Técnicas de Seleção de Atributos utilizando Paradigmas de Algoritmos Disciplina de Projeto e Análise de Algoritmos

Theo Silva Lins, Luiz Henrique de Campos Merschmann
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: theosl@gmail.com,luizhenrique@iceb.ufop.br

***Resumo*—O processo de descoberta de conhecimento em bases de dados – processo KDD (*Knowledge Discovery in Databases*) – é formado basicamente por três etapas: pré-processamento dos dados, mineração de dados e pós-processamento. Uma das possíveis tarefas no pré-processamento dos dados é a redução do número de atributos das bases de dados. Essa redução do número de atributos é realizada a partir de técnicas de seleção de atributos. Uma dessas ténicas de seleção de atributos é a *Consistency-based Feature Selection*, que é baseada na consistência dos subconjuntos de atributos. No trabalho foi feito um estudo sobre essa técnica de seleção de atributos e sua métrica, além da implementação dos algoritmos com a utilização dos paradigmas de programação estudados na disciplina de PAA.**

***Keywords*-seleção de atributos, mineração de dados, consistência.**

I. Introdução

Mineração de dados é o processo de descoberta de informações relevantes a partir de um grande conjunto de dados [1]. Com o grande avanço da tecnologia, um grande número de informações estão sendo guardadas de formas digitais criando-se assim várias bases de dados, com informações utéis para diversas áreas.

A mineração de dados é dividida em várias tarefas e a classificação é uma das tarefas mais importantes. 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 [2]. Com isso um dos grandes desafios 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 [3].

Um problema que pode ocorrer nas bases de dados é o grande número de atributos. A seleção de atributos é um processo que visa identificar e remover a maior quantidade possível de informações irrelevantes e redundantes. De maneira geral, nem todos os atributos da base de dados são necessários para discriminar a classe de maneira precisa, e incluí-los no modelo de classificação pode até mesmo gerar resultados inferiores do que seriam obtidos se eles fossem removidos da base.

A seleção de atributos tem recebido atenção especial em aplicações que usam base de dados com muitos atributos. Exemplos:

- Processamento de texto.
- Recuperação de informação em banco de imagens.
- Bioinformática.

Foram desenvolvidas diferentes técnicas de seleção de atributos, que utilizam critérios de seleção e algoritmos de busca distintos para avaliar e encontrar de forma heurística o subconjunto de atributos mais adequado.

Essas técnicas de seleção de atributos geralmente são divididas em três categorias:

- *Embedded*
- *Wrapper*
- *Filtro*

As estratégias do tipo *embedded* são diretamente incorporadas no algoritmo responsável pela indução do modelo de classificação. E as estratégias *wrapper* e filtro são executadas em uma fase de pré-processamento dos dados. Na filtro o subconjunto de atributos é avaliado por uma medida independente e na *wrapper* é utilizado o próprio algoritmo de classificação. Neste trabalho foi usada uma técnica de seleção que se encaixa na categoria filtro.

Técnicas do tipo filtro podem avaliar os atributos individualmente e escolher os melhores, como nas técnicas *Information Gain Attribute Ranking* [3]. e *Relief* [4], ou podem avaliar subconjuntos de atributos, buscando de forma heurística o melhor subconjunto. Neste último caso, as técnicas mais conhecidas são a *Correlation-based Feature Selection* [5]. e a *Consistency-based Feature Selection* [6]. A *Consistency-based Feature Selection* foi a utilizada na implementação dos algoritmos com os paradigmas de programação estudados em PAA.

O artigo está organizado da seguinte forma. A Seção II

apresenta estudo feito sobre a técnica de seleção baseada em consistência, suas métricas e algoritmos. As análises de complexidade são apresentados na Seção III. O experimentos realizados com os algoritmos de consistência são apresentados na Seção IV. Na Seção V estão as conclusões sobre o trabalho.

II. Métodos

A métrica baseada em consistência proposta em [6] avalia a qualidade de um subconjunto de atributos pelo nível de consistência dos valores da classe quando as instâncias de treinamento são projetadas no subconjunto de atributos.

A consistência de um subconjunto nunca será maior que a do conjunto total de atributos, logo a prática usual é utilizar essa métrica em conjunto com uma busca exaustiva ou heurística que procure pelos menores subconjuntos com a consistência mais próxima possível da consistência do conjunto total de atributos [7].

A. Medidas de Consistência

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

1) Duas instâncias são consideradas inconsistentes se elas possuirem os mesmos valores de atributos e pertencerem a diferentes classes.
2) Para um conjunto de instâncias com os mesmos valores de atributos (exceto para o atributo classe), a taxa de insconsistência é dada pelo número total de instâncias desse conjunto menos o número de instâncias associadas com a 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 $c1$ instâncias estão associadas com a classe $C1$, $c2$ instâncias estão associadas com a classe $C2$ e $c3$ instâncias estão associadas com a classe $C3$. Portanto, temos que $n = c1 + c2 + c3$. Considerando que $c3$ é o maior valor (entre $c1$, $c2$ e $c3$), a taxa de inconsistência para esse conjunto de instâncias é dado por $n - c3$.
3) A taxa de inconsistência um subconjunto de atributos S de uma base de dados D é dado pela soma das taxas de insconsistências 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 um exemplo. Seja D a seguinte base de dados ilustrada na figura 1:

Neste exemplo é calculado a consistência do subconjunto de atributos $S = Outlook, Temperature, Windy$.

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

Figura 1: Base de Dados do Golf

A Figura 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.

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

Figura 2: Cálculo das taxas de inconsistência.

A taxa de inconsitência $(TI(S, D))$ será:
$TI(S, D) = \frac{\sum_{i=1}^{m} TI(S_i)}{numerodeinstancias}$,
onde m é o número de conjuntos de instâncias (padrões).

Portanto, teremos a seguinte taxa de inconsitência $(TI(S, D))$:
$TI(S, D) = \frac{1+0+0+0+0+0+0+0+0+0+0}{14} = 0,0714.$
Por fim, a consistência desse subconjunto S será:
$C(S) = 1 - TI(S, D) = 1 - 0,0714 = 0,929.$

B. Algoritmos e Heurísticas

A idéia dos algoritmos de consistência é gerar subconjuntos de atributos e avaliá-los quanto ao seu tamanho e a sua inconsistência em relação à classe. Ao final, o subconjunto de atributos selecionado será aquele que, possuir a menor inconsistência e o menor tamanho.

Em uma base de dados com n atributos, tem-se 2^n subconjuntos de atributos possíveis. Portanto, realizar uma busca exaustiva nesse espaço de solucões 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 [2].

A Figura 3 mostra um grafo que representa todos os subconjuntos de atributos possíveis para um problema contendo quatro atributos. Os nós desse grafo são rotulados com sequências 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.

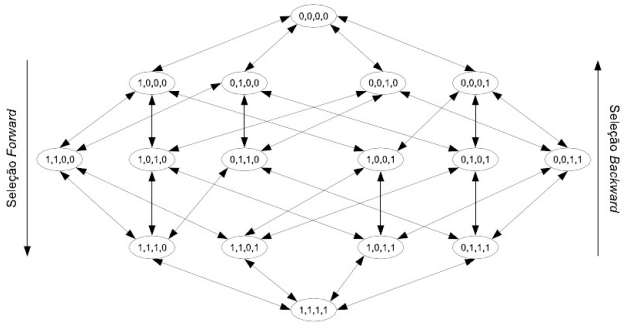

Figura 3: Espaço de soluções para um problema contendo quatro atributos [2]

Segundo os autores do trabalho proposto em [8], 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 influência diretamente na escolha dos operadores que serão utilizados para percorrer o 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*.

O próximo passo consiste em determinar qual estratégia será utilizada para se percorrer o espaço de soluções. A forma de avaliação nesse caso é a de consistência. Para finalizar deve-se definir um critério de parada, que podem ser número de iterações, soluções não encontradas, entre outras.

C. Algoritmos Implementados

O algoritmo *Backtracking* quando executa uma base de dados com um grande número de atributos, torna-se computacionalmente inviável, devido ao espaço de soluções crescer exponencialmente. A sua implementação tem como objetivo testar com até quantos números em média de atributos em um base de dados ele ainda é viável, apesar de que ele também pode ser combinado a uma heurística e obter melhores resultados. No entanto o Backtracking sempre consegue chegar a solução ótima. A implementação do algoritmo Max Tries foi realizada a partir do pseudocodigo citado pelo próprio autor [6]. Esse algoritmo funciona da seguinte maneira:

- Define-se o número de tentativas e taxa de inconsitência máxima aceitável.
- A cada tentativa ele sorteia um subconjunto de atributos.
- Verifica se a inconsistência desse subconjunto é aceitável.
- Imprime o subconjunto.

O algoritmo Max-Tries é baseado no algoritmo *Las Vegas Filter* (LVF) [9], para seleção de subconjuntos de atributos, onde a probabilidade de gerar qualquer subconjunto são iguais. O LVF utiliza a taxa de consistência mínima como medida de avaliação, após gerar o subconjunto de atributos ele testa o valor da consistência obtida e somente irá imprimir o resultado se o valor for maior que a consistência mínima. Na figura 4 segue o pseudo-código do LVF.

Algorithm. LVF
Input: Data D, feature set S, $MaxTries$
Output: Consistent Feature Subset
1. $\delta = \text{inConCal}(S, D)$
2. $T = S$
3. for $j = 1$ to $MaxTries$
4. randomly choose a subset of features, S_j
5. if $|S_j| \leqslant |T|$
6. if $inConCal(S_j, D) \leqslant \delta$
7. if $|S_j| < |T|$
8. $T = S_j$
9. output S_j
10. else
11. append S_j to T
12. output S_j as 'yet another solution'
13. return T

Figura 4: Algoritmo Las Vegas Filter

No algoritmo *hill-climbing*, em cada iteração, expande-se o subconjunto corrente gerando-se várias soluções vizinhas, e move-se para o melhor subconjunto gerado. O processo é interrompido quando nenhum das subconjuntos gerados na expansão for melhor do que o subconjunto corrente.

III. Análise de Complexidade

A complexidade do algoritmo *Max-Tries* varia de acordo com o número de tentativas que for definido, com isso sua complexidade de tempo será $f(n) = O(m)$, onde m é o número de tentativas, o algoritmo será finalizado assim que o número de tentativas terminar.

A complexidade do algoritmo *Backtracking* varia de acordo com o número de atributos, ele possui uma complexidade de tempo $f(n) = O(2^n)$ onde n é o número de atributos. Por que em todos os casos ele será exponencial pois executará sempre todas as soluções possíveis. Portanto ele sempre retornará a solução ótima.

IV. Experimentos

Os experimentos foram realizados com as bases de dados retiradas de [10]. O objetivo dos testes era fazer uma comparação entre os algoritmos implementados. Para isso todos foram implementados na mesma plataforma de desenvolvimento: DEV C++ 4.9.9.2, utilizado-se a mesma linguagem C++, e o mesmo computador para realização dos testes.

No algoritmo *Max-Tries* que necessita da definição da variável tentativas foi utilizado 1000, (com isso é determinada a parada do algoritmo) e 0,8 na variável consistência mínima (o que descartava as demais consistências). Nas Tabelas I e II, podemos verificar que nenhum dos algoritmos teve superiodade absoluta.

O algoritmo *Max-Tries* foi superior ao *backtracking* somente na base de dados *Spectf Heart*, pois o *backtracking* não foi concluído devido tempo que gastaria o experimento, devido a base de dados ter 44 atributos, o que leva a um espaço de soluções superior a 17 trilhões.

Já o *Backtracking* obteve superioridade na base de dados *Breast Cancer*, que continha 10 atributos e também na *Congress Voting* que apesar de ter obtido os mesmos números de consistência, o número de atributos selecionados foi menor, mas o tempo de execuçao que é em segundos foi quase 20 vezes maior. E na base de dados Golf os dois algoritmos obtiveram os mesmos resultados.

Tabela I: Max Tries

Base de dados	Instancias	Atrib.	Atrib. Selecionados	Tempo	Consistência
Golf	14	4	3	0,001	0.928
Breast Cancer	699	10	10	19,4	0,992
SPECTF Heart	187	44	9	19,9	0,994
Congress. Voting	435	16	10	41,8	0,990

Tabela II: Backtracking

Base de dados	Instancias	Atrib.	Atrib. Selecionados	Tempo	Consistência
Golf	14	4	3	0,001	0.928
Breast Cancer	699	10	7	27,1	1
SPECTF Heart	187	44	X	X	X
Congress. Voting	435	16	9	611,7	0,990

V. Conclusões

A técnica de seleção de atributos é uma implementação importante para a mineração de dados, que busca melhorar o desempenho dos classificadores através da redução de atributos. Um estudo foi feito em cima da técnica *Consistency-based Feature Selection* e uma métrica, que é baseada na consistência dos subconjuntos dos atributos.

Foram feitas implementações utilizando esta técnica. Com os experimentos podemos concluir que o *backtracking* demostrou bons resultados com a base de dados que contém um número pequeno de atributos e que com base de dados maiores é melhor se utilizar uma heurística ou o próprio *backtracking* juntamente com uma heurística. O *Max-Tries* possue o número de tentativa como um parâmetro, o que permite alterar de acordo com a base de dados. Também é possível configurar uma taxa mínima de consistência, de acordo com sua necessidade.

Nos algoritmos de seleção quanto maior o número de atributos, maior a chance de se ter uma consistência alta, com isso o algoritmo *Max-Tries* sempre conseguiu valores altos de consistência, mas o grande desafio da seleção de atributos é conseguir uma consistência alta juntamente com número baixo de atributos.

Referências

[1] R. B. Pereira, "Seleção lazy de atributos para a tarefa de classificação," Master's thesis, UFF - Universidade Federal Fluminense, Brazil, Julho 2009.

[2] L. H. de Campos Merschmann, "Classificação probabilística baseada em análise de padrões," PhD Thesis, UFF - Universidade Federal Fluminense, Brazil, 2007, www.ic.uff.br/ plastino/TeseLuiz.pdf.

[3] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization." 1997, pp. 412–420.

[4] I. Kononenko, "Estimating attributes: Analysis and extensions of relief." 1994, pp. 171–182.

[5] M. A. Hall, "A correlation-based feature selection for discrete and numeric class machine learning," vol. 17, 2000, pp. 359–366.

[6] H. Liu and R. Setiono, "A probabilistic approach to feature selection a filter solution," vol. 825, 1996, pp. 319–327.

[7] M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge ande Data Engineering*, vol. 6, pp. 1437–1447, 2003.

[8] A. L. Blum and P. Langley, “Selection of relevant features and examples in machine learning.” *Artificial Intelligence 97*, vol. 97, no. 1-2, pp. 245–271, 1996.

[9] G. Brassard and P. Bratley, *Fundamentals of Algorithms*, 1st ed. Prentice Hall, 1996.

[10] D. Aha, “Uci - machine learning repository,” 2011, http://archive.ics.uci.edu/ml/datasets.html, visitado em 10/07/2011.