita. A Seção 2.5.1 apresenta uma descrição mais detalha dessa técnica de discretização.

Além disso, as bases de dados podem conter instâncias caracterizadas por atributos cujos valores são desconhecidos. Dentre as possibilidades de tratamento para os valores desconhecidos de atributos apresentados na Seção 2.5.2, escolheu-se aquela que considera um valor desconhecido como uma informação útil para a etapa de classificação. Sendo assim, os valores desconhecidos de atributos são mapeados para o valor inteiro zero.

4.3 O Método Proposto

A estratégia adotada pelo HiSP é classificar uma instância a partir da avaliação dos seus subconjuntos de valores de atributos, ou seja, dada uma instância a ser classificada, os subconjuntos de valores de atributos que melhor caracterizam uma determinada classe da base de dados de treinamento são utilizados na classificação da mesma.

Seja $D = \{d_1, d_2, \ldots, d_n\}$ o conjunto de instâncias que constituem a base de dados de treinamento e $A_{d_j} = \{a_{j1}, a_{j2}, \ldots, a_{jz}\}$ o conjunto de valores de atributos que caracteriza cada instância $d_j \in D$. Se $C = \{C_1, C_2, \ldots, C_m\}$ é o conjunto das classes existentes na base de dados de treinamento, cada instância $d_j \in D$ está associada a uma classe $C_i \in C$. Seja X uma instância que se deseja classificar. Para cada classe $C_i \in C$ e para cada subconjunto de valores de atributos $t \subseteq A_X = \{x_1, x_2, \ldots, x_z\}$, calculam-se as probabilidades a *posteriori* $P(C_i|t)$ por meio da seguinte equação:

$$P(C_i|t) = \frac{P(C_i \wedge t)}{P(t)}. \tag{4.1}$$

$P(C_i \wedge t)$ corresponde à probabilidade de uma instância pertencer à classe C_i e possuir o subconjunto de valores de atributos t. $P(t)$ é a probabilidade de uma instância da base conter os valores de atributos de t. Essas probabilidades são estimadas a partir da base

de dados de treinamento da seguinte maneira:

$$P(C_i \wedge t) = \frac{F_{C_i t}}{N}, \tag{4.2}$$

onde $F_{C_i t}$ é o número de instâncias que contêm o subconjunto de valores de atributos t e são rotuladas com a classe C_i , e N é o número total de instâncias de treinamento. E

$$P(t) = \frac{F_t}{N}, \tag{4.3}$$

onde F_t é o número de instâncias que contêm o subconjunto de valores de atributos t.

A decisão de qual classe será atribuída à instância X é tomada a partir da análise de um grupo formado pelos subconjuntos de valores de atributos associados às maiores probabilidades a *posteriori* $P(C_i|t)$.

A hipótese que deu origem a essa nova estratégia de classificação foi a seguinte: dada uma instância a ser classificada, se os seus subconjuntos de valores de atributos formarem uma lista ordenada decrescentemente de acordo com as probabilidades a *posteriori* $P(C_i|t)$, espera-se que, dentre os primeiros subconjuntos que compõem essa lista, a maioria esteja associada com a classe da instância em questão. Sendo assim, a classe mais freqüente, dentre aquelas que aparecem associadas com os k primeiros subconjuntos da lista, é utilizada na classificação da nova instância. Em caso de empate, a freqüência dessas classes na base de dados de treinamento é utilizada como critério de desempate.

A utilização dessa estratégia de classificação exige a definição do número de elementos da lista de subconjuntos de valores de atributos a serem considerados na determinação da classe mais freqüente. Na estratégia aqui proposta decidiu-se considerar todos os subconjuntos cuja probabilidade a *posteriori* $P(C_i|t)$ fosse maior ou igual a um valor de *limite_ mínimo*.

O valor do *limite_ mínimo* é definido de acordo com as características de cada base de dados, uma vez que, para as bases de dados que contêm um número maior de classes e mais sobreposição de instâncias entre as mesmas, as probabilidades a *posteriori* $P(C_i|t)$ tendem a possuir valores menores. Por exemplo, numa base de dados contendo duas classes, um determinado subconjunto de valores de atributos t poderia estar associado com as duas classes, resultando em $P(C_1|t) = 0,6$ e $P(C_2|t) = 0,4$. Já numa base de dados contendo cinco classes, se esse mesmo subconjunto estivesse associado com a maioria delas, as probabilidades seriam, por exemplo, $P(C_1|t) = 0,3$, $P(C_2|t) = 0,25$, $P(C_3|t) = 0,20$, $P(C_4|t) = 0,25$ e $P(C_5|t) = 0$. Ou seja, os valores de $P(C_i|t)$ na base contendo somente duas classes tendem a ser superiores aos valores de $P(C_i|t)$ calculados

a partir da base de dados que contém cinco classes. Por esse motivo, um dos fatores considerados na definição do valor do *limite_ mínimo* em cada base de dados é o número de classes contidas na mesma.

Outro fator que deve ser levado em consideração na definição do *limite_ mínimo* é o grau de sobreposição existente entre as classes da base de dados. Quanto maior for a sobreposição entre as classes na base de dados, mais diluídas entre elas ficarão as probabilidades a *posteriori* $P(C_i|t)$. Se esse fator não for considerado no cálculo do *limite_ mínimo*, pode-se chegar a uma situação extrema onde o *limite_ mínimo* adotado numa determinada base de dados é superior a todas as probabilidades a *posteriori* calculadas para todos os subconjuntos de valores de atributos obtidos a partir de uma instância que se deseja classificar. Nesse caso, seria impossível classificar a nova instância, já que não existiria classe mais freqüente entre aquelas associadas com os subconjuntos de valores de atributos cuja $P(C_i|t) \geq$ *limite_ mínimo*.

A maneira escolhida para associar o valor do *limite_ mínimo* ao grau de sobreposição entre as classes da base de dados foi utilizar, para cada instância que se deseja classificar, o valor máximo de $P(C_i|t)$ no cálculo do *limite_ mínimo*. Desse modo, sendo $C = \{C_1, C_2, \ldots, C_m\}$ o conjunto de classes da base de dados de treinamento e considerando que o *limite_ mínimo* deve diminuir com o aumento do número de classes da base, dada uma instância X a ser classificada, caracterizada por um conjunto de valores de atributos $A_X = \{x_1, x_2, \ldots, x_z\}$, o cálculo do valor do *limite_ mínimo* é dado por:

$$limite\_\ mínimo = \frac{maxProb}{\sqrt{m}}, \tag{4.4}$$

onde:

$$maxProb = max\{P(C_i|t)\} \quad \forall t \subseteq A_X,\ t \neq \emptyset, \quad \forall C_i \in C. \tag{4.5}$$

Na Equação 4.4 pode-se observar que o valor do *limite_ mínimo* é inversamente proporcional à raiz quadrada do número de classes da base de dados de treinamento. A idéia básica é reduzir gradativamente o valor do *limite_ mínimo* na medida em que o número de classes aumenta. Em vez da raiz quadrada do número de classes, poder-se-ia ter utilizado simplesmente o número de classes como o denominador da Equação 4.4. O gráfico da Figura 4.1 mostra o comportamento do valor do *limite_ mínimo* em relação ao número de classes para dois denominadores distintos para a Equação 4.4. Pode-se observar que utilizar a raiz quadrada do número de classes como denominador da Equação 4.4 faz com que o decréscimo do valor do *limite_ mínimo* com o aumento do número de classes seja mais lento do que utilizar simplesmente o número de classes. Por esse motivo, optou-se

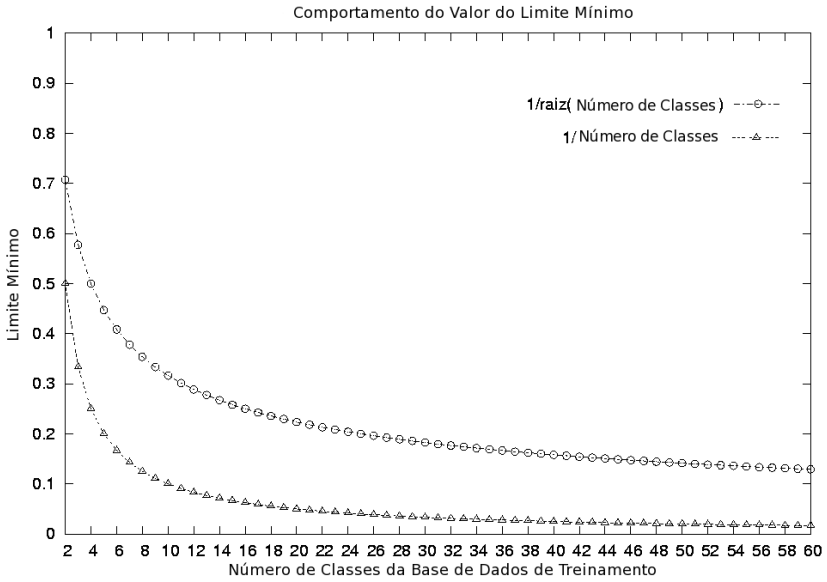


Figura 4.1: Comportamento do valor do limite mínimo.

por utilizar a raiz quadrada do número de classes como denominador.

O pseudocódigo do HiSP é apresentado na Figura 4.2. Sejam $C = \{C_1, C_2, \ldots, C_m\}$ o conjunto das classes existentes na base de dados de treinamento e *BaseDadosTreinamento* $= \{d_1, d_2, \ldots, d_n\}$ o conjunto de instâncias que a constituem. Cada instância da base de dados de treinamento pertence a uma classe $C_i \in C$. Dada uma instância X a ser classificada, ela será atribuída a uma classe $C_k \in C$ por meio do procedimento `CLASSIFICADOR`. A_X representa o conjunto de valores de atributos que caracterizam a instância X e t é um subconjunto de valores de atributos. Nas linhas 1, 2 e 3, as variáveis *melhorClasse*, *maxProb* e *limite_ mínimo* são inicializadas. Os vetores $F[t]$ (freqüência do subconjunto t) e $F[t][i]$ (freqüência do subconjunto t quando associado com classe C_i) são inicializados nas linhas 5 e 6, e a base de dados de treinamento é consultada de modo a computar essas freqüências (linha 6 até 12). Na linha 17, o cálculo da variável *maxProb* é realizado conforme a Equação 4.5, ou seja, receberá o valor da maior probabilidade a *posteriori* $P(C_i|t)$, considerando-se todas as classes $C_i \in C$ e todos os subconjuntos de valores de atributos $t \subseteq A_X$. A partir do valor de *maxProb* e do número de classes da base de dados de treinamento, o valor do *limite_ mínimo* é calculado na linha 21 de acordo com a Equação 4.4. Entre as linhas 22 e 28, as classes C_i, cujas probabilidades $P(C_i|t)$ são maiores ou iguais ao *limite_ mínimo*, são armazenadas na lista LS. Em seguida, entre as linhas 29 e 33, caso uma única classe seja a mais freqüente na lista LS, ela é atribuída à variável *melhorClasse*. Caso contrário, o critério de desempate (para as classes envolvidas no empate, escolhe-se a mais freqüente na base de dados de treinamento) é utilizado para

definir o valor de *melhorClasse*. Finalmente, a classificação da instância X é retornada na linha 34.

procedimento CLASSIFICADOR(C, $BaseDadosTreinamento$, X)
1: $melhorClasse \leftarrow$ SEM CLASSE;
2: $maxProb \leftarrow 0$;
3: $limite\_mínimo \leftarrow 0$;
4: $F[t] \leftarrow 0$; $\forall t \subseteq A_X$
5: $F[t][i] \leftarrow 0$; $\forall i = 1, \ldots, m,\ \forall t \subseteq A_X$
6: **para** cada instância $d_j \in BaseDadosTreinamento$ **faça**
7: $T = \{A_X \cap A_{d_j}\}$
8: **para** cada subconjunto $t \subseteq T$ (tal que $t \neq \emptyset$) **faça**
9: $F[t][s] \leftarrow F[t][s] + 1$, onde C_s é a classe da instância d_j;
10: $F[t] \leftarrow F[t] + 1$;
11: **fim para**
12: **fim para**
13: **para** cada subconjunto $t \subseteq A_X$ (tal que $t \neq \emptyset$) **faça**
14: **para** cada classe $C_i \in C$ **faça**
15: $P(C_i|t) \leftarrow F[t][i]/F[t]$;
16: **se** $P(C_i|t) > maxProb$ **então**
17: $maxProb \leftarrow P(C_i|t)$;
18: **fim se**
19: **fim para**
20: **fim para**
21: $limite\_mínimo \leftarrow maxProb/\sqrt{m}$;
22: **para** cada subconjunto $t \subseteq A_X$ (tal que $t \neq \emptyset$) **faça**
23: **para** cada classe $C_i \in C$ **faça**
24: **se** $(P(C_i|t) \geq limite\_mínimo)$ **então**
25: $LS \leftarrow C_i$;
26: **fim se**
27: **fim para**
28: **fim para**
29: **se** uma única classe for a mais freqüente em LS **então**
30: $melhorClasse \leftarrow$ classe mais freqüente na lista LS;
31: **senão**
32: $melhorClasse \leftarrow$ classe definida pelo critério de desempate;
33: **fim se**
34: Retorne ($melhorClasse$);
fim.

Figura 4.2: Pseudocódigo do HiSP.

4.4 Considerações Adicionais

Estratégias de classificação denominadas *eager*, como por exemplo, árvores de decisão [71], redes neurais [39] e CBA [58], utilizam os dados de treinamento para construir um modelo

que posteriormente será utilizado na classificação de novas instâncias. Diferentemente dessas, as estratégias denominadas *lazy*, como por exemplo o k-NN [20], não realizam qualquer processamento sobre os dados de treinamento enquanto não existir uma instância a ser classificada, ou seja, nenhum modelo é previamente construído.

Por adotar uma abordagem *lazy*, uma desvantagem do HiSP com relação às estratégias *eager* é o tempo gasto na classificação de uma instância, uma vez que, para cada nova instância a ser classificada, um processamento sobre toda a base de dados de treinamento se faz necessário. Desse modo, na fase de classificação, o HiSP, geralmente, consome mais tempo do que outras técnicas *eager*.

Por outro lado, enquanto as estratégias *eager* constroem um único modelo que é otimizado para, na média, obter um bom desempenho preditivo para qualquer tipo de novas instâncias, as abordagens *lazy* podem apresentar melhor desempenho preditivo por tirarem proveito de características particulares de uma nova instância, já que uma avaliação específica é realizada para cada instância que se deseja classificar [90].

Além disso, a característica *lazy* do método HiSP permite uma redução na quantidade de processamento e memória consumidos na classificação de uma instância, já que são processados somente os subconjuntos de valores de atributos específicos da instância que se deseja classificar. Se o método HiSP fosse implementado segundo uma abordagem *eager*, a necessidade de se processar todos os subconjuntos de valores de atributos existentes na base de dados de treinamento poderia consumir quantidades inviáveis de recursos computacionais.

No entanto, mesmo a estratégia *lazy* permitindo uma redução na quantidade de subconjuntos a serem avaliados pelo HiSP, dependendo das dimensões da base de dados e das características da instância que se deseja classificar, o custo computacional do processamento dos subconjuntos provenientes da instância em questão pode ser muito elevado.

Desse modo, para tornar viável a utilização do HiSP com bases de dados de qualquer dimensão, em alguns casos, pode ser necessária uma etapa de pré-processamento da base de dados visando a redução do seu número de atributos. No caso dos experimentos computacionais realizados neste trabalho, geralmente as bases de dados contendo mais de 15 atributos sofreram uma redução do número de atributos antes de serem utilizadas pelo HiSP. Para tanto, diversas técnicas de seleção de atributos propostas na literatura foram utilizadas. Portanto, mesmo para bases de dados com grandes dimensões, o HiSP passa a ser aplicável se uma etapa de pré-processamento realizar a redução do número de atributos das mesmas.

Capítulo 5

Resultados Experimentais

5.1 Introdução

Neste capítulo, apresentam-se os experimentos realizados com o intuito de avaliar o desempenho do HiSP com relação à acurácia, tempo computacional e escalabilidade. Primeiramente, a acurácia preditiva foi o critério utilizado na avaliação comparativa entre o método proposto e outras importantes técnicas de classificação. Em seguida, avaliou-se também o desempenho do HiSP com relação ao tempo computacional, ou seja, o tempo de CPU gasto para classificar uma instância. Por fim, um estudo foi realizado com o objetivo de verificar se o método é escalável com relação ao número de instâncias da base de dados.

A avaliação comparativa foi realizada utilizando-se quarenta bases de dados de domínio público freqüentemente utilizadas em experimentos de mineração de dados, obtidas no repositório de dados *UCI Machine Learning Repository* [12]. Essas bases de dados estão relacionadas com diferentes aplicações e, portanto, variam em termos de conteúdo, número de instâncias, número de atributos e número de classes.

A avaliação da acurácia preditiva dos classificadores foi realizada utilizando-se a k-validação cruzada (com k=10), sendo cada partição obtida de forma aleatória. Portanto, para cada base de dados, o valor da acurácia corresponde a uma média dos valores obtidos em cada uma das dez partições. Vale ressaltar que as mesmas partições foram utilizadas por todos os métodos envolvidos nos experimentos. Todos os experimentos foram realizados em uma máquina AMD XP 2600+, com 512 MB de RAM.

O restante deste capítulo está organizado da seguinte forma. A Seção 5.2 contém a avaliação comparativa entre o método proposto nesta tese e outras técnicas de classifi-

cação. Em seguida, os experimentos para verificação de escalabilidade do método e seu desempenho com relação ao tempo computacional, são apresentados na Seção 5.3.

5.2 Avaliação Comparativa

O HiSP foi comparado com as seguintes técnicas: árvore de decisão, k-NN, classificação Bayesiana simples e classificação associativa. Os experimentos envolvendo os métodos árvore de decisão, k-NN e classificação Bayesiana foram realizados utilizando-se os algoritmos J48, IBk (com k=1, 3 e 5) e *NaiveBayes*, respectivamente, implementados na ferramenta Weka [95]. No caso da classificação associativa, o algoritmo utilizado foi o CBA [Versão 2.0] (*Classification Based on Association*) [58], cuja implementação foi fornecida por um dos seus autores.

As execuções dos algoritmos J48 e *NaiveBayes* foram realizadas com os seguintes parâmetros. No caso do J48, o parâmetro *confidenceFactor*, usado na etapa de poda da árvore de decisão, ficou ajustado em 0,25, e o parâmetro *minNumObj*, que define o número mínimo de instâncias para cada nó terminal da árvore, recebeu o valor 2. Já para o algoritmo *NaiveBayes*, o parâmetro *useKernelEstimator* ficou ajustado com o valor *False*, indicando que a distribuição normal deveria ser considerada para os atributos contínuos, e o parâmetro *useSupervisedDiscretization* recebeu o valor *False*, definindo que os atributos contínuos não deveriam ser discretizados. Para o CBA, os parâmetros adotados foram os mesmos utilizados pelos seus autores nos experimentos apresentados em [58], os quais mostraram a sua superioridade com relação a outras técnicas de classificação. Os valores adotados para os parâmetros foram *suporte mínimo* = 1%, *confiança mínima=50%* e *número máximo de regras=80000*.

Conforme mencionado na Seção 4.4, dependendo das dimensões da base de dados e das características da instância que se deseja classificar, o HiSP pode apresentar um custo computacional elevado para processar os subconjuntos de valores de atributos provenientes da instância a ser classificada, inviabilizando, em alguns casos, a aplicação do método. Desse modo, para tornar viável a utilização do HiSP com bases de dados de quaisquer dimensões, em alguns casos, foi necessária uma etapa de pré-processamento da base de dados visando a redução do seu número de atributos. No caso dos experimentos computacionais realizados neste trabalho, as bases de dados foram divididas em dois grupos:

- Grupo 1: composto por 22 bases que não sofreram redução do número de atributos

antes de serem processadas pelo HiSP. A maioria das bases de dados desse grupo contém menos de 16 atributos (sem contar o atributo classe).

- Grupo 2: formado por 18 bases que sofreram uma redução do número de atributos antes de serem utilizadas pelo HiSP. As bases de dados que compõem esse grupo, originalmente, continham mais de 15 atributos (sem considerar o atributo classe).

Os resultados experimentais para as bases de dados pertencentes ao Grupo 1 são mostrados na Tabela 5.1. As bases de dados são listadas na primeira coluna e suas características (número de instâncias, número de atributos desconsiderando o atributo classe e o número de classes) são descritas na segunda coluna. Da terceira coluna até a oitava, são apresentados os resultados médios de acurácia obtidos pelos algoritmos J48, IBk (k=1), IBk (k=3), IBk (k=5), NB (*NaiveBayes*) e CBA. Da mesma forma, os resultados médios de acurácia para o HiSP são mostrados na última coluna com seus respectivos desvios-padrão entre colchetes. Para cada base de dados apresentada nessa tabela, o valor em negrito representa o melhor valor de acurácia entre todos aqueles obtidos pelas técnicas de classificação utilizadas.

Conforme apresentado na última coluna da Tabela 5.1, com exceção da base de dados Shuttle-landing, cujo desvio-padrão foi de 42,16%, para todas a demais bases esse valor ficou entre 0% e 11,59%. O valor mais elevado de desvio-padrão para a base de dados Shuttle-landing deve-se ao reduzido número de instâncias nas bases de dados de teste (uma ou duas instâncias).

Na última linha da Tabela 5.1, são apresentados os resultados médios de acurácia para cada uma das técnicas. Entre todas elas, o HiSP obteve o melhor resultado médio de acurácia, alcançando o valor de 79,06%. Listando os algoritmos que implementam as demais técnicas em ordem decrescente de acurácia média, tem-se: IBk (k=1) (78,28%), NB (78,22%), IBk (k=3) (78,20%), IBk (k=5) (77,50%), J48 (76,69%) e CBA (75,68%).

A Tabela 5.2 apresenta os resultados de uma análise comparativa entre o HiSP e cada uma das outras técnicas utilizadas nos experimentos. As linhas dessa tabela indicam quantas vezes o HiSP obteve resultados de acurácia superiores (linha 1), iguais (linha 2) e inferiores (linha 3) aos das demais técnicas para as 22 bases de dados do Grupo 1. Por exemplo, percorrendo-se as linhas dessa tabela para a coluna IBk ($k = 3$), observa-se que o HiSP obteve resultados de acurácia superiores aos do IBk ($k = 3$) para 16 bases de dados, igual para uma base e inferiores para outras 5 bases.

Para as 22 bases de dados utilizadas neste experimento, os dados apresentados na

Tabela 5.1: Resultados de acurácia para as bases de dados do Grupo 1.

Bases de Dados	**Número de instâncias, número de atributos, número de classes**	**J48**	**IBk (k=1)**	**IBk (k=3)**	**IBk (k=5)**	**NB**	**CBA**	**HiSP**
Balance-scale	625, 4, 3	69,13	69,29	69,29	69,29	72,17	71,85	**73,61** [4,00]
Breast-cancer	286, 9, 2	73,50	67,19	70,33	73,45	71,74	66,48	**74,88** [9,97]
Breast-w	699, 9, 2	94,57	96,57	**97,14**	97,00	**97,14**	95,43	96,71 [2,03]
Credit-a	690, 15, 2	**87,39**	83,77	85,22	85,51	86,38	85,23	87,10 [4,29]
Diabetes	768, 8, 2	77,06	76,42	77,46	77,85	77,59	76,52	**77,97** [6,41]
Glass	214, 9, 6	73,46	**79,46**	77,62	74,87	73,94	76,67	74,42 [8,46]
Hayes-roth	160, 4, 3	**53,75**	**53,75**	**53,75**	**53,75**	**53,75**	**53,73**	**53,75** [7,34]
Heart-cleveland	303, 13, 2	77,83	81,84	82,16	82,82	83,14	82,12	**83,48** [3,90]
Heart-hungarian	294, 13, 2	79,57	82,68	83,68	83,32	84,01	83,04	**83,69** [11,59]
Iris	150, 4, 3	94,00	92,67	**94,67**	**94,67**	**94,67**	93,32	93,33 [4,44]
Labor	57, 16, 2	88,00	96,33	91,33	87,67	98,00	89,33	**100,00** [0,00]
Liver-disorders	345, 6, 2	**63,23**	**63,23**	**63,23**	**63,23**	**63,23**	**63,23**	57,96 [6,18]
Postoperative	90, 8, 3	70,00	62,22	67,78	70,00	68,89	61,13	**71,11** [9,37]
Primary-tumor	339, 17, 21	43,40	39,23	44,53	46,60	**48,39**	39,83	45,72 [11,36]
Shuttle-landing	15, 6, 2	50,00	70,00	60,00	45,00	**75,00**	55,00	70,00 [42,16]
Solar-flare1	323, 12, 6	70,26	66,19	65,94	67,18	65,00	**70,58**	69,03 [5,87]
Solar-flare2	1066, 12, 6	**74,58**	73,08	74,02	73,83	74,02	34,27	74,29 [3,37]
Statlog-heart	270, 13, 2	81,85	83,33	81,48	81,85	83,33	**83,72**	83,70 [5,00]
Tic-tac-toe	958, 9, 2	84,77	98,75	98,75	98,75	69,62	**99,07**	78,61 [5,47]
Vote	435, 16, 2	**95,64**	92,41	92,41	91,73	89,87	93,56	94,50 [3,77]
Wine	178, 13, 3	92,09	97,78	96,63	95,52	98,89	97,73	**99,44** [1,76]
Zoo	101, 17, 7	93,18	**96,00**	93,09	91,09	92,18	93,09	**96,00** [5,16]
	Média	76,69	78,28	78,20	77,50	78,22	75,68	**79,06**

Tabela 5.2: Comparação do HiSP com as outras técnicas.

		J48	IBk ($k = 1$)	IBk ($k = 3$)	IBk ($k = 5$)	NB	CBA
1	Resultados Superiores	14	16	16	15	15	16
2	Resultados Iguais	1	3	1	1	1	1
3	Resultados Inferiores	7	3	5	6	6	5

Tabela 5.2 confirmam que, em termos de acurácia preditiva, o HiSP é competitivo, e freqüentemente superior às demais técnicas utilizadas neste estudo.

Nos experimentos realizados com as bases de dados pertencentes ao Grupo 2, três técnicas de seleção de atributos foram escolhidas para reduzir o número de atributos dessas bases. Essas técnicas, comumente utilizadas em trabalhos de mineração de dados, geralmente realizam a seleção combinando heurísticas de busca e métricas de avaliação de atributos. A Seção 2.5.3 apresenta uma revisão das principais técnicas de seleção de atributos. As técnicas utilizadas neste trabalho, conhecidas como *Correlation-based Feature Selection*, *Consistency-based Feature Selection* e *Information Gain Attribute Ranking*, encontram-se implementadas na ferramenta Weka [95] denominadas por *CfsSubsetEval*, *ConsistencySubsetEval* e *InfoGainAttributeEval*, respectivamente. As duas primeiras técnicas foram utilizadas com diferentes heurísticas de busca, também implementadas na ferramenta Weka.

A técnica *Correlation-based Feature Selection* foi utilizada em conjunto com a heurística *Best First*, denominada na ferramenta Weka por *BestFirst*. Os valores *default* foram utilizados para os parâmetros dessa heurística, exceto para o parâmetro *searchTermination*, cujo valor foi ajustado para 50. Esse parâmetro está associado ao critério de parada, definindo a máxima quantidade de iterações consecutivas sem que melhores resultados sejam alcançados.

A técnica *Consistency-based Feature Selection* foi utilizada em conjunto com a heurística *Stepwise Selection*, denominada na ferramenta Weka por *GreedyStepwise*. Os valores *default* foram utilizados para os parâmetros dessa heurística.

Já a técnica *Information Gain Attribute Ranking*, em vez de uma heurística de busca, foi utilizada em conjunto com um algoritmo denominado *Ranker*, cuja função é selecionar os m melhores atributos de acordo com a métrica Ganho de Informação.

Os resultados de acurácia referentes às bases do Grupo 2 são apresentados em seis tabe-

las: experimentos com redução do número de atributos por meio da técnica *Correlation-based Feature Selection* (Tabelas 5.4 e 5.5), experimentos com redução do número de atributos por meio da técnica *Consistency-based Feature Selection* (Tabelas 5.7 e 5.8) e aqueles com redução do número de atributos por meio da técnica *Information Gain Attribute Ranking* (Tabelas 5.10 e 5.11). Embora o HiSP, nesses experimentos, tenha sido aplicado somente a partir das bases reduzidas (resultantes da seleção de atributos), para as demais técnicas utilizadas neste trabalho, os experimentos foram realizados tanto com as bases reduzidas como com as bases de dados originais (bases que não sofreram redução pelas técnicas de seleção de atributos). Desse modo, as Tabelas 5.4, 5.7 e 5.10 apresentam os resultados de acurácia para os experimentos realizados com as bases de dados reduzidas (para todas as técnicas) e as Tabelas 5.5, 5.8 e 5.11 os resultados para os experimentos realizados com as bases de dados originais (exceto para o HiSP, que utilizou as bases reduzidas).

A Tabela 5.3 mostra o número de atributos selecionados em cada base de dados pelas técnicas adotadas neste trabalho. No caso das técnicas *Correlation-based Feature Selection* e *Consistency-based Feature Selection*, o número de atributos selecionados foi determinado pelas suas próprias heurísticas de busca. Já no caso da técnica *Information Gain Attribute Ranking*, como o número de atributos é um parâmetro de entrada, adotou-se o valor 15 para todas as bases, exceto para as três bases com maiores quantidades de instâncias, cujo parâmetro foi ajustado para 10 ou 13. Esses valores foram escolhidos para garantir ao HiSP um baixo consumo de tempo computacional na classificação de uma instância.

Tabela 5.3: Número de atributos selecionados.

Bases de Dados	Número de instâncias, número de atributos, número de classes	Número de Atributos Selecionados		
		Correlation based Feature Selection	Consistency based Feature Selection	Information Gain Attribute Ranking
Anneal	898, 38, 5	7	8	15
Audiology	226, 69, 24	6	13	15
Autos	205, 25, 6	5	6	15
Chess-Kr-vs-Kp	3196, 36, 2	3	6	15
Flags	194, 29, 8	5	8	15
Hepatitis	155, 19, 2	8	9	15
Horse-colic	368, 27, 2	4	3	15
Ionosphere	351, 34, 2	13	7	15
Letter-recognition	20000, 16, 26	9	13	10
Lymph	148, 18, 4	9	9	15
Mol-bio-promoters	106, 58, 2	4	4	15
Mol-bio-splice	3190, 61, 3	6	10	15
Pendigits	10992, 16, 10	11	10	10
Soybean-large	683, 35, 19	14	13	15
Spambase	4601, 57, 2	10	16	15
Statlog-segment	2310, 19, 7	6	9	15
Statlog-Vehicle	846, 18, 4	9	16	15
Waveform-5000	5000, 40, 3	15	12	13

Em cada uma das tabelas de resultados das bases de dados do Grupo 2, os nomes das bases encontram-se listados na primeira coluna e suas respectivas características (número de instâncias, número de atributos desconsiderando o atributo classe e o número de classes) são descritas na segunda coluna. Da terceira coluna até a oitava são apresentados os resultados médios de acurácia obtidos pelos algoritmos J48, IBk (k=1), IBk (k=3), IBk (k=5), NB (*NaiveBayes*) e CBA, respectivamente. Por fim, na última coluna, são apresentados os resultados médios de acurácia do HiSP (com seus respectivos desvios-padrão entre colchetes) para as bases de dados reduzidas. Para cada base de dados apresentada nessas tabelas, o valor em negrito representa o melhor valor de acurácia entre todos aqueles obtidos pelas técnicas de classificação utilizadas. A última linha dessas tabelas contém o valor médio de acurácia para cada um dos métodos utilizados nesses experimentos.

Os resultados apresentados última linha das Tabelas 5.4 e 5.5 mostram que, na média, o HiSP apresentou desempenho, em termos de acurácia preditiva, superior ao de todas as demais técnicas, independentemente de elas terem utilizado as bases de dados reduzidas (com seleção de atributos) ou as bases originais (sem seleção de atributos). O valor médio de acurácia obtido pelo HiSP a partir das bases de dados reduzidas foi de 85,93%. Para os demais algoritmos, os seguintes valores foram alcançados (em ordem decrescente de desempenho): IBk (k=1) com seleção de atributos (85,26%), IBk (k=3) com seleção de atributos (84,41%), IBk (k=1) sem seleção de atributos (84,38%), J48 sem seleção de atributos (83,95%), NB com seleção de atributos (83,73%), IBk (k=5) com seleção de atributos (83,61%), IBk (k=3) sem seleção de atributos (83,57%), IBk (k=5) sem seleção de atributos (82,98%), J48 com seleção de atributos (82,65%), NB sem seleção de atributos (82,49%), CBA com seleção de atributos (78,97%) e CBA sem seleção de atributos (78,83%).

Uma análise comparativa entre o HiSP e cada uma das outras técnicas utilizadas nesses experimentos é apresentada na Tabela 5.6. As linhas dessa tabela mostram para quantas bases de dados do Grupo 2 o HiSP atingiu acurácias superiores (linha 1), iguais (linha 2) e inferiores (linha 3) às obtidas pelas demais técnicas. Por exemplo, os valores contidos na coluna NB (sem seleção de atributos) indicam que o HiSP teve desempenho superior à classificação Bayesiana (NB) para 14 bases de dados, não apresentou desempenho igual para nenhuma base, e foi inferior em outras 4 bases de dados.

Os resultados apresentados na Tabela 5.6, que foram obtidos a partir da técnica de seleção de atributos *Correlation-based Feature Selection*, evidenciam que o HiSP apresenta, na maioria das vezes, desempenho superior ao das demais técnicas utilizadas neste

Tabela 5.4: Resultados de acurácia para as bases de dados do Grupo 2 (todos os métodos utilizaram as bases reduzidas a partir da técnica *Correlation-based Feature Selection*).

Bases de Dados	**Número de instâncias, número de atributos, número de classes**	**Com seleção de atributos**						
		J48	**IBk (k=1)**	**IBk (k=3)**	**IBk (k=5)**	**NB**	**CBA**	**HiSP**
Anneal	898, 38, 5	97,22	**97,77**	96,55	96,10	96,33	96,69	95,99 [1,97]
Audiology	226, 69, 24	**69,94**	69,43	67,69	65,06	66,82	63,78	66,34 [10,88]
Autos	205, 25, 6	78,05	**86,69**	76,57	74,64	79,40	76,06	81,40 [6,47]
Chess-Kr-vs-Kp	3196, 36, 2	**90,43**	**90,43**	**90,43**	**90,43**	**90,43**	90,42	**90,43** [1,15]
Flags	194, 29, 8	56,18	58,74	**61,34**	59,79	59,87	60,24	60,84 [5,35]
Hepatitis	155, 19, 2	81,17	85,75	86,38	85,00	86,38	86,33	**86,96** [7,00]
Horse-colic	368, 27, 2	85,07	84,52	84,23	84,78	**87,50**	86,97	83,16 [5,94]
Ionosphere	351, 34, 2	92,60	91,46	91,47	91,48	92,32	**93,72**	93,46 [4,44]
Letter-recognition	20000, 16, 26	79,37	90,22	87,80	86,41	72,96	3,54	**91,73** [0,54]
Lymph	148, 18, 4	76,38	83,24	82,52	81,81	82,38	82,48	**83,86** [11,04]
Mol-bio-promoters	106, 58, 2	73,45	87,64	90,55	89,55	**94,27**	89,45	93,27 [7,89]
Mol-bio-splice	3190, 61, 3	93,17	89,78	88,43	88,03	**93,54**	92,66	93,42 [1,89]
Pendigits	10992, 16, 10	87,72	95,32	94,75	94,22	87,56	76,41	**96,24** [0,51]
Soybean-large	683, 35, 19	**91,95**	91,07	89,32	87,26	90,48	88,57	91,35 [2,99]
Spambase	4601, 57, 2	91,31	92,07	91,78	91,63	91,72	**92,63**	92,33 [1,16]
Statlog-segment	2310, 19, 7	95,80	95,67	93,55	92,03	93,07	92,24	**95,89** [1,35]
Statlog-Vehicle	846, 18, 4	**71,39**	69,85	67,02	66,31	61,10	67,49	66,30 [4,88]
Waveform-5000	5000, 40, 3	76,52	75,06	79,06	80,50	80,98	81,78	**83,74** [1,52]
	Média	82,65	85,26	84,41	83,61	83,73	78,97	**85,93**

Tabela 5.5: Resultados de acurácia para as bases de dados do Grupo 2 (somente o HiSP utilizou as bases reduzidas a partir da técnica *Correlation-based Feature Selection*).

Bases de Dados	**Número de instâncias, número de atributos, número de classes**	**Sem seleção de atributos**						**Com seleção de atributos**
		J48	**IBk (k=1)**	**IBk (k=3)**	**IBk (k=5)**	**NB**	**CBA**	**HiSP**
Anneal	898, 38, 5	98,78	**99,22**	97,89	96,77	94,88	98,01	95,99 [1,97]
Audiology	226, 69, 24	**78,75**	71,25	60,95	58,70	66,80	70,75	66,34 [10,88]
Autos	205, 25, 6	78,48	**85,33**	78,93	76,95	72,02	78,58	81,40 [6,47]
Chess-Kr-vs-Kp	3196, 36, 2	**99,41**	96,03	96,56	96,18	87,70	98,78	90,43 [1,15]
Flags	194, 29, 8	**62,53**	56,74	57,87	61,47	60,89	57,27	60,84 [5,35]
Hepatitis	155, 19, 2	76,00	83,08	**86,96**	84,96	84,54	81,87	**86,96** [7,00]
Horse-colic	368, 27, 2	**86,96**	77,70	75,83	74,22	81,80	81,87	83,16 [5,94]
Ionosphere	351, 34, 2	90,02	93,17	90,61	89,19	90,31	**93,72**	93,46 [4,44]
Letter-recognition	20000, 16, 26	78,85	**91,78**	90,42	89,86	74,02	3,87	91,73 [0,54]
Lymph	148, 18, 4	76,33	85,19	82,48	82,43	**85,76**	79,01	83,86 [11,04]
Mol-bio-promoters	106, 58, 2	75,36	82,18	83,00	79,09	89,64	71,90	**93,27** [7,89]
Mol-bio-splice	3190, 61, 3	94,33	74,51	77,34	79,59	**95,20**	91,70	93,42 [1,89]
Pendigits	10992, 16, 10	88,23	**97,18**	96,84	96,58	87,90	83,91	96,24 [0,51]
Soybean-large	683, 35, 19	**93,27**	91,95	91,65	90,62	89,45	90,05	91,35 [2,99]
Spambase	4601, 57, 2	93,18	92,89	93,15	93,22	90,24	**93,34**	92,33 [1,16]
Statlog-segment	2310, 19, 7	95,15	94,16	93,68	92,86	91,65	93,90	**95,89** [1,35]
Statlog-Vehicle	846, 18, 4	68,90	**72,10**	71,38	71,02	61,34	69,01	66,30 [4,88]
Waveform-5000	5000, 40, 3	76,62	74,30	78,70	79,86	80,72	81,34	**83,74** [1,52]
	Média	83,95	84,38	83,57	82,98	82,49	78,83	**85,93**

Tabela 5.6: Comparação do HiSP com as outras técnicas.

		Sem seleção de atributos						Com seleção de atributos					
		J48	IBk $k=1$	IBk $k=3$	IBk $k=5$	NB	CBA	J48	IBk $k=1$	IBk $k=3$	IBk $k=5$	NB	CBA
1	Res. Sup.	9	9	11	12	14	12	12	12	12	14	12	13
2	Res. Iguais	0	0	1	0	0	0	1	1	1	1	1	0
3	Res. Inf.	9	9	6	6	4	6	5	5	5	3	5	5

trabalho. No pior caso, o HiSP obteve desempenho equivalente ao de outras técnicas de classificação. Isso pode ser observado a partir dos resultados apresentados na segunda e terceira colunas (árvore de decisão (J48) e IBk ($k = 1$) – sem seleção de atributos) da Tabela 5.6, os quais indicam que o HiSP é equiparável a essas técnicas em termos de número de bases de dados para as quais apresentaram melhores desempenhos. Tanto no caso do J48 quanto para o IBk ($k = 1$) – sem seleção de atributos –, o HiSP apresentou melhor desempenho para 9 bases de dados e pior para outras 9 bases. Apesar de essas técnicas serem equivalentes ao HiSP em relação ao número de bases de dados para as quais apresentam melhores acurácias, os resultados da última linha da Tabela 5.5 mostram que, em termos de acurácia média, o HiSP (85,93%) apresenta desempenho superior ao do J48 – sem seleção de atributos (83,95%) – e ao do IBk ($k = 1$) – sem seleção de atributos (84,38%).

As Tabelas 5.7 e 5.8 apresentam os resultados de acurácia para as bases que sofreram redução do número de atributos por meio da técnica *Consistency-based Feature Selection*. Assim como na Tabela 5.5, são apresentados na Tabela 5.8 os resultados de acurácia preditiva para todos os algoritmos (exceto para o HiSP) executados a partir das bases de dados originais (sem seleção de atributos).

O resultado médio de acurácia alcançado pelo HiSP a partir das bases de dados que sofreram redução do número de atributos por meio da técnica *Consistency-based Feature Selection*, também foi superior ao de todas as demais técnicas. Como pode ser observado na última linha das Tabelas 5.7 e 5.8, o HiSP obteve uma acurácia média de 85,87%, enquanto os outros algoritmos alcançaram os seguintes valores (em ordem decrescente de desempenho): IBk (k=1) sem seleção de atributos (84,38%), IBk (k=1) com seleção de atributos (84,34%), J48 sem seleção de atributos (83,95%), IBk (k=3) sem seleção de atributos (83,57%), IBk (k=5) sem seleção de atributos (82,98%), IBk (k=3) com seleção de atributos (82,53%), NB sem seleção de atributos (82,49%), J48 com seleção de atributos (82,37%), NB com seleção de atributos (81,80%), IBk (k=5) com seleção de atributos (81,55%), CBA sem seleção de atributos (78,83%) e CBA com seleção de

Tabela 5.7: Resultados de acurácia para as bases de dados do Grupo 2 (todos os métodos utilizaram as bases reduzidas a partir da técnica *Consistency-based Feature Selection*).

Bases de Dados	**Número de instâncias, número de atributos, número de classes**	**Com seleção de atributos**						
		J48	**IBk (k=1)**	**IBk (k=3)**	**IBk (k=5)**	**NB**	**CBA**	**HiSP**
Anneal	898, 38, 5	99,00	**99,67**	98,55	98,00	97,89	98,46	97,11 [1,67]
Audiology	226, 69, 24	75,65	**77,43**	68,08	65,83	67,75	69,00	74,35 [7,04]
Autos	205, 25, 6	76,05	**86,33**	73,05	65,29	75,52	78,48	83,40 [7,68]
Chess-Kr-vs-Kp	3196, 36, 2	**94,34**	**94,34**	**94,34**	**94,34**	**94,34**	**94,34**	**94,34** [1,23]
Flags	194, 29, 8	59,29	59,42	60,39	57,29	59,84	55,26	**60,97** [9,28]
Hepatitis	155, 19, 2	83,13	87,63	84,33	84,96	86,38	88,24	**90,17** [7,94]
Horse-colic	368, 27, 2	66,35	**71,19**	66,59	66,33	64,65	66,64	69,58 [7,17]
Ionosphere	351, 34, 2	90,60	90,32	91,46	91,46	90,90	**92,00**	90,61 [4,82]
Letter-recognition	20000, 16, 26	79,15	92,21	90,51	89,57	74,60	3,87	**94,08** [0,44]
Lymph	148, 18, 4	77,05	75,76	77,76	75,10	79,00	79,77	**81,10** [10,45]
Mol-bio-promoters	106, 58, 2	78,27	86,00	83,91	84,91	**91,45**	86,72	89,73 [10,18]
Mol-bio-splice	3190, 61, 3	93,79	86,68	86,65	86,52	**94,45**	92,74	94,42 [1,30]
Pendigits	10992, 16, 10	87,55	93,90	93,25	92,59	85,26	49,47	**95,12** [0,53]
Soybean-large	683, 35, 19	**91,36**	87,27	84,03	83,75	84,18	86,11	88,72 [2,00]
Spambase	4601, 57, 2	91,78	92,00	92,13	91,81	88,87	92,02	**93,13** [0,92]
Statlog-segment	2310, 19, 7	95,32	94,46	92,94	91,90	93,29	93,47	**96,06** [1,09]
Statlog-Vehicle	846, 18, 4	69,25	**71,98**	71,74	71,03	63,11	71,49	71,50 [5,79]
Waveform-5000	5000, 40, 3	74,76	71,56	75,88	77,20	80,92	78,46	**81,32** [1,72]
	Média	82,37	84,34	82,53	81,55	81,80	76,47	**85,87**

Tabela 5.8: Resultados de acurácia para as bases de dados do Grupo 2 (somente o HiSP utilizou as bases reduzidas a partir da técnica *Consistency-based Feature Selection*).

Bases de Dados	**Número de instâncias, número de atributos, número de classes**	**Sem seleção de atributos**						**Com seleção de atributos**
		J48	**IBk (k=1)**	**IBk (k=3)**	**IBk (k=5)**	**NB**	**CBA**	**HiSP**
Anneal	898, 38, 5	98,78	**99,22**	97,89	96,77	94,88	98,01	97,11 [1,67]
Audiology	226, 69, 24	**78,75**	71,25	60,95	58,70	66,80	70,75	74,35 [7,04]
Autos	205, 25, 6	78,48	**85,33**	78,93	76,95	72,02	78,58	83,40 [7,68]
Chess-Kr-vs-Kp	3196, 36, 2	**99,41**	96,03	96,56	96,18	87,70	98,78	94,34 [1,23]
Flags	194, 29, 8	**62,53**	56,74	57,87	61,47	60,89	57,27	60,97 [9,28]
Hepatitis	155, 19, 2	76,00	83,08	86,96	84,96	84,54	81,87	**90,17** [7,94]
Horse-colic	368, 27, 2	**86,96**	77,70	75,83	74,22	81,80	81,87	69,58 [7,17]
Ionosphere	351, 34, 2	90,02	93,17	90,61	89,19	90,31	**93,72**	90,61 [4,82]
Letter-recognition	20000, 16, 26	78,85	91,78	90,42	89,86	74,02	3,87	**94,08** [0,44]
Lymph	148, 18, 4	76,33	85,19	82,48	82,43	**85,76**	79,01	81,10 [10,45]
Mol-bio-promoters	106, 58, 2	75,36	82,18	83,00	79,09	89,64	71,90	**89,73** [10,18]
Mol-bio-splice	3190, 61, 3	94,33	74,51	77,34	79,59	**95,20**	91,70	94,42 [1,30]
Pendigits	10992, 16, 10	88,23	**97,18**	96,84	96,58	87,90	83,91	95,12 [0,53]
Soybean-large	683, 35, 19	**93,27**	91,95	91,65	90,62	89,45	90,05	88,72 [2,00]
Spambase	4601, 57, 2	93,18	92,89	93,15	93,22	90,24	**93,34**	93,13 [0,92]
Statlog-segment	2310, 19, 7	95,15	94,16	93,68	92,86	91,65	93,90	**96,06** [1,09]
Statlog-Vehicle	846, 18, 4	68,90	**72,10**	71,38	71,02	61,34	69,01	71,50 [5,79]
Waveform-5000	5000, 40, 3	76,62	74,30	78,70	79,86	80,72	**81,34**	81,32 [1,72]
	Média	83,95	84,38	83,57	82,98	82,49	78,83	**85,87**

atributos (76,47%).

As bases de dados apresentadas nas Tabelas 5.4 (para todas as técnicas) e 5.5 (somente para o HiSP) sofreram redução do número de atributos por meio da técnica *Correlation-based Feature Selection*, enquanto para as bases das Tabelas 5.7 (para todas as técnicas) e 5.8 (somente para o HiSP) a redução foi obtida a partir da técnica *Consistency-based Feature Selection*. Portanto, a partir de uma mesma base de dados original, formaram-se diferentes bases de dados reduzidas em cada uma dessas avaliações. As diferenças estão relacionadas com os atributos que compõem cada base de dados reduzida e com a quantidade de atributos selecionados. Por exemplo, a seleção de atributos a partir da base de dados *Anneal*, utilizando-se a técnica *Correlation-based Feature Selection*, resultou no subconjunto $S_1 = \{family, hardness, strength, surfacequality, chrom, ferro, thick\}$, enquanto o subconjunto selecionado a partir dessa mesma base pela técnica *Consistency-based Feature Selection* foi $S_2 = \{family, steel, carbon, hardness, strength, surfacequality, enamelability, thick\}$. Apesar disso, a acurácia média obtida pelo HiSP a partir das bases de dados que sofreram redução do número de atributos por meio da técnica *Consistency-based Feature Selection* (85,87%) ficou muito próxima daquela obtida a partir das bases que sofreram redução do número de atributos por meio da técnica *Correlation-based Feature Selection* (85,93%). Ou seja, mesmo variando a técnica de seleção de atributos, o HiSP manteve o bom desempenho.

Assim como na Tabela 5.6, na qual se compara o HiSP com cada uma das outras técnicas de classificação, obtêm-se os resultados apresentados na Tabela 5.9. Novamente, o HiSP alcançou resultados de acurácia superiores aos das demais técnicas para a maioria das bases de dados do Grupo 2, cuja seleção de atributos foi obtida utilizando-se a técnica *Consistency-based Feature Selection*. A única exceção foi o empate ocorrido com o IBk ($k = 1$) – sem seleção de atributos –, onde o HiSP superou o IBk ($k = 1$) em nove bases de dados, mas perdeu em outras nove.

Tabela 5.9: Comparação do HiSP com as outras técnicas.

		Sem seleção de atributos						Com seleção de atributos					
		J48	IBk $k=1$	IBk $k=3$	IBk $k=5$	NB	CBA	J48	IBk $k=1$	IBk $k=3$	IBk $k=5$	NB	CBA
1	Res. Sup.	11	9	10	11	14	11	14	12	14	15	13	15
2	Res. Iguais	0	0	1	0	0	0	1	1	1	1	1	1
3	Res. Inf.	7	9	7	7	4	7	3	5	3	2	4	2

O desempenho, em termos de acurácia, do HiSP e das demais técnicas utilizadas nos experimentos para as bases de dados reduzidas a partir da técnica *Information Gain*

Attribute Ranking é apresentado nas Tabelas 5.10 e 5.11.

Assim como nos outros experimentos realizados com as bases de dados do Grupo 2 (Tabelas 5.4, 5.5, 5.7 e 5.8), os resultados apresentados nas Tabelas 5.10 e 5.11 demonstram a superioridade de desempenho do HiSP com relação às outras técnicas de classificação, independentemente de elas terem utilizado as bases de dados reduzidas ou originais. Em termos de acurácia média, pode-se observar, na última linha das Tabelas 5.10 e 5.11, que o HiSP alcançou 86,88%, enquanto os algoritmos que implementam as demais técnicas obtiveram (em ordem decrescente de desempenho): IBk (k=1) com seleção de atributos (84,39%), IBk (k=1) sem seleção de atributos (84,38%), IBk (k=3) com seleção de atributos (84,21%), J48 sem seleção de atributos (83,95%), IBk (k=3) sem seleção de atributos (83,57%), IBk (k=5) com seleção de atributos (83,47%), J48 com seleção de atributos (83,36%), IBk (k=5) sem seleção de atributos (82,98%), NB sem seleção de atributos (82,49%), NB com seleção de atributos (82,46%), CBA sem seleção de atributos (78,83%) e CBA com seleção de atributos (75,96%).

A superioridade do HiSP também pode ser observada quando comparado individualmente com cada uma das outras técnicas de classificação, contabilizando-se para quantas bases de dados ele apresentou acurácia superior, igual e inferior às demais técnicas. A Tabela 5.12 apresenta essa avaliação comparativa. Por exemplo, quando o HiSP é comparado com o IBk ($k = 3$) – sem seleção de atributos –, ele apresenta melhores acurácias para 14 bases de dados e piores para quatro bases de dados. Os demais resultados apresentados nessa tabela mostram que o HiSP sempre obtém desempenho superior ao das outras técnicas para a maioria das bases de dados do Grupo 2.

Observando-se o desempenho do HiSP para cada base de dados reduzida pelas técnicas de seleção de atributos utilizadas neste trabalho, percebe-se que, para a maioria dessas bases, quanto maior o número de atributos selecionados, maior foi a acurácia alcançada pelo HiSP.

As bases de dados reduzidas apresentadas nas Tabelas 5.10 e 5.11, continham, em média, 51% do número de atributos das bases de dados originais. Para essas bases reduzidas, o HiSP obteve 86,88% de acurácia média, um resultado superior àqueles apresentados nas Tabelas 5.4 e 5.5 (85,93%), e Tabelas 5.7 e 5.8 (85,87%), cujas bases de dados reduzidas tinham, em média, 30% (Tabelas 5.4 e 5.5) e 35% (Tabelas 5.7 e 5.8) do número de atributos das bases de dados originais.

A partir dos resultados apresentados nas Tabelas 5.3, 5.4, 5.5, 5.7, 5.8, 5.10 e 5.11, verifica-se também que, para bases de dados diferentes, mas reduzidas por uma mesma

Tabela 5.10: Resultados de acurácia para as bases de dados do Grupo 2 (todos os métodos utilizaram as bases reduzidas a partir da técnica *Information Gain Attribute Ranking*).

Bases de Dados	**Número de instâncias, número de atributos, número de classes**	**Com seleção de atributos**						
		J48	**IBk (k=1)**	**IBk (k=3)**	**IBk (k=5)**	**NB**	**CBA**	**HiSP**
Anneal	898, 38, 5	98,00	95,20	93,67	96,89	92,76	**98,68**	98,11 [1,58]
Audiology	226, 69, 24	**75,24**	74,28	69,78	65,43	68,56	72,14	72,55 [3,60]
Autos	205, 25, 6	78,10	84,10	79,70	74,45	74,43	79,05	**87,26** [5,96]
Chess-Kr-vs-Kp	3196, 36, 2	**97,09**	96,46	93,81	95,46	89,61	96,70	95,87 [0,64]
Flags	194, 29, 8	61,03	57,29	**66,40**	63,45	62,45	55,22	61,39 [7,10]
Hepatitis	155, 19, 2	77,24	82,50	81,58	84,42	83,83	85,08	**87,71** [6,37]
Horse-colic	368, 27, 2	**87,70**	79,91	82,67	81,57	83,96	79,67	82,91 [6,57]
Ionosphere	351, 34, 2	91,21	93,45	91,17	91,17	91,74	**94,56**	93,45 [4,47]
Letter-recognition	20000, 16, 26	79,40	90,89	88,48	87,21	73,33	3,54	**92,65** [0,61]
Lymph	148, 18, 4	75,57	85,86	83,86	82,43	**86,43**	79,67	85,81 [10,23]
Mol-bio-promoters	106, 58, 2	75,83	83,91	86,73	84,73	**91,45**	75,90	89,55 [8,39]
Mol-bio-splice	3190, 61, 3	91,89	83,35	85,33	85,67	**95,80**	94,56	95,11 [1,36]
Pendigits	10992, 16, 10	86,78	91,94	91,58	90,97	82,18	29,07	**93,55** [0,40]
Soybean-large	683, 35, 19	**87,86**	87,85	85,51	84,19	81,97	86,39	84,47 [2,93]
Spambase	4601, 57, 2	91,52	91,65	91,98	91,89	91,11	92,53	**92,94** [0,88]
Statlog-segment	2310, 19, 7	**95,32**	93,12	92,64	92,08	91,26	93,10	94,76 [1,33]
Statlog-Vehicle	846, 18, 4	**73,74**	71,07	71,04	69,50	62,05	69,96	72,09 [5,16]
Waveform-5000	5000, 40, 3	76,90	76,22	79,86	80,90	81,28	81,48	**83,62** [1,22]
	Média	83,36	84,39	84,21	83,47	82,46	75,96	**86,88**

Tabela 5.11: Resultados de acurácia para as bases de dados do Grupo 2 (somente o HiSP utilizou as bases reduzidas a partir da técnica *Information Gain Attribute Ranking*).

Bases de Dados	**Número de instâncias, número de atributos, número de classes**	**Sem seleção de atributos**						**Com seleção de atributos**
		J48	**IBk (k=1)**	**IBk (k=3)**	**IBk (k=5)**	**NB**	**CBA**	**HiSP**
Anneal	898, 38, 5	98,78	**99,22**	97,89	96,77	94,88	98,01	98,11 [1,58]
Audiology	226, 69, 24	**78,75**	71,25	60,95	58,70	66,80	70,75	72,55 [3,60]
Autos	205, 25, 6	78,48	85,33	78,93	76,95	72,02	78,58	**87,26** [5,96]
Chess-Kr-vs-Kp	3196, 36, 2	**99,41**	96,03	96,56	96,18	87,70	98,78	95,87 [0,64]
Flags	194, 29, 8	**62,53**	56,74	57,87	61,47	60,89	57,27	61,39 [7,10]
Hepatitis	155, 19, 2	76,00	83,08	86,96	84,96	84,54	81,87	**87,71** [6,37]
Horse-colic	368, 27, 2	**86,96**	77,70	75,83	74,22	81,80	81,87	82,91 [6,57]
Ionosphere	351, 34, 2	90,02	93,17	90,61	89,19	90,31	**93,72**	93,45 [4,47]
Letter-recognition	20000, 16, 26	78,85	91,78	90,42	89,86	74,02	3,87	**92,65** [0,61]
Lymph	148, 18, 4	76,33	85,19	82,48	82,43	85,76	79,01	**85,81** [10,23]
Mol-bio-promoters	106, 58, 2	75,36	82,18	83,00	79,09	**89,64**	71,90	89,55 [8,39]
Mol-bio-splice	3190, 61, 3	94,33	74,51	77,34	79,59	**95,20**	91,70	95,11 [1,36]
Pendigits	10992, 16, 10	88,23	**97,18**	96,84	96,58	87,90	83,91	93,55 [0,40]
Soybean-large	683, 35, 19	**93,27**	91,95	91,65	90,62	89,45	90,05	84,47 [2,93]
Spambase	4601, 57, 2	93,18	92,89	93,15	93,22	90,24	**93,34**	92,94 [0,88]
Statlog-segment	2310, 19, 7	**95,15**	94,16	93,68	92,86	91,65	93,90	94,76 [1,33]
Statlog-Vehicle	846, 18, 4	68,90	**72,10**	71,38	71,02	61,34	69,01	72,09 [5,16]
Waveform-5000	5000, 40, 3	76,62	74,30	78,70	79,86	80,72	81,34	**83,62** [1,22]
	Média	83,95	84,38	83,57	82,98	82,49	78,83	**86,88**

Tabela 5.12: Comparação do HiSP com as outras técnicas.

		Sem seleção de atributos						Com seleção de atributos					
		J48	IBk $k=1$	IBk $k=3$	IBk $k=5$	NB	CBA	J48	IBk $k=1$	IBk $k=3$	IBk $k=5$	NB	CBA
1	Res. Sup.	10	13	14	13	15	14	12	13	16	17	13	14
2	Res. Iguais	0	0	0	0	0	0	0	1	0	0	0	0
3	Res. Inf.	8	5	4	5	3	4	6	4	2	1	5	4

técnica de seleção de atributos, não existe uma correlação direta entre o grau de redução da base e o desempenho do HiSP. Ou seja, há bases de dados que sofreram grandes reduções e bases de dados que sofreram pequenas reduções, para as quais, comparando-se com os resultados das outras técnicas, o HiSP apresentou desempenho superior. Existem também bases de dados que sofreram grandes reduções e bases de dados que sofreram pequenas reduções, para as quais, comparando-se com os resultados das outras técnicas, o HiSP apresentou desempenho inferior.

Vale observar que em todos os experimentos realizados com as bases de dados do Grupo 2, cujos resultados são apresentados nas Tabelas 5.4, 5.5, 5.7, 5.8, 5.10 e 5.11, a técnica de classificação associativa (implementada pelo algoritmo CBA) sempre apresentou os piores resultados em termos de acurácia média. Apesar de não ser a única responsável, a base de dados que mais contribuiu para esse baixo desempenho foi a *Letter-recognition*, cujas acurácias ficaram sempre abaixo dos 4%. Mesmo que os resultados de acurácia dessa base de dados fossem desconsiderados nos cálculos da acurácia média do CBA apresentados nas Tabelas 5.4, 5.5, 5.7, 5.8, 5.10 e 5.11, o CBA continuaria tendo desempenho pior do que o do HiSP e de algumas outras técnicas utilizadas nesses experimentos.

5.3 Tempo Computacional e Escalabilidade

Na seção anterior, mostrou-se que o HiSP é um classificador que apresenta um bom desempenho em termos de acurácia, bastante competitivo quando comparado com outras técnicas de classificação. Nesta seção, será avaliado o desempenho do HiSP com relação ao tempo computacional (tempo de CPU) gasto para classificar uma instância. Por fim, um estudo será apresentado para comprovar a escalabilidade do método com relação ao número de instâncias das bases de dados.

Os resultados da análise de tempo para cada grupo de base de dados descrito na Seção 5.2 serão apresentados nas Tabelas 5.13 e 5.14. Para cada base de dados do Grupo 1, a Tabela 5.13 mostra o tempo médio de CPU gasto pelo HiSP para classificar uma

Tabela 5.13: Tempo de CPU do HiSP para as bases de dados do Grupo 1.

Bases de Dados	**Número de instâncias, número de atributos, número de classes**	**Tempo (s) [Desvio-padrão]**
Balance-scale	625, 4, 3	0.0067 [0.0047]
Breast-cancer	286, 9, 2	0.0222 [0.0107]
Breast-w	699, 9, 2	0.1917 [0.1584]
Credit-a	690, 15, 2	2.4346 [1.3387]
Diabetes	768, 8, 2	0.1017 [0.0236]
Glass	214, 9, 6	0.0742 [0.0353]
Hayes-roth	160, 4, 3	0.0049 [0.0050]
Heart-cleveland	303, 13, 2	0.6301 [0.2836]
Heart-hungarian	294, 13, 2	1.6955 [0.8678]
Iris	150, 4, 3	0.0013 [0.0033]
Labor	57, 16, 2	0.2604 [0.1668]
Liver-disorders	345, 6, 2	0.0481 [0.0055]
Postoperative	90, 8, 3	0.0076 [0.0048]
Primary-tumor	339, 17, 21	10.6012 [5.6733]
Shuttle-landing	15, 6, 2	0.0013 [0.0035]
Solar-flare1	323, 12, 6	0.6750 [0.2919]
Solar-flare2	1066, 12, 6	2.7835 [1.1352]
Statlog-heart	270, 13, 2	1.0989 [0.4194]
Tic-tac-toe	958, 9, 2	0.0411 [0.0066]
Vote	435, 16, 2	5.5862 [4.1793]
Wine	178, 13, 3	0.3901 [0.3904]
Zoo	101, 17, 7	3.2452 [2.0672]

instância e o seu desvio-padrão. Já a Tabela 5.14, apresenta essas mesmas informações para as bases de dados do Grupo 2, de acordo com cada técnica de seleção de atributos utilizada.

Normalmente, o HiSP, por ser uma técnica *lazy*, é mais lento do que técnicas de classificação *eager*, como por exemplo, árvores de decisão. No entanto, como pode ser observado nas Tabelas 5.13 e 5.14, o tempo gasto pelo HiSP na classificação de uma instância foi geralmente menor do que um segundo.

Os dados apresentados na Tabela 5.13 mostram que, para aproximadamente 70% das bases de dados do Grupo 1, o HiSP gastou no máximo um segundo para classificar uma instância, para 20% gastou entre um e três segundos, e para os 10% restantes, mais do que três segundos. Vale observar que as bases de dados para as quais o HiSP gastou mais de três segundos, são exatamente as três maiores bases do Grupo 1, todas contendo 16 ou

Tabela 5.14: Tempo de CPU do HiSP para as bases de dados do Grupo 2.

Bases de Dados	Correlation Tempo (s) [Desvio-padrão]	Consistency Tempo (s) [Desvio-padrão]	Information Gain Tempo (s) [Desvio-padrão]
Anneal	0.0800 [0.0271]	0.1088 [0.0384]	8.1128 [4.3246]
Audiology	0.0066 [0.0050]	0.4779 [0.2389]	1.8478 [1.2233]
Autos	0.0012 [0.0033]	0.0023 [0.0042]	0.5805 [0.3591]
Chess-Kr-vs-Kp	0.0234 [0.0083]	0.2669 [0.0726]	46.2952 [22.4591]
Flags	0.0023 [0.0042]	0.0052 [0.0051]	0.2010 [0.1195]
Hepatitis	0.0183 [0.0100]	0.0403 [0.0225]	1.1394 [1.0333]
Horse-colic	0.0019 [0.0039]	0.0004 [0.0020]	0.1659 [0.1739]
Ionosphere	0.5976 [0.4689]	0.0090 [0.0067]	0.9800 [0.8646]
Letter-recognition	0.2076 [0.1224]	1.1705 [1.1462]	0.2551 [0.1903]
Lymph	0.0299 [0.0107]	0.0111 [0.0055]	0.4260 [0.2091]
Mol-bio-promoters	0.0009 [0.0030]	0.0000 [0.0000]	0.0164 [0.0163]
Mol-bio-splice	0.0467 [0.0278]	0.1350 [0.0826]	0.7389 [0.5338]
Pendigits	0.2005 [0.0846]	0.1208 [0.0498]	0.1369 [0.0574]
Soybean-large	1.8470 [1.4337]	0.3294 [0.1661]	8.0881 [7.6929]
Spambase	1.7713 [1.1623]	22.6650 [31.8878]	17.8498 [16.1751]
Statlog-segment	0.0138 [0.0089]	0.0426 [0.0246]	1.2984 [1.8359]
Statlog-Vehicle	0.1160 [0.0549]	5.1056 [3.9047]	2.9954 [2.7743]
Waveform-5000	0.4710 [0.4163]	0.1733 [0.0609]	0.2426 [0.1920]

mais atributos (sem contar o atributo classe).

Apesar de as bases de dados do Grupo 2 serem originalmente maiores do que as do Grupo 1 em termos de número de instâncias, de atributos, ou de ambos, após sofrerem a redução a partir das técnicas de seleção de atributos citadas na Seção 5.2, o tempo de classificação gasto pelo HiSP tornou-se tão reduzido quanto aqueles apresentados para as bases do Grupo 1 (Tabela 5.13). Como pode ser observado na segunda coluna da Tabela 5.14, para todas as bases de dados cuja seleção de atributos foi realizada a partir da técnica *Correlation-based Feature Selection*, o HiSP gastou, na média, menos do que dois segundos para classificar uma instância. Para ser mais específico, para 16 dessas 18 bases de dados, esse tempo corresponde somente a uma pequena fração de um segundo.

Uma situação muito semelhante ocorreu com as bases de dados cuja redução do número de atributos foi obtida a partir do método *Consistency-based Feature Selection*. Os dados apresentados na terceira coluna da Tabela 5.14 mostram que, para 15 dentre as 18 bases de dados, o HiSP não chega a gastar nem um segundo para classificar uma instância. Somente para a base de dados *Spambase*, que mesmo após a redução continuou tendo um número elevado de atributos, o tempo médio de classificação de uma instância foi superior ao das demais bases de dados, alcançando 22,665 segundos.

Por fim, os resultados apresentados na quarta coluna da Tabela 5.14, referentes às bases de dados que sofreram redução do número de atributos a partir da técnica *Information Gain Attribute Ranking*, continuam mostrando que o HiSP, para a maioria das

bases, gastou no máximo um segundo para classificar uma instância. Pelo fato de a redução dessas bases de dados ter sido geralmente menor do que aquelas obtidas pelas técnicas *Correlation-based Feature Selection* e *Consistency-based Feature Selection*, para duas bases de dados, *Chess-Kr-vs-Kp* e *Spambase*, o tempo médio de classificação de uma instância ficou acima dos 10 segundos.

Para verificar a escalabilidade do HiSP com realação ao número de instâncias das bases de dados, foram utilizadas três das 40 bases de dados adotadas nos experimentos realizados nesta tese. Foram selecionadas as duas maiores bases (em termos de número de instâncias) do Grupo 1, a *Solar-flare2* e a *Tic-tac-toe*, e uma base de dados pertencente ao Grupo 2, a *Waveform-5000*.

Nessa análise, para cada base de dados, selecionou-se aleatoriamente 90% das instâncias da base original para treinamento e os 10% restantes para teste. Em seguida, quatro bases de dados de treinamento foram formadas, contendo, respectivamente, 20%, 40%, 60% e 80% das instâncias de treinamento. Todas essas bases de dados (treinamento e teste) continham o mesmo número de atributos.

A Figura 5.1 mostra a escalabilidade linear do tempo médio de CPU gasto pelo HiSP para classificar uma instância das bases de dados *Solar-flare2*, *Tic-tac-toe* e *Waveform-5000*. Vale ressaltar que, nesses experimentos, utilizou-se a base de dados *Waveform-5000* reduzida a partir da técnica *Correlation-based Feature Selection*.

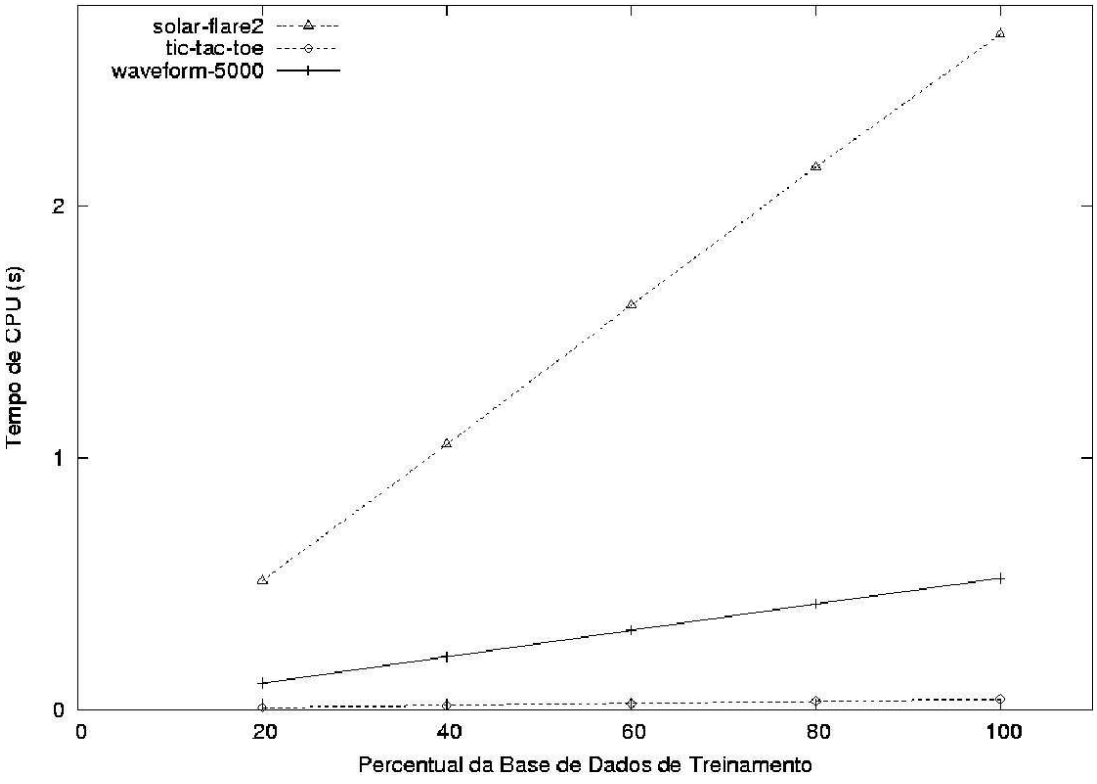


Figura 5.1: Escalabilidade segundo o número de instâncias.

Capítulo 6

Conclusões

Nas últimas décadas, os avanços tecnológicos na área da computação têm contribuído para um enorme crescimento da quantidade de dados armazenados em computadores espalhados pelo mundo. Desse modo, a necessidade por ferramentas capazes de realizar análises nesses dados torna-se evidente, fazendo com que pesquisas voltadas para a criação de métodos computacionais eficientes sejam cada vez mais importantes.

Devido à grande quantidade de dados biológicos gerados pela comunidade científica, a área da biologia molecular tem utilizado cada vez mais técnicas computacionais para auxiliar no processo de análise desses dados. Particularmente, várias técnicas de mineração de dados vêm sendo utilizadas por pesquisadores para solucionar problemas biológicos, tal como o Problema de Classificação de Proteínas (PCP).

A classificação de proteínas é de extrema importância para a biologia molecular, uma vez que, identificadas as classes das proteínas, é possível inferir quais são as suas funções. Apesar dos avanços e da intensa pesquisa nessa área, o PCP continua sendo um desafio, já que sua solução envolve um grande número de variáveis.

Na busca por soluções para o PCP, diferentes abordagens foram propostas em trabalhos da área de bioinformática. Nesta tese, a abordagem utilizada realiza a classificação de proteínas em famílias funcionais a partir de suas composições de motivos, supondo que proteínas de uma mesma família possuam funções semelhantes.

Com o objetivo de apresentar um método de classificação de proteínas computacionalmente eficiente e capaz de alcançar altas taxas de acurácia, superando resultados apresentados em trabalhos anteriores, propôs-se a criação do método HiSP-Prot (***Hi****ghest* ***S****ubset* ***P****robability - for* ***Prot****ein Classification Problem*), cuja idéia central é classificar uma proteína a partir da avaliação dos seus subconjuntos de motivos. Esse classificador

atribui à proteína a classe que melhor é descrita por um dos subconjuntos de seus motivos.

Vários experimentos foram conduzidos para se avaliar o desempenho do método proposto. A primeira avaliação foi realizada comparando-se os resultados obtidos por meio da abordagem proposta com aqueles apresentados por outros dois classificadores: método baseado em autômatos finitos [70] e árvore de decisão [38]. O desempenho do HiSP-Prot, em termos de acurácia preditiva, foi superior aos dos métodos utilizados em [70] e [38] em todas as bases de dados testadas. Além disso, testes realizados em uma base de dados (*genprosite*) contendo um elevado número de proteínas e classes, mostraram que o HiSP-Prot manteve o ótimo desempenho apresentado para as outras bases de dados.

Nessa primeira avaliação comparativa, somente a acurácia preditiva foi utilizada como medida de desempenho dos classificadores. No entanto, para bases de dados com distribuições de classes desbalanceadas, utilizar somente a acurácia preditiva como medida de desempenho pode ser insuficiente para se obter conclusões precisas. Portanto, uma segunda avaliação foi realizada com o objetivo de averiguar qual era o impacto do desbalanceamento de classes das bases no desempenho do HiSP-Prot. Os resultados dessa avaliação mostraram que, apesar de existir um considerável desbalanceamento entre as classes das bases de dados testadas, o HiSP-Prot obteve bons desempenhos tanto para classes majoritárias quanto para as minoritárias. Acredita-se que a estratégia de classificação adotada pelo HiSP-Prot, isto é, a procura por subconjuntos de motivos que tenham alta probabilidade de pertencer a alguma classe, contribua para o seu bom desempenho mesmo para classes pouco representadas nas bases de dados, dado que a medida de probabilidade pode apresentar valores elevados tanto para classes minoritárias quanto para majoritárias.

Os experimentos mostraram também a viabilidade do método em relação ao tempo de CPU consumido na classificação das proteínas. No entanto, esse bom resultado é devido à quantidade de motivos ser relativamente pequena para as proteínas extraídas do banco de dados PROSITE (no máximo 10 motivos por proteína). Se o cenário fosse diferente, ou seja, se as proteínas fossem caracterizadas por um número maior de motivos, o HiSP-Prot apresentaria custos computacionais mais elevados na classificação das mesmas, podendo, em alguns casos, inviabilizar sua aplicação. Por esse motivo, apresentou-se também nesta tese uma alteração na versão inicial do HiSP-Prot para viabilizar a sua utilização mesmo para proteínas caracterizadas por uma quantidade maior de motivos. A alteração proposta foi a inclusão de um pré-processador responsável por selecionar, segundo um critério de qualidade, somente os k melhores motivos da proteína a ser classificada antes de se

iniciar o procedimento de classificação proposto no HiSP-Prot. Experimentos realizados adotando-se $k = 5$ mostraram que a seleção de motivos não prejudicou o desempenho do classificador, apresentando, ao invés disso, uma acurácia preditiva ligeiramente superior ao do HiSP-Prot sem o pré-processamento.

Portanto, pode-se concluir que o método proposto nesta tese evidenciou um apropriado e eficiente comportamento para o problema de classificar proteínas a partir de seus motivos, apresentando desempenho superior ao de outros métodos já propostos.

Os bons resultados obtidos com o HiSP-Prot serviram como motivação para a criação de um método que pudesse ser utilizado com outros tipos de bases de dados e, portanto, com outras aplicações. Desse modo, o segundo método proposto neste trabalho, denominado HiSP (***Hi****ghest* ***S****ubset* ***P****robability*), corresponde a uma generalização do HiSP-Prot.

Na estratégia adotada pelo HiSP, uma instância é classificada a partir da avaliação dos seus subconjuntos de valores de atributos. Os subconjuntos com as maiores probabilidades de pertencerem a determinadas classes definem a classificação da nova instância.

No entanto, dependendo das dimensões da base de dados e das características da instância a ser classificada, o custo computacional do processamento realizado pelo HiSP pode ser elevado. Desse modo, para tornar viável a utilização do método com bases de dados de qualquer dimensão, em alguns casos, pode ser necessária uma etapa de pré-processamento da base visando a redução do seu número de atributos.

A avaliação do HiSP foi realizada a partir de quarenta bases de dados de domínio público freqüentemente utilizadas em experimentos de mineração de dados, obtidas no repositório de dados *UCI Machine Learning Repository*. Primeiramente, a acurácia preditiva foi o critério utilizado numa avaliação comparativa entre o método proposto e outras importantes técnicas de classificação, tais como: árvore de decisão, k-NN, classificação Bayesiana simples e classificação associativa. Em seguida, avaliou-se também o desempenho do HiSP com relação ao tempo de CPU gasto para classificar uma instância. Por fim, um estudo foi realizado com o objetivo de verificar se o método é escalável com relação ao número de instâncias das bases de dados.

Nesses experimentos, grande parte das bases de dados contendo mais de 15 atributos sofreram uma redução do seu número de atributos antes de serem processadas pelo HiSP. Para tanto, três técnicas de seleção de atributos foram utilizadas: *Correlation-based Feature Selection*, *Consistency-based Feature Selection* e *Information Gain Attribute Ranking*.

Os resultados experimentais mostraram que o HiSP, em termos de acurácia preditiva,

foi sempre competitivo, e freqüentemente superior às demais técnicas avaliadas. Isso foi observado tanto nos experimentos realizados com as bases de dados originais quanto naqueles realizados com as bases de dados reduzidas. No entanto, a possibilidade de a redução do número de atributos implicar em perda de informação útil para a classificação, pode caracterizar uma limitação do método proposto neste trabalho.

Além disso, mesmo sendo caracterizado como uma estratégia de classificação *lazy*, o HiSP apresentou um ótimo desempenho com relação ao tempo médio de CPU gasto na classificação de uma instância. Para a grande maioria das bases de dados utilizadas nos experimentos, esse tempo médio foi inferior a um segundo.

Os testes de escalabilidade foram realizados a partir de três bases de dados escolhidas entre aquelas que foram utilizadas nos experimentos. Os resultados mostraram que o HiSP é escalável com relação ao crescimento do número de instâncias das bases de dados.

Portanto, os resultados obtidos nos experimentos realizados neste trabalho demonstraram que o HiSP é um método que pode ser utilizado de forma adequada e eficiente para diferentes tipos de bases de dados pertencentes a aplicações distintas.

Trabalhos Futuros

A utilização de técnicas de seleção de atributos numa etapa de pré-processamento, visando a redução do número de atributos das bases de dados, torna o método HiSP aplicável a bases de dados de dimensões maiores. No entanto, para algumas bases de dados, a redução do número de atributos pode implicar em perda de informação útil para a classificação, resultando na degradação do desempenho do classificador a partir da base de dados reduzida.

Desse modo, como trabalho futuro, em vez de se utilizar técnicas de seleção de atributos para reduzir o número de atributos das bases de dados, pretende-se construir heurísticas com o objetivo de reduzir o custo computacional do processamento dos subconjuntos de valores de atributos das instâncias a serem classificadas.

Com o objetivo de melhorar a acurácia preditiva de técnicas de classificação, dois métodos que realizam a combinação de classificadores foram propostos: *Bagging* [16] e *Boosting* [32]. O método *Bagging* consiste em utilizar múltiplos subconjuntos de instâncias, selecionadas aleatoriamente na base de dados de treinamento, para criar diferentes classificadores. Nesse caso, a decisão da classificação final é obtida a partir do voto de cada classificador. Já no método *Boosting*, os diferentes classificadores são obtidos a par-

tir de um processo iterativo. Nesse processo, em cada iteração, um novo classificador é construído utilizando pesos atribuídos às instâncias de treinamento na iteração anterior. Esses pesos são utilizados com o objetivo de corrigir a classificação de instâncias erroneamente classificadas nas iterações anteriores. Nesse método, a classificação final é dada combinando-se os votos de cada classificador, sendo que o peso associado ao voto de cada um deles está diretamente relacionado com a sua acurácia preditiva. Como trabalho futuro, pretende-se também avaliar o desempenho do HiSP utilizando-se os métodos *Bagging* e *Boosting*. Além disso, pode-se também avaliar a combinação do HiSP com outros algoritmos de classificação por meio da técnica *Selective Fusion*.

Referências

[1] Agrawal, R., Imielinski, T., Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (Washington, D.C., May 1993), p. 207–216.

[2] Agrawal, R., Srikant, R. Fast algorithms for mining association rules. In *Proceedings of* 20^{th} *International Conference on Very Large Data Bases, VLDB* (1994).

[3] Aha, D. W. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies 36*, 2 (1992), 267–287.

[4] Altschul, S. F., Madden, T. L., Schffer, A. A., Zhang, J., Zhang, Z., Miller, W., Lipman, D. J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research 25*, 17 (1997), 3389–3402.

[5] Antonie, M. L., Zaïane, O. R. An associative classifier based on positive and negative rules. In *Proceedings of* 9^{th} *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (Paris, France, June 2004), p. 64–69.

[6] Attwood, T. K., Croning, M. D. R., Flower, D. R., Lewis, A. P., Mabey, J. E., Scordis, P., Selley, J., Wright, W. Prints-S: The database formerly known as prints. *Nucleic Acids Research 28* (2000), 225–227.

[7] Avner, S. Discovery of comprehensible symbolic rules in a neural network. In *Proceedings of International Symposium on Intelligence in Neural and Biological Systems* (1995), p. 64–67.

[8] Bailey, T. L., Baker, M. E., Elkan, C., Grundy, W. N. MEME, MAST, and Meta-MEME: New tools for motif discovery in protein sequences. In *Pattern Discovery in Biomolecular Data*. 1999, p. 30–54.

[9] Barker, W. C., Garavelli, J. S., Huang, H. The protein information resource (PIR). *Nucleic Acids Research*, 28 (2000), 263–266.

[10] Baxevanis, A. D. The molecular biology database collection: an online compilation of relevant database resources. *Nucleic Acids Research 28*, 1 (2000), 1–7.

[11] Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., Bourne, P. The protein data bank. *Nucleic Acids Research 28* (2000), 235–242.

[12] Blake, C., Newman, D., Hettich, S., Merz, C. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998. University of California, Irvine, Dept. of Information and Computer Sciences.

[13] Blekas, K., Fotiadis, D. I., Likas, A. Motif-based protein sequence classification using neural networks. *Journal of Computational Biology 12*, 1 (2005), 64–82.

[14] Blum, A. L., Langley, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence 97* (1997), 245–271.

[15] Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.-C., Estreicher, A., Gasteiger, E., Martin, M. J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S., Schneider, M. The SWISS-PROT protein knowledgebase and its supplement trEMBL in 2003. *Nucleic Acids Research 31*, 1 (2003), 365–370.

[16] Breiman, L. Bagging predictors. *Machine Learning 24* (1996), 123–140.

[17] Burges, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery 2*, 2 (1998), 121–167.

[18] Catlett, J. On changing continuous attributes into ordered discrete attributes. In *Proceedings of the European Working Session on Learning* (Berlin, Germany, 1991), Y. Kodratoff, Ed., Spring-Verlag, p. 164–178.

[19] Cooper, G., Herskovits, E. A bayesian method for the induction of probabilistic networks from data. *Machine Learning 9*, 4 (1992), 309–347.

[20] Cover, T. M., Hart, P. E. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*, 1 (1967), 21–27.

[21] Dayhoff, M. O., Barker, W. C., Hunt, L. T. Establishing homologies in protein sequences. *Methods in Enzymology 91* (1983), 524–545.

[22] Diplaris, S., Tsoumakas, G., Mitkas, P. A., Vlahavas, I. Protein classification with multiple algorithms. In *Proceedings of* 10*th Panhellenic Conference on Informatics* (Volos, Greece, November 2005), P. Bozanis and E. Houstis, Eds., LNCS 3746, Springer-Verlag, p. 448–456.

[23] Dong, G., Li, J. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the* 5^{th} *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Diego, CA, August 1999), ACM Press, p. 43–52.

[24] Dong, G., Zhang, X., Wong, L., Li, J. CAEP: Classification by aggregating emerging patterns. *Discovery Science* (1999), 30–42.

[25] Dougherty, J., Kohavi, R., Sahami, M. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference* (San Francisco, CA, 1995), Morgan Kaufmann Publishers, p. 194–202.

[26] Duda, R., Hart, P. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[27] Dumais, S., Platt, J., Heckerman, D., Sahami, M. Inductive learning algorithms and representations for text categorization. In *Proceedings of the International Conference on Information and Knowledge Management* (1998), p. 148–155.

[28] Efron, B., Tibshirani, R. *An Introduction to the Bootstrap*. Chapman and Hall, New York, 1993.

[29] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. From data mining to knowledge discovery in databases. *AI Magazine 17*, 3 (1996), 37–54.

[30] Fayyad, U. M., Irani, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (Chambéry, France, 1993), Morgan Kaufmann, p. 1022–1029.

[31] Fix, E., Júnior, J. L. H. Discriminatory analysis, non-parametric discrimination: Consistency properties. Tech Report 21-49-004(4), USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[32] Freund, Y., Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55* (1997), 119–139.

[33] Gehrke, J., Ramakrishnan, R., Ganti, V. Rainforest: A framework for fast decision tree construction of large datasets. In *Proceedings of* 24^{th} *International Conference on Very Large Data Bases, VLDB* (New York, August 1998), p. 416–427.

[34] Hall, M. A. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, Departament of Computer Science, University of Waikato, Hamilton, New Zealand, 1998.

[35] Hall, M. A. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the* 17^{th} *International Conference on Machine Learning* (2000).

[36] Hall, M. A., Holmes, G. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering 15*, 6 (November-December 2003), 1437–1447.

[37] Han, J., Kamber, M. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, New York, 2001.

[38] Hatzidamianos, G., Diplaris, S., Athanasiadis, I., Mitkas, P. A. Genminer: A data mining tool for protein analysis. In *Proceedings of the* 9^{th} *Panhellenic Conference On Informatics* (Thessaloniki, Greece, 2003), p. 346–360.

[39] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Macmillan Publishing Company, New York, 1994.

[40] Heckerman, D., Geiger, D., Chickering, D. M. Learning bayesian networks: The combination of knowledge and statistical data. Tech Report MSR-TR-94-09, Microsoft Research, Redmond, Washington, March 1994.

[41] Henikoff, J. G., Greene, E. A., Pietrokovski, S., Henikoff, S. Increased coverage of protein families with the blocks database severs. *Nucleic Acids Research 28* (2000), 228–230.

[42] Henikoff, S., Henikoff, J. G. Protein family databases. In *Encyclopedia of life sciences*. Macmillan Publishers Ltd, Nature Publishing Group, 2001. http://www.els.net.

[43] Holte, R. C. Very simple classification rules perform well on most commonly used datasets. *Machine Learning 11* (1993), 63–90.

[44] Horn, F., Weare, J., Beukers, M. W., Hörsch, S., Bairoch, A., Chen, W., Edvardsen, O., Campagne, F., Vriend, G. GPCRDB: An information system for g protein coupled receptors. *Nucleic Acids Research 21*, 1 (1998), 227–281.

[45] Hulo, N., Bairoch, A., Bulliard, V., Cerutti, L., Castro, E. D., Langendijk-Genevaux, P., Pagni, M., Sigrist, C. The PROSITE database. *Nucleic Acids Research 34* (2006), 227–230.

[46] Japkowicz, N. Class imbalances: Are we focusing on the right issue? In *Proc. of Workshop on Learning from Imbalanced Data Sets II (ICML)* (2003).

[47] Johns, M. V. *Studies in Item Analysis and Prediction*. Stanford University Press, Palo Alto, CA, 1961.

[48] Júnior, E. R. H. *Imputação Bayesiana no Contexto da Mineração de Dados*. PhD thesis, Universidade Federal do Rio de Janeiro, Outubro 2003.

[49] Kerber, R. ChiMerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (1992), MIT Press, p. 123–128.

[50] Kira, K., Rendell, L. A practical approach to feature selection. In *Proceedings of the Ninth Internation Conference on Machine Learning* (1992), Morgan Kaufmann, p. 249–256.

[51] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of* 14^{th} *International Joint Conference on Artificial Intelligence* (Montreal, Canada, August 1995), vol. 2, p. 1137–1143.

[52] Kohavi, R., John, G. H. Wrappers for feature subset selection. *Artificial Intelligence 97* (1997), 273–34.

[53] Kononenko, I. Estimating attributes: Analysis and extensions of relief. In *Proceedings of the Seventh European Conference on Machine Learning* (1994), Springer-Verlag, p. 171–182.

[54] Lauritzen, S. L. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis 19* (1995), 191–201.

[55] Lent, B., Swami, A., Widom, J. Clustering association rules. In *Proceedings of the IEEE International Conference on Data Engineering* (Birmingham, England, April 1997), p. 220–231.

[56] Li, J., Dong, G., Ramamohanarao, K. JEP-classifier: Classification by aggregating jumping patterns. Tech report, University of Melbourne, 1999.

[57] Li, J., Dong, G., Ramamohanarao, K., Wong, L. DeEPs: A new instance-based discovery and classification system. *Machine Learning 54* (2004), 99–124.

[58] Liu, B., Hsu, W., Ma, Y. Integrating classification and association rule mining. In *Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining* (New York, August 1998), p. 80–86.

[59] Liu, H., Setiono, R. A probabilistic approach to feature selection: A filter solution. In *Proceedings of the* 13^{th} *International Conference on Machine Learning* (1996), M. Kaufmann, Ed., p. 319–327.

[60] Mehta, M., Agrawal, R., Rissanen, J. Sliq: A fast scalable classifier for data mining. In *Proceedings of International Conference on Extending Database Technology* (Avignon, France, March 1996), p. 18–32.

[61] Merschmann, L., Plastino, A. A bayesian approach for protein classification. In *Proceedings of the* 21^{st} *Annual ACM Symposium on Applied Computing* (Dijon, France, April 23-27 2006).

[62] Merschmann, L., Plastino, A. HiSP: A probabilistic data mining technique for protein classification. In *Proceedings of the 2006 International Workshop on Bioinformatics Research and Applications* (Reading, U.K., May 28-31 2006), LNCS 3992, p. 863–870.

[63] Merschmann, L., Plastino, A. A lazy data mining approach for protein classification. *IEEE Transactions on Nanobioscience 6*, 1 (March 2007), 36–42.

[64] Mitra, S., Acharya, T. *Data Mining: Multimedia, Soft Computing and Bioinformatics*. Jonh Wiley & Sons, 2003.

[65] Murzin, A. G., Brenner, S. E., Hubbart, T., Chothia, C. SCOP: A structural classification proteins database for investigations of sequences and structures. *Journal of Molecular Biology 247* (1995), 536–540.

[66] Pearl, F. M. G., Bennett, C. F., Bray, J. E., Harrison, A. P., Martin, N., Shepherd, A., Sillitoe, I., Thornton, J., Orengo, C. A. The cath database: An extended protein family resource for structural and functional genomics. *Nucleic Acids Research 31*, 1 (2003), 452–455.

[67] Pearson, W. R. Flexible sequence similarity searching with the FASTA3 program package. *Methods in Molecular Biology 132* (2000), 185–219.

[68] Peng, H., Ding, C., Long, F. Minimum redundancy-maximum relevance feature selection. *IEEE Intelligent Systems* (November-December 2005), 70–71.

[69] Prati, R. C., Batista, G. E. A. P. A., Monard, M. C. Class imbalances versus class overlapping: An analysis of a learning system behaviour. In *Proc. of the III Mexican International Conference on Artificial Intelligence – (MICAI)* (2004), LNAI 2972, Springer, p. 312–321.

[70] Psomopoulos, F., Diplaris, S., Mitkas, P. A. A finite state automata based technique for protein classification rules induction. In *Proceedings of the* 2^{nd} *European Workshop on Data Mining and Text Mining in Bioinformatics* (Pisa, Italy, 2004), p. 54–60.

[71] Quinlan, J. R. Induction of decision trees. *Machine Learning 1*, 1 (1986), 81–106.

[72] Quinlan, J. R. Unknown attribute values in induction. In *Proceedings of the Sixth International Workshop on Machine Learning* (Ithaca, NY, 1989), M. Kaufmann, Ed., p. 164–168.

[73] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.

[74] Ramamohanarao, K., Bailey, J., Fan, H. Efficient mining of contrast patterns and their application to classification. In *Proceedings of the* 3^{rd} *International Conference on Intelligent Sensing and Information Processing* (Bangalore, India, December 2005).

[75] Rawlings, N. D., Barret, A. J. MEROPS: The peptidase database. *Nucleic Acids Research 28* (2002), 323–325.

[76] Rezende, S. O. *Sistemas Inteligentes: Fundamentos e Aplicações*. Manole, Barueri, SP, 2005.

[77] Richeldi, M., Rossotto, M. Class-driven statistical discretization of continuous attributes. In *Proceedings of European Conference on Machine Learning* (New York, 1995), LNAI 914, Springer Verlag, p. 335–338.

[78] Rissanen, J. Stochastic complexity and modeling. *Annals of Statistic 14*, 3 (1986), 1080–1100.

[79] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review 65*, 6 (1958), 386–408.

[80] Rumelhart, D. E., Hinton, G. E., Williams, R. J. *Learning Internal Representations by Error Propagation*, vol. 1 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, MA, 1986.

[81] Russell, S., Binder, J., Koller, D., Kanazawa, K. Local learning in probabilistic networks with hidden variables. In *Proceedings of* 14^{th} *International Joint Conference on Artificial Intelligence* (Monteral, Canada, August 1995), vol. 2, Morgan Kaufmann, p. 1146–1152.

[82] Saito, K., Nakano, R. Medical diagnostic expert system based on pdp model. In *Proceedings of IEEE International Conference on Neural Networks* (San Mateo, CA, 1988), vol. 1, p. 255–262.

[83] Seidman, C. *Data Mining with Microsoft SQL Server*. Microsoft Press, Redmond, WA, 2000.

[84] Shafer, J., Agrawal, R., Mehta, M. Sprint: A scalable parallel classifier for data mining. In *Proceedings of* 22^{th} *International Conference on Very Large Data Bases, VLDB* (Bombay, India, September 1996), p. 544–555.

[85] Stone, M. Cross-validation choice and assessment of statistical predictions. *Journal of the Royal Statistical Society 36* (1974), 111–147.

[86] Towell, G. G., Shavlik, J. W. Extracting refined rules from knowledge-based neural networks. *Machine Learning 13* (October 1993), 71–101.

[87] Tsunoda, D. F. *Abordagens Evolucionárias para a Descoberta de Padrões e Classificação de Proteínas*. PhD thesis, Centro Federal de Educação Tecnológica do Paraná, Dezembro 2004.

[88] Tsunoda, D. F., Lopes, H. S., Freitas, A. A. An evolutionary approach for motif discovery and transmembrane protein classification. In *Proceedings of EvoBIO-2005: 3rd European Workshop on Evolutionary Bioinformatics*, F. R. et al., Ed., LNCS 3449, Springer, p. 105–114.

[89] Vapnik, V. N. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[90] Veloso, A., Jr, W. M. Eager, lazy and hybrid algorithms for multi-criteria associative classification. In *Proceedings do* 1° *Workshop sobre Algoritmos de Mineração de Dados* (Uberlândia, MG, Outubro 2005), p. 17–24.

[91] Wang, D., Wang, X., Honavar, V., Dobbs, D. L. Data-driven generation of decision trees for motif-based assignment of protein sequences to functional families. In *Proceedings of the Atlantic Symposium on Computational Biology, Genome Information Systems & Technology* (North Carolina, USA, 2001).

[92] Wang, J. T. L., Zaki, M. J., Toivonen, H. T. T., Shasha, D., Eds. *Data Mining in Bioinformatics*. Springer, 2005.

[93] Wang, X., Schroeder, D., Dobbs, D., Honavar, V. Automated data-driven discovery of motif-based protein function classifiers. *Information Sciences 155* (2003), 1–18.

[94] White, A. P., Liu, W. Z. Technical note: Bias in information-based measures in decision tree induction. *Machine Learning 15*, 3 (June 1994), 321–329.

[95] Witten, I. H., Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[96] Yang, Y., Pedersen, J. O. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning* (1997), p. 412–420.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo