MARCUS VINICIUS SILVA SOARES

Orientador: Luiz Henrique de Campos Merschmann

AVALIAÇÃO DE UMA ABORDAGEM *LAZY* DE SELEÇÃO DE ATRIBUTOS BASEADA NA MEDIDA DE CONSISTÊNCIA

Ouro Preto

Dezembro de 2010

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

AVALIAÇÃO DE UMA ABORDAGEM *LAZY* DE SELEÇÃO DE ATRIBUTOS BASEADA NA MEDIDA DE CONSISTÊNCIA

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

MARCUS VINICIUS SILVA SOARES

Ouro Preto
Dezembro de 2010

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Avaliação de uma Abordagem *Lazy* de Seleção de Atributos Baseada na Medida de Consistência

MARCUS VINICIUS SILVA SOARES

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Luiz Henrique de Campos Merschmann – Orientador
Universidade Federal de Ouro Preto

Dr. Álvaro Rodrigues Pereira Júnior
Universidade Federal de Ouro Preto

Dr. David Menotti Gomes
Universidade Federal de Ouro Preto

Ouro Preto, Dezembro de 2010

Resumo

O processo de descoberta de informação (*Knowledge Discovery in Databases - KDD*) é composto das seguintes fases: limpeza dos dados, integração dos dados, redução de dados, transformação de dados, mineração de dados, pós-processamento e visualização dos resultados.

O desenvolvimento de novas técnicas de redução de dados que aumentem a eficácia do processo de descoberta de informação pode trazer contribuições importantes para a área de mineração de dados. A seleção de atributos é uma etapa de redução dos dados que visa identificar atributos relevantes para a tarefa de classificação. O seu objetivo é remover atributos da base de dados que não contribuam para a classificação ou que possam prejudicar a capacidade preditiva do classificador.

Tradicionalmente, técnicas de seleção de atributos são executadas na fase de pré-processamento dos dados e suas decisões são definitivas para a fase de classificação. Essas técnicas são classificadas como seleção de atributos *eager*. Diferentemente dessa estratégia *eager*, neste trabalho, propõe-se a avaliação de uma abordagem de seleção de atributos *lazy*, onde a escolha dos atributos a serem utilizados é adiada ao momento em que uma instância é submetida à classificação. Mais especificamente, a técnica *Consistency-based Feature Selection* foi adaptada para funcionar segundo a abordagem *lazy*. Experimentos foram realizados utilizando-se bases de dados obtidas no repositório da *UCI Machine Learning Repository* e o classificador k-NN foi adotado nessa avaliação. Em termos de acurácia preditiva, os resultados obtidos indicam que a abordagem *lazy* proposta pode ser vantajosa para algumas bases de dados.

Palavras-chave: Redução de dados. Seleção de atributos. Classificação.

Abstract

The Knowledge Discovery in Databases (KDD) consists of the following phases: data cleansing, data integration, data reduction, data transformation, data mining, post-processing and visualization of results.

The development of new data reduction techniques that increase the efficiency of the discovery of information can make significant contributions to the area of data mining. The feature selection is one step of data reduction that aims to identify relevant attributes for the classification task. Its goal is to remove attributes from the database that do not contribute to the classification or which may affect the predictive ability of the classifier.

Traditionally, feature selection techniques are performed in the pre-processing of data and its decisions are final for the classification phase. These techniques are classified as feature selection eager. Differently of this strategy eager, in this work, we propose an approach to evaluation of feature selection lazy, where the choice of attributes to be used is deferred to when a instance is subjected to the classification . More specifically, the technique *Consistency-based Feature Selection* was adapted to function according to the approach lazy. Experiments were conducted using databases collected in the repository of *UCI Machine Learning Repository* and the sorter k-NN was used in this assessment. In terms of predictive accuracy, the results indicate that the approach lazy proposal may be advantageous to some databases.

Keywords: Data reduction. Feature selection. Classification.

Dedico este trabalho ao Deus todo poderoso, por toda sua Graça e Amor. Ao meu pai por todo amor, apoio, por ter acreditado em mim e por ser uma benção de Deus em minha vida.

Agradecimentos

Agradeço ao meu Deus pela vida, pelas bênçãos derramadas e pela oportunidade de ter concretizado mais um sonho. Toda Honra e Glória seja dada ao Senhor.

Ao meu pai e a Isabel, por toda preocupação e dedicação para comigo, por muitas vezes abrirem mão das suas próprias vontades para que os meus objetivos fossem alcançados.

Ao meu irmão, por simplesmente ser um grande irmão em minha vida.

Aos meus familiares, pelo inestimável apoio, incentivo e carinho mesmo às vezes estando tão longe.

A Jaqueline, pelo amor, carinho e paciência nos momentos felizes e nos momentos tristes.

A minha segunda família (REPÚBLICA DUSMININU), por simplesmente serem bênçãos de Deus em minha vida.

Aos meus queridos amigos da UFOP, em especial Degeo, Rodrigão, Rodolfo, Pedro, Larissa, Cecília, Tales e aos demais que participaram comigo dessa caminhada.

Aos demais amigos espalhados por este mundo, que durante essa jornada me acompanharam e sempre torceram pelo meu sucesso.

Ao professor Luiz Merschmann, orientador deste trabalho, por ter aceitado me orientar, pela sua dedicação, preocupação, paciência e apoio na produção deste trabalho. Sem dúvidas foi uma ótima orientação.

Aos professores e funcionários da UFOP, pelo ensinamento e por muito contribuírem para a minha formação acadêmica.

E a todos que, direta ou indiretamente, contribuíram para a minha formação.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Atualmente empresas e organizações estão cada vez mais coletando e armazenando grandes quantidades de dados devido à queda dos preços de meios de armazenamento e computadores. A popularização na utilização de armazém de dados, ou *data warehouses*, que são grandes bancos de dados criados para análise e suporte à decisão, tende a aumentar ainda mais a quantidade de informações disponíveis. Os métodos tradicionais de análise de dados, como planilhas e consultas SQL não são apropriados para tais volumes de dados, pois podem criar relatórios informativos sobre os dados, mas não conseguem analisar o conteúdo destes relatórios a fim de obter diversos tipos de padrões e comportamentos relevantes.

Diante deste contexto, 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 [Fayyad et al. (1996)]. Mineração de dados é o processo de análise de conjuntos de dados que tem por objetivo a descoberta de padrões interessantes que possam representar informações úteis. Estes padrões podem ser expressos na forma de regras, fórmulas e outras.

Na verdade, mineração de dados faz parte de um processo mais amplo denominado KDD (*Knowledge Discovery in Databases*) - processo de descoberta de conhecimento em bases de dados. O mesmo é composto das seguintes etapas:

1. **Limpeza dos dados:** etapa onde são eliminados ruídos e dados inconsistentes.
2. **Integração dos dados:** etapa onde diferentes fontes de dados podem ser combinadas produzindo um único repositório de dados.
3. **Redução de dados:** etapa onde se obtém uma representação reduzida do conjunto de dados original. A redução de dados pode envolver a redução do número de instâncias, de atributos e de valores de um atributo.

4. **Transformação dos dados:** etapa onde os dados são transformados no formato apropriado para aplicação de algoritmos de mineração (por exemplo, através de operações de agregação).

5. **Mineração:** etapa essencial do processo consistindo na aplicação de técnicas inteligentes a fim de se extrair os padrões de interesse.

6. **Avaliação ou Pós-processamento:** etapa onde são selecionados os padrões interessantes de acordo com algum critério do usuário.

7. **Visualização dos Resultados:** etapa onde são utilizadas técnicas de representação de conhecimento a fim de apresentar ao usuário o conhecimento minerado.

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.

Dentre as tarefas da área de mineração de dados, destaca-se a tarefa de classificação, que tem sido alvo de grande esforço de estudo e pesquisa devido à sua aplicabilidade e sucesso em diversos domínios. Desse modo, um dos grandes desafios dessa área de pesquisa é o desenvolvimento de classificadores precisos e eficientes que sejam capazes de lidar com bases de dados grandes em termos de volume e dimensão. Um aspecto importante para o bom desempenho das técnicas de classificação é a qualidade dos dados da base de treinamento. Atributos redundantes e/ou irrelevantes nas bases de dados de treinamento podem prejudicar a qualidade do classificador e, além disso, tornar o processo de construção do classificador mais lento.

Normalmente os dados disponíveis para análise não estão em um formato adequado para a etapa de mineração de dados, ou seja, é muito comum existirem bases de dados contendo ruídos, dados inconsistentes e instâncias com valores de atributos desconhecidos. Além disso, em virtude de limitações como tempo de processamento ou recursos computacionais, em muitas situações não é possível a aplicação direta dos algoritmos de mineração de dados aos dados disponíveis. Desse modo, uma fase de preparação dos dados (pré-processamento) pode ser utilizada com o intuito de melhorar a qualidade dos mesmos. Na etapa de pré-processamento podem ser realizadas transformações nos dados como: limpeza, integração, seleção, transformação e redução de dados. A redução de dados pode envolver a redução do número de instâncias, de atributos e de valores de um atributo [Weiss e Indurkhya (1998)].

A redução do número de atributos é realizada a partir da seleção de um subconjunto dos atributos existentes de modo a manter a integridade original dos dados. Existem algumas técnicas de seleção de atributos cuja abordagem considera cada atributo individualmente, enquanto outras avaliam subconjuntos de atributos com o objetivo de encontrar aquele que maximiza a acurácia preditiva do classificador.

Tradicionalmente, técnicas de seleção de atributos são executadas na fase de pré-processamento – ou preparação – dos dados e suas decisões são definitivas para a fase de construção do modelo ou classificação propriamente dita. Estas são classificadas como técnicas de seleção prévia (*eager*).

Diferentemente da abordagem *eager*, neste trabalho propõe-se uma estratégia de seleção de atributos *lazy* onde a idéia é adiar a seleção de atributos até o momento em que tivermos uma instância para ser classificada. Nesse caso, para cada instância a ser classificada, diferentes atributos poderão ser escolhidos para a execução da classificação. A expectativa é de que essa nova abordagem de seleção de atributos contribua para melhorar a acurácia preditiva do classificador.

O restante deste trabalho está organizado como especificado a seguir. O Capítulo 2 contém uma revisão bibliográfica sobre a tarefa de seleção de atributos. O Capítulo 3 apresenta a proposta central deste trabalho - a abordagem *lazy* de seleção de atributos. Os resultados obtidos nos experimentos realizados são relatados no Capítulo 4. Por fim, o Capítulo 5 apresenta as conclusões deste trabalho e propostas de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

2.1 Seleção de Atributos

Geralmente a utilização de uma estratégia de seleção de atributos tem como objetivo um ou mais dos seguintes resultados [Guyon e Elisseeff (2006)]: (a) aumentar a acurácia preditiva de um classificador; (b) reduzir o tamanho da base de dados; (c) acelerar o processamento da tarefa de classificação; e (d) simplificar o modelo de classificação gerado.

Além de as técnicas de seleção de atributos poderem seguir uma abordagem *eager* ou *lazy*, elas podem ser categorizadas como *embedded* (embutido), *wrapper* (empacotado) ou *filter* (filtro) [Liu e Motoda (2008)]. Nas seções a seguir, cada uma delas será apresentada dando-se ênfase maior para o tipo *filter*, dado que esta abordagem foi a adotada neste trabalho.

2.1.1 *Embedded*

Nas estratégias do tipo *embedded*, a seleção de atributos está diretamente incorporada no algoritmo responsável pela indução do modelo de classificação. Métodos *embedded* selecionam o conjunto de atributos no próprio processo de construção do modelo de classificação, durante a fase de treinamento, e são geralmente específicos para um dado algoritmo de classificação [Guyon e Elisseeff (2003)].

2.1.2 *Wrapper*

Métodos do tipo *wrapper* utilizam o próprio algoritmo de classificação como uma caixa preta para avaliar os subconjuntos de atributos de acordo com a sua capacidade preditiva [Guyon e Elisseeff (2003)].

As abordagens *wrapper* precisam realizar uma busca entre os possíveis subconjuntos a serem avaliados. Nos métodos *wrapper* o algoritmo de classificação é executado para cada subconjunto de atributo e a avaliação geralmente é feita em termos da acurácia preditiva retornada pelo algoritmo.

Métodos *wrapper* geralmente produzem resultados melhores (em termos de acurácia preditiva de classificação) do que *filter*, pois a seleção de atributos é guiada pelo próprio algoritmo de classificação que será usado na fase de mineração de dados [Hall (2000)]. Porém, como esse algoritmo é executado diversas vezes para subconjuntos distintos de atributos, métodos *wrapper* podem ter um custo computacional muito elevado tornando-se inviáveis em bases de dados com um número grande de atributos.

2.1.3 *Filter*

Técnicas de seleção de atributos do tipo *filter* selecionam o subconjunto de atributos de forma independente do classificador que será utilizado na fase de mineração de dados [Guyon e Elisseeff (2003)].

O nome "filtro" deriva da idéia de que os atributos irrelevantes são filtrados da base de dados antes da aplicação do algoritmo de classificação [Blum e Langley (1997)]. Os filtros usam as informações da própria base de treinamento para escolher os atributos a serem utilizados posteriormente. Algumas técnicas do tipo *filter* avaliam os atributos individualmente e escolhem um subconjunto selecionando os K atributos melhor avaliados. Esse é o procedimento utilizado pelas técnicas *Information Gain Attribute Ranking* [Yang e Pedersen (1997), Dumais et al. (1998)] e *Relief* [Kira e Rendell (1992), Kononenko (1994)]. Existem também as técnicas que avaliam subconjuntos de atributos, buscando o melhor subconjunto de acordo com alguma métrica. As técnicas mais conhecidas desse grupo são a *Correlation-based Feature Selection* [Hall (2000)] e a *Consistency-based Feature Selection* [Liu e Setiono (1996)].

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 pelos métodos que avaliam subconjuntos de atributos. Duas estratégias comumente utilizadas nos trabalhos de seleção de atributos são a *Hill-climbing* e a *Best-first* [Kohavi e John (1997)].

A seguir, será apresentada a técnica de seleção de atributos abordada neste trabalho, denominada *Consistency-based Feature Selection*.

2.1.3.1 *Consistency-based Feature Selection*

A técnica de seleção de atributos *Consistency-based Feature Selection* avalia subconjuntos de atributos utilizando a medida de consistência. Essa técnica busca por combinações de atributos cujos valores dividam os dados em subconjuntos associados a uma classe majoritária. Para isso, ela utiliza alguma heurística para percorrer o espaço de soluções em busca de um bom subconjunto de atributos.

Medida de Consistência

O valor da medida de consistência de um subconjunto de atributos S ($C(S)$) de uma base de dados D é dado por 1 - taxa de inconsistência(S, D). A taxa de inconsistência ($TI(S, D)$) é calculada da seguinte maneira:

a) Duas instâncias são consideradas inconsistentes se elas possuírem os mesmos valores de atributos e pertencerem a diferentes classes.

b) Para um conjunto de instâncias com os mesmos valores de atributos (exceto para o atributo classe), a taxa de inconsistência é dada pelo número total de instâncias desse conjunto menos o número de instâncias associadas à classe majoritária nesse conjunto. Por exemplo, seja um conjunto com n instâncias como os mesmos valores de atributos (exceto para o atributo classe), das quais c_1 instâncias estão associadas com a classe C_1, c_2 instâncias estão associadas com a classe C_2 e c_3 instâncias estão associadas com a classe C_3. Portanto, temos que $n = c_1 + c_2 + c_3$. Considerando que c_3 é o maior valor (entre c_1, c_2 e c_3), a taxa de inconsistência para esse conjunto de instâncias é dado por $n - c_3$.

c) A taxa de inconsistência de um subconjunto de atributos S de uma base de dados D é dado pela soma das taxas de inconsistências de cada um dos seus conjuntos de instâncias S_i (cada conjunto S_i é formado por instâncias que possuem os mesmos valores de atributos – exceto para o atributo classe) dividido pelo número total de instâncias de D.

A seguir temos um exemplo. Seja D a base de dados na Tabela 2.1:

Tabela 2.1: Base de Dados D.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	FALSE	yes
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

O cálculo da consistência do subconjunto de atributos S = {Outlook, Temperature, Windy} é apresentado neste exemplo. A Tabela 2.2 apresenta o cálculo da taxa de inconsistência para cada um dos conjuntos de instâncias que possuem os mesmos valores para o subconjunto de atributos em questão. A taxa de inconsistência ($TI(S, D)$) será:

$$TI(S, D) = \frac{\sum_{i=1}^{m} TI(S_i)}{numero\ de\ instancias},$$

onde m é o número de conjuntos de instâncias (padrões).

Tabela 2.2: Cálculo das taxas de inconsistência.

i	Padrão S_i Outlook, Temperature, Windy	Classe (Freq.)	Taxa de inconsistência de S_i
1	sunny, hot, FALSE	no(1), yes(1)	2 - 1 = 1
2	overcast, hot, FALSE	yes(2)	2 - 2 = 0
3	rainy, mild, FALSE	yes(2)	2 - 2 = 0
4	rainy, cool, FALSE	yes(1)	1 - 1 = 0
5	rainy, cool, TRUE	no(1)	1 - 1 = 0
6	overcast, cool, TRUE	yes(1)	1 - 1 = 0
7	sunny, mild, FALSE	no(1)	1 - 1 = 0
8	sunny, cool, FALSE	yes(1)	1 - 1 = 0
9	sunny, mild, TRUE	yes(1)	1 - 1 = 0
10	overcast, mild, TRUE	yes(1)	1 - 1 = 0
11	rainy, mild, TRUE	no(1)	1 - 1 = 0

Portanto, teremos a seguinte taxa de inconsistência ($TI(S, D)$):

$$TI(S, D) = \frac{1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0}{14} = 0,0714.$$

Por fim, a consistência desse subconjunto S será:

$$C(S) = 1 - TI(S, D) = 1 - 0,0714 = 0,929.$$

Cabe ressaltar que a consistência de um subconjunto nunca será maior que a do conjunto total de atributos [Dash e Liu (2003)].

2.2 Trabalhos Relacionados

Conforme já foi mencionado, tradicionalmente, as técnicas de seleção de atributos seguem a abordagem *eager*. Até onde se conhece, atualmente, poucos trabalhos estão relacionados com a estratégia *lazy* de seleção de atributos.

Em trabalhos anteriores, testes iniciais da seleção *lazy* de atributos foram desenvolvidos e apresentados em [Pereira et al. (2008) e Menezes et al. (2009)]. Nesses trabalhos, os atributos foram avaliados individualmente a partir de medidas baseadas nos conceitos de entropia e teste estatístico chi-quadrado. Os resultados experimentais indicaram que determinadas bases de dados podem se beneficiar desse tipo de estratégia.

A principal diferença entre os trabalhos anteriores e a proposta aqui apresentada está relacionada com forma de avaliação dos atributos para a definição do melhor subconjunto. Neste trabalho, ao invés da avaliação individual de atributos, a medida de consistência é

utilizada para avaliar conjuntos de atributos.

Capítulo 3

Abordagem Proposta

3.1 Considerações Iniciais

Em estratégias de seleção de atributos tradicionais (abordagem *eager*), os atributos são selecionados na fase de pré-processamento - ou preparação - dos dados, e aqueles não selecionados são descartados da base de dados e não participam mais do processo de classificação.

Neste trabalho, propõe-se uma estratégia de seleção de atributos do tipo *lazy*, baseada na idéia de que adiar a seleção de atributos até o ponto em que a instância é submetida ao classificador pode contribuir na identificação dos melhores atributos para a correta classificação dessa instância em particular. Para cada diferente instância a ser classificada, é possível selecionar um subconjunto de atributos distinto e mais apropriado para ela.

A seguir é dado um exemplo para ilustrar o fato de que a classificação de certas instâncias poderia se beneficiar de atributos que provavelmente seriam descartados por estratégias convencionais de seleção de atributos (abordagem *eager*). Ou seja, o exemplo evidencia que atributos podem ter maior ou menor importância dependendo da instância a ser classificada.

Na Tabela 3.1, a mesma base de dados, descrita por três atributos - X, Y, e a classe C - é representada duas vezes. A representação da esquerda está ordenada pelos valores do atributo X e a da direita está ordenada pelos valores de Y. Na parte esquerda, pode ser observado que os valores de X estão fortemente correlacionados com os valores da classe, indicando que este é um atributo importante para a tarefa de classificação. Apenas o valor $X = 4$ não discrimina perfeitamente a classe. Por outro lado, como mostrado na representação da direita, o atributo Y seria um forte candidato a ser eliminado da base por uma estratégia de seleção de atributos *eager*, já que a maioria dos seus valores não discrimina bem as classes. Porém, há uma forte correlação do valor 4 do atributo Y com o valor B do atributo classe, que seria perdido se esse atributo fosse eliminado. No caso deste exemplo, a classificação de uma instância com o valor 4 no atributo Y poderia claramente beneficiar-se com a presença desse atributo. No entanto, uma estratégia de seleção de atributos *eager* provavelmente selecionaria o atributo X em detrimento do atributo Y , independentemente das instâncias que fossem submetidas para

classificação.

Tabela 3.1: Exemplo de Base de Dados: Motivação

Base ordenada por X			Base ordenada por Y		
- X -	- Y -	- C -	- X -	- Y -	- C -
1	2	B	2	1	A
1	3	B	3	1	B
1	4	B	4	1	A
2	1	A	1	2	B
2	2	A	2	2	A
2	3	A	3	2	B
3	1	B	1	3	B
3	2	B	2	3	A
3	4	B	4	3	B
4	1	A	1	4	B
4	3	B	3	4	B
4	4	B	4	4	B

De acordo com [Liu e Motoda (2008)], uma vantagem importante de abordagens do tipo *lazy* é que elas podem responder a condições inesperadas de maneiras que não são possíveis para as abordagens *eager*, pois elas não perdem informações cruciais que poderão ser utilizadas para gerar predições mais acuradas. Logo, a principal motivação da seleção *lazy* de atributos proposta é a capacidade de acessar os valores dos atributos da instância a ser classificada, e utilizar essa informação para selecionar atributos que discriminem bem as classes para esses valores em particular. Como resultado, para cada instância, é possível selecionar atributos que são úteis para a classificação dessa instância específica.

Neste trabalho, avaliou-se a adaptação da técnica *Consistency-based Feature Selection* para a abordagem *lazy*. Essa técnica avalia subconjuntos de atributos utilizando a medida de consistência. Desse modo, a adaptação para a forma *lazy* envolveu uma adequação da métrica de consistência. Nessa abordagem *lazy*, para cada instância a ser classificada seleciona-se um subconjunto de atributos que pode (ou não) ser diferente dos subconjuntos selecionados para outras instâncias. Além disso, a técnica aqui proposta utilizou a estratégia *Best-first* para percorrer o espaço de soluções em busca de um bom subconjunto de atributos, cuja avaliação foi feita conforme a medida de consistência adaptada para abordagem *lazy*.

O restante deste capítulo encontra-se organizado como descrito a seguir. Na Seção 3.2 a adaptação da medida de consistência para a abordagem *lazy* é apresentada. A Seção 3.3 contém uma breve descrição da heurística *Best-first*. Em seguida, na Seção 3.4, pseudocódigos são utilizados para descrever o funcionamento da abordagem *lazy* para a técnica *Consistency-based Feature Selection*.

3.2 Adaptação da Medida de Consistência

A adaptação da medida de consistência para trabalharmos segundo a abordagem *lazy* foi feita da seguinte forma:

- A consistência de um subconjunto de valores de atributos S_i é dada por $C(S_i) = 1 - TI(S_i)$, onde $TI(S_i)$ é a taxa de inconsistência de S_i.
- A taxa de inconsistência de um subconjunto de valores de atributos S_i em uma base de dados D é dada pelo número total de ocorrências de S_i na base de dados D menos o número de vezes que S_i aparece na base associado com a classe majoritária (referente ao conjunto de instâncias S_i na base de dados D) dividido pelo número total de ocorrências de S_i em D.

Considerando a base de dados do exemplo apresentado na Seção 2.1.3, a consistência do subconjunto de valores S_1 ={Outlook = sunny, Temperature = hot e Windy = FALSE} será $C(S_1) = 1 - TI(S_1)$.

A taxa de inconsistência ($TI(S_1)$) será:

$$TI(S_1) = \frac{2-1}{2} = 0,5.$$

Por fim, a consistência do subconjunto de valores S_1 será:

$$C(S_1) = 1 - TI(S_1) = 1 - 0,5 = 0,5.$$

3.3 Avaliação de Subconjuntos de Atributos

Em uma avaliação de subconjuntos de atributos para uma base de 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.

A Figura 3.1 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 [Blum e Langley (1997)], a natureza dos métodos heurísticos de seleção de atributos é determinada por quatro itens básicos:

a) Ponto de partida.

b) A estratégia que será utilizada para percorrer o espaço de soluções.

c) A forma de avaliação dos subconjuntos de atributos.

d) Critério de parada.

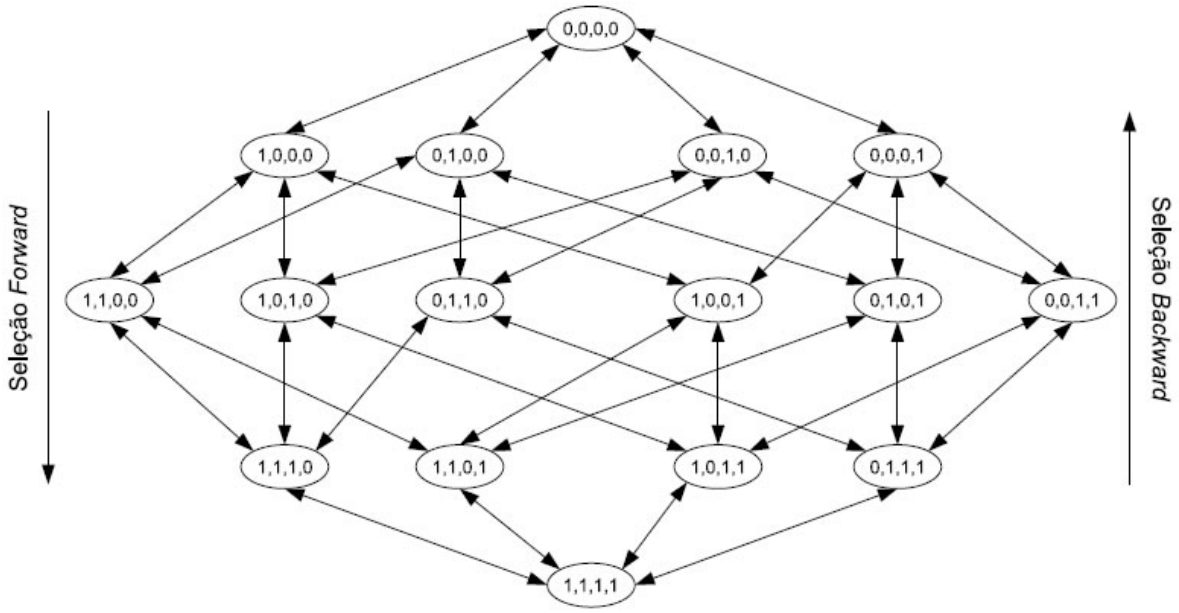

Figura 3.1: Espaço de Soluções

Em primeiro lugar, deve-se determinar o ponto(s) de partida no espaço de soluções. Pode-se, por exemplo, começar com o 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 3.1 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* [Kohavi e John (1997)]. No caso do trabalho proposto a heurística *Best-first* foi utilizada.

Na estratégia *Best-first*, 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 estratégia em questão permite que soluções geradas em

iterações anteriores, que ainda não tenham sido expandidas, se tornem a solução corrente do processo de busca.

3.4 Métodos Propostos

A seleção de atributos de forma *lazy* para a técnica *Consistency-based Feature Selection* foi realizada de duas maneiras: a seleção *lazy* pura e seleção híbrida.

A seleção de atributos *lazy* pura é uma estratégia que utiliza somente a técnica de seleção de atributos proposta aqui neste trabalho. O Algoritmo 3.1 apresenta o núcleo da estratégia *lazy* pura de seleção de atributos. São parâmetros de entrada: uma base de dados $D(A_1, A_2, ..., A_n, C)$, onde $A_1, A_2, ...$ *e* A_n são os atributos independentes e C é o atributo classe, e uma instância a ser classificada $I[v_1, v_2, ..., v_n]$. A partir desses dados, nas linhas 2 e 3 as variáveis *melhorSubconjunto* e *maiorConsistência* são inicializadas. Nas linhas 4 e 5 calcula-se a consistência *lazy* para cada subconjunto de valores gerado pela heurística *Best-first* e armazena-se essa consistência na variável CL. Feito isso, entre as linhas 6 e 8 é definido, através do valores de consistência de cada subconjunto de valores S_i, qual é o melhor subconjunto de atributo. Por fim, na linha 9 é retornado o subconjunto de atributo que foi selecionado e que será utilizado pelo algoritmo de classificação.

Algoritmo 3.1: Algoritmo de Seleção *Lazy* Pura

```
1 Procedimento SeleçãoLazyPura (D(A_1, A_2, ..., A_n, C), I[v_1, v_2, ..., v_n])
2 melhorSubconjunto ← SEM SUBCONJUNTO;
3 maiorConsistência ← −1;
4 for cada subconjunto de valores S_i ⊆ I gerado pelo Best-first do
5 |   CL ← ConsistencyLazy(S_i);
6 |   if (maiorConsistência < CL) then
7 |   |   melhorSubconjunto ← S_i;
8 |   |   maiorConsistência ← CL;
9 Retorne melhorSubconjunto;
```

Além da seleção de atributos *lazy* pura, desenvolveu-se nesse trabalho uma estratégia de seleção de atributos híbrida. Essa estratégia mescla a seleção de atributos *eager* e a técnica de seleção de atributos *lazy* pura aqui proposta.

O desenvolvimento da seleção de atributos híbrida se deve ao fato de que na seleção de atributos *lazy*, alguns subconjuntos de valores de atributos selecionados podem ter sua medida de consistência não tão boa quanto a medida de consistência obtida com a seleção de atributos *eager*. Acontecendo isso, um determinado subconjunto de atributos selecionado de forma *lazy*, provavelmente não seria tão interessante para se fazer classificação de uma instância específica. Diante disso, desenvolveu-se uma estratégia híbrida que compara a consistência do

subconjunto obtido na seleção de atributos *lazy* com a medida de consistência do subconjunto obtido na seleção de atributos *eager*. A partir disso, define-se qual das duas seleções de atributos será considerada para uma determinada instância. O Algoritmo 3.2 apresenta o núcleo da estratégia utilizada na seleção hibrida. São parâmetros de entrada: uma base de dados $D(A_1, A_2, ..., A_n, C)$, onde $A_1, A_2, ...$ *e* A_n são os atributos independentes e C é o atributo classe, e uma instância a ser classificada $I[v_1, v_2, ..., v_n]$. A partir desses dados, nas linhas 2 e 3 as variáveis *melhorSubconjunto* e *maiorConsistência* são inicializadas. Na linha 4, o maior valor de consistência obtido na seleção de atributos *eager* é guardado na variável *maiorConsistênciaEager*, e na linha 5 o subconjunto de atributos selecionado de forma *eager* é armazenado na variável *melhorSubconjuntoEager*. Entre as linhas 6 e 10, seleciona-se e armazena-se usando a estratégia *lazy* pura, um subconjunto de atributo e o seu valor de consistência. Após isso, na linha 11 é feita uma comparação que verifica se o melhor valor de consistência obtido na seleção *lazy* é maior ou igual ao melhor valor de consistência obtido na seleção *eager*, e se o melhor subconjunto de atributos selecionados de forma *lazy* é menor do que o melhor subconjunto de atributos selecionado de forma *eager*. Se a condição for verdadeira retorna-se o subconjunto atributo selecionado com a abordagem *lazy* (Linha 12), caso contrário, retorna-se o subconjunto de atributo selecionado com a abordagem *eager* (Linha 14).

Algoritmo 3.2: Algoritmo de Seleção Híbrida

```
1  Procedimento SeleçãoHibrida (D(A_1, A_2, ..., A_n, C), I[v_1, v_2, ..., v_n])
2  melhorSubconjuntoLazy ← SEM SUBCONJUNTO;
3  maiorConsistênciaLazy ← −1;
4  maiorConsistênciaEager ← ConsistencyEager(D);
5  melhorSubconjuntoEager ← SUBCONJUNTO SELECIONADO DE FORMA EAGER;
6  for cada subconjunto de valores S_i ⊆ I, gerado pelo Best-first do
7      CL ← ConsistencyLazy(S_i);
8      if (maiorConsistênciaLazy < CL) then
9          melhorSubconjuntoLazy ← S_i;
10         maiorConsistênciaLazy ← CL;
11 if
   (melhorConsistênciaLazy ≥ melhorConsistênciaEager)∧(Tamanho(melhorSubconjuntoLazy) < Tamanho(melhorSubconjuntoEager))
   then
12     Retorne melhorSubconjuntoLazy;
13 else
14     Retorne melhorSubconjuntoEager;
```

Através do subconjunto de atributos selecionado, o algoritmo de classificação deverá estimar a classe da instância $I[v_1, v_2, ..., v_n]$.

Capítulo 4

Experimentos e Resultados

Neste trabalho, optou-se por realizar os experimentos utilizando-se o algoritmo de classificação k-NN [Cover e Hart (1967), Dasarathy (1991)]. Os algoritmos de seleção *lazy* pura e seleção híbrida foram integrados à implementação original do k-NN, disponível na ferramenta Weka com o nome IBK. O objetivo principal dos experimentos realizados é mostrar que, para determinadas bases de dados, a estratégia de seleção de atributos *lazy* pode ser mais eficiente que a seleção de atributos feita de forma *eager*. Para tanto, serão comparadas a estratégia de seleção de atributos *eager*, do tipo *filter*, baseada na medida de consistência, implementada na ferramenta Weka (*Waikato Environment for Knowledge Analysis*) [Witten e Frank (2005)] com o nome *Consistency-based Feature Selection*, e a estratégia *lazy* proposta neste trabalho, que foi implementada em Java e incorporada à mesma ferramenta. Em todos os experimentos realizados neste trabalho, os parâmetros foram configurados para os seus valores *default* na ferramenta Weka versão 3.7.1 (*Developer version*).

A acurácia preditiva foi o critério utilizado na avaliação comparativa entre o método proposto e a seleção de atributos *eager*. Os valores de acurácia preditiva obtidos pelos classificadores foram avaliados a partir de um procedimento de validação cruzada (*Cross-validation*) com dez partições [Han e Kamber (2006)], na qual cada partição foi obtida de maneira aleatória. A acurácia dos classificadores corresponde a uma média dos valores obtidos em cada partição. As mesmas partições foram utilizadas nos experimentos de cada técnica empregada.

A seleção *lazy* ocorre quando o classificador recebe uma nova instância a ser classificada. Como para cada nova instância um subconjunto distinto de atributos deve ser considerado pelo classificador, os atributos não selecionados pelas estratégias *lazy* para uma dada instância não são removidos da base de dados, mas apenas desconsiderados pelo procedimento de classificação.

Na Tabelas 4.1, 4.2 e 4.3 apresenta-se as acurácias preditivas do classificador k-NN, com k igual a 1, k igual a 3 e k igual a 5, tanto para as estratégias de seleção *lazy* e híbrida, quanto para a estratégia de seleção *eager*. A avaliação comparativa das técnicas foi realizada com 3 bases pertencentes ao repositório de dados *UCI Machine Learning Repository* [Asuncion

e Newman (2007)]. As bases utilizadas são: *Wine* (com 13 atributos, 178 instâncias e 3 classes), *Iris* (com 4 atributos, 222 instâncias e 3 classes) e *Breast-cancer* (com 9 atributos, 286 instâncias e 2 classes). As linhas das tabelas referem-se aos resultados dos testes para as bases citadas anteriormente. Valores em negrito indicam os resultados superiores e os valores entre colchetes correspondem ao desvio padrão.

Observa-se na Tabela 4.1 que, para a base *Wine*, as estratégias de seleção *lazy* e híbrida obtiveram os melhores resultados para todos os valores de k. O valor de acurácia (**98,30%**) para as duas estratégias permaneceu sempre o mesmo, independente do valor do parâmetro k. Já para a seleção *eager* na medida em que o k foi aumentando o valor de acurácia diminuiu. Esses resultados ilustram que a seleção de atributos *lazy* e híbrida foram mais eficientes (levando-se em consideração o valor de acurácia) do que a seleção *eager*.

A Tabela 4.2 apresenta os resultados obtidos para a base *Iris*. Observa-se que a seleção de atributos *eager* e a seleção híbrida, obtiveram os melhores resultados para todos os valores de k. A medida em que k aumentou, o valor de acurácia do classificador com a seleção *eager* e com seleção híbrida aumentou, já com a seleção *lazy* permaneceu o mesmo. Esses resultados ilustram que a seleção de atributos híbrida e *eager* foram igualmente as melhores.

Conforme a Tabela 4.3, no caso da base *Breast-cancer* observa-se que as estratégias de seleção *lazy* e híbrida obtiveram os melhores resultados para $k = 1$. Porém para $k = 3$ e $k = 5$, a seleção *eager* obteve melhores resultados. Pode-se afirmar então que para essa base, na maioria dos testes, a seleção de atributos *eager* proporcionou os melhores resultados de acurácia.

Tabela 4.1: Resultados obtidos para a base de dados *Wine*

wine	**k-NN** ($k = 1$)	**k-NN** ($k = 3$)	**k-NN** ($k = 5$)
Eager	95,56 [5,74]	94,97 [4,87]	90,42 [6,00]
Lazy	**98,30** [3,79]	**98,30** [3,79]	**98,30** [3,79]
Híbrido	**98,30** [3,79]	**98,30** [3,79]	**98,30** [3,79]

Tabela 4.2: Resultados obtidos para a base de dados *Iris*

iris	**k-NN** ($k = 1$)	**k-NN** ($k = 3$)	**k-NN** ($k = 5$)
Eager	**94,00** [5,21]	**94,67** [5,56]	**96,00** [4,84]
Lazy	92,67 [5,21]	92,67 [5,21]	92,67 [5,21]
Híbrido	**94,00** [5,21]	**94,67** [5,56]	**96,00** [4,84]

Tabela 4.3: Resultados obtidos para a base de dados *Breast-cancer*

breast-cancer	**k-NN** ($k = 1$)	**k-NN** ($k = 3$)	**k-NN** ($k = 5$)
Eager	67,07 [8,58]	**71,24** [9,02]	**73,04** [6,68]
Lazy	**70,23** [5,75]	70,92 [6,02]	70,57 [6,91]
Híbrido	69,85 [6,45]	70,57 [5,64]	70,57 [6,91]

Todos os experimentos foram conduzidos em uma máquina Intel Pentium Dual Core 3.0 GHz com 2 Gbytes de RAM. O tempo de CPU médio gasto na classificação de uma instância utilizando-se o procedimento de seleção de atributos *lazy* variou de 7,40 milissegundos (para a base de dados *Iris*) até 31,55 milissegundos (para a base *Breast-cancer*). Já o tempo médio de CPU gasto na classificação de uma instância quando a abordagem eager foi utilizada para seleção dos atributos variou de 1,48 milissegundos (para a base de dados *Iris*) até 1,75 milissegundos (para a base *Breast-cancer*).

Portanto, o custo computacional da estratégia *lazy* para seleção de atributos e classificação foi maior do que o custo computacional da estratégia *eager*. Isso acontece porque na estratégia *lazy* a seleção de atributos é realizada toda vez que uma instância é submetida para classificação, enquanto que na estratégia *eager* a seleção de atributos é realizada somente uma vez antes da etapa de classificação.

Capítulo 5

Conclusões

Neste trabalho foi proposta uma abordagem *lazy* de seleção de subconjuntos de atributos para o problema de classificação. Trata-se de uma proposta em que a seleção de atributos não ocorre na fase de pré-processamento dos dados (como na abordagem *eager*), mas é adiada para o momento da classificação de uma instância. Acredita-se que, dessa forma, a seleção pode se beneficiar do conhecimento dos valores dos atributos da instância a ser classificada. Os resultados experimentais apresentados neste trabalho indicam que determinadas bases de dados podem se beneficiar das estratégias aqui propostas (seleção *lazy* pura e seleção híbrida).

Como trabalhos futuros sugere-se:

- Avaliar essa estratégia de seleção de atributos *lazy* com um número maior de bases de dados.
- Avaliar os casos em que a seleção de atributos *lazy* não se mostra interessante para tentar compreender as limitações dessa abordagem.

Referências Bibliográficas

Asuncion, A. e Newman, J. (2007). Uci machine learning repository. University of California, Irvine. http://www.ics.uci.edu/ mlearn/MLRepository.html.

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

Cover, T. e Hart, P. (1967). Nearest neighbor pattern classification. pp. 13(1):21–27. IEEE Transactions on Information Theory 13.

Dasarathy, B. V. (1991). Nearest neighbor (nn) norms: Nn pattern classification techniques. IEEE Computer Society Press.

Dash, M. e Liu, H. (2003). Consistency-based search in feature selection. pp. 155–176. Artificial Intelligence 151 (2003).

Dumais, S.; Platt, J.; Heckerman, D. e Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. pp. 149–155. In Proceedings of the International Conference on Information and Knowledge Management.

Fayyad, U.; Piatetsky-Shapiro, G. e Smyth, P. (1996). From data mining to knowledge discovery in databases. pp. 17(3):37–54. Artificial Intelligence Magazine.

Guyon, I. e Elisseeff, A. (2003). An introduction to variable and feature selection. pp. 1157–1182. Journal of Machine Learning Research 3.

Guyon, I. e Elisseeff, A. (2006). An introduction to feature extraction. pp. 1–24. In Guyon, I.,Gunn, S., Nikravesh, M., and Zadeh, L., editors, Feature Extraction, Foundations and Applications. Springer.

Hall, M. A. (2000). A correlation-based feature selection for discrete and numeric class machine learning. ICML'00, pp. 1157–1182. In Proceedings of the 17th International Conference on Machine Learning.

Han, J. e Kamber, M. (2006). Data mining: Concepts and techniques. Morgan Kaufmann, 2nd edition.

Kira, K. e Rendell, L. (1992). A practical approach to feature selection. ICML'92, pp. 249–256. In Proceedings of the 9th International Conference on Machine Learning.

Kohavi, R. e John, G. H. (1997). Wrappers for feature subset selection. pp. 273–34. Artificial Intelligence 97.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. ECML'94, pp. 171–182. In Proceedings of the 7th European Conference on Machine Learning.

Liu, H. e Motoda, H. (2008). Less is more. pp. 3–17. In Liu, H. and Motoda, H., editors, Computational Methods of Feature Selection. Chapman Hall/CRC.

Liu, H. e Setiono, R. (1996). A probabilistic approach to feature selection: A filter solution. pp. 319–327. In Proceedings of the 13th International Conference on Machine Learning, M. Kaufmann, Ed.

Menezes, R. W.; Plastino, A.; Zadrozny, B.; Pereira, R. B.; Merschmann, L. e Freitas, A. A. (2009). Avaliação de uma nova medida para seleção lazy de atributos baseada no teste chi-quadrado. In *Anais do V Workshop em Algoritmos e Aplicações de Mineração de Dados*, Fortaleza, CE. V Workshop em Algoritmos e Aplicações de Mineração de Dados, em conjunto com o XXIV Simpósio Brasileiro de Banco de Dados.

Pereira, R. B.; Plastino, A.; Zadrozny, B.; Merschmann, L. e Freitas, A. A. (2008). Seleção lazy de atributos - uma nova perspectiva. In *Anais do IV Workshop em Algoritmos e Aplicações de Mineração de Dados*, pp. 1–9, Campinas, SP. IV Workshop em Algoritmos e Aplicações de Mineração de Dados, em conjunto com o XXIII Simpósio Brasileiro de Banco de Dados.

Weiss, S. M. e Indurkhya, N. (1998). Predictive data mining: A practical guide. Morgan Kaufmann Publishers, San Francisco, CA.

Witten, I. H. e Frank, E. (2005). Data mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2nd edition.

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