Melhorando a Performance do Algoritmo Naive Bayes para Regressão Através da Combinação de Atributos

Aloísio Carlos de Pina, Gerson Zaverucha

COPPE/Sistemas – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68.511 – 21.945-970 – Rio de Janeiro – RJ – Brasil
{long,gerson}@cos.ufrj.br

Abstract. *Naive Bayes for Regression (NBR) uses the Naive Bayes methodology to numeric prediction tasks. The main reason for its poor performance is the independence assumption. Although many recent researches try to improve the performance of Naive Bayes by relaxing the independence assumption, none of them can be directly applied to the regression framework. The objective of this work is to present a new approach to improve the results of the NBR algorithm, by combining attributes by means of auxiliary regression algorithms.*

Resumo. *O algoritmo Naive Bayes para Regressão (NBR) usa a metodologia do Naive Bayes para tarefas de predição numéricas. A principal razão de sua pobre performance é a suposição de independência. Embora muitas pesquisas recentes tentem melhorar a performance do Naive Bayes pelo relaxamento da suposição de independência, nenhuma delas pode ser aplicada diretamente a problemas de regressão. O objetivo deste trabalho é apresentar uma nova abordagem para melhorar os resultados do algoritmo NBR, combinando atributos por meio de algoritmos de regressão auxiliares.*

1. Introdução

O Classificador Naive Bayes (NBC) [Duda e Hart 1973] baseia-se na suposição de independência condicional entre os atributos dada a classe. Entretanto, essa suposição raramente é verdadeira em problemas reais. Assim, embora o algoritmo Naive Bayes tenha se mostrado competitivo quando comparado a outros algoritmos mais complexos e que representam o estado-da-arte para o problema de classificação, pesquisas recentes tentam melhorar sua performance pelo relaxamento da suposição de independência. Friedman e Goldzmidt (1996) propuseram o Augmented Naive Bayes (Naive Bayes aumentado), que pode expressar relações parciais de dependência entre atributos. Uma vez que encontrar o melhor Augmented Naive Bayes é um problema NP-difícil, Friedman e Goldzmidt restringiram a rede a uma topologia de árvore, de modo que pode-se encontrar eficientemente a melhor Tree-Augmented Naive Bayes (TAN). Keogh e Pazzani (2002) apresentaram dois métodos para encontrar a melhor TAN: um baseado em uma busca hill-climbing e outro, mais eficiente do ponto de vista computacional, chamado SuperParent. Outras abordagens foram propostas para melhorar TANs (veja, por exmplo, [Zhang e Ling 2001], [Wang *et al.* 2003] e [Wang *et al.* 2004]). Lazy Bayesian Rules (LBR) [Zheng e Webb 2000], nas quais cada atributo depende não somente da classe, mas também depende de um conjunto de atributos escolhidos, alcançam performance comparável às TANs. Recentemente, Webb *et al.*

(2005) propuseram os Averaged One-Dependence Estimators (AODE), que fazem a média de todos os modelos a partir de uma classe restrita de classificadores de dependência única. Dentre outras abordagens desenvolvidas para aumentar a precisão do NBC está incluída [Kohavi 1994], na qual usam-se wrappers para selecionar um subconjunto de atributos. Embora essa abordagem possa diminuir o problema causado pela suposição de independência, ela não lida diretamente com o problema de atributos relacionados. Kononenko (1991) propôs a combinação de dois ou mais atributos relacionados a fim de lidar com o problema: os atributos originais são substituídos por uma combinação deles, cujos valores possíveis são as combinações de todos os valores possíveis para cada um dos atributos originais (atributos numéricos são discretizados). Um teste estatístico é realizado para decidir que atributos devem ser combinados. Pazzani (1996) usa uma abordagem similar, combinando atributos baseando-se na acurácia preditiva, conseguindo resultados muito melhores.

O algoritmo Naive Bayes para Regressão (NBR) [Frank *et al.* 2000] usa a metodologia do Naive Bayes para tarefas de predição numéricas, modelando a distribuição de probabilidade do valor objetivo com estimadores de densidades por kernels. Entretanto, a incrível performance do Naive Bayes para problemas de classificação não é observada para problemas de regressão. Baseados em resultados experimentais que isolam a suposição de independência como a principal razão da pobre performance do NBR, Frank *et al.* (2000) concluíram que o algoritmo só deve ser aplicado a problemas de regressão quando a suposição de independência é verdadeira. Infelizmente, alguns dos métodos mencionados anteriormente não podem ser aplicados ao NBR e outros necessitariam do uso de discretização. O objetivo deste trabalho é apresentar a análise e avaliação experimental de uma nova abordagem para melhorar os resultados do algoritmo NBR. Isso é realizado através da combinação de atributos por meio de algoritmos de regressão auxiliares.

Este artigo está organizado como segue. A próxima seção contém uma revisão sobre o algoritmo Naive Bayes para Regressão e suas características mais relevantes para esta pesquisa. Então nossa abordagem para melhorar a performance do NBR é apresentada na Seção 3. Na Seção 4, é reportada uma extensa avaliação experimental comparando os resultados alcançados pela nova abordagem com os resultados alcançados pelo NBR. Finalmente, na Seção 5, as conclusões e os planos para pesquisas futuras são apresentados.

2. Naive Bayes para Regressão

Seja C um valor objetivo numérico e A um exemplo consistindo de m atributos A_1, ..., A_m, onde A_i é numérico ou nominal. O algoritmo Naive Bayes para Regressão tem que predizer C dado A. Uma vez que $p(C|A)$ é usualmente desconhecido, deve ser estimada a partir dos dados. O algoritmo NBR realiza isso pela aplicação do Teorema de Bayes e assumindo independência dos atributos A_1, ..., A_m dado o valor objetivo C:

$$P(C \mid A) = \frac{P(A_1 \mid C) \cdots P(A_m \mid C) P(C)}{\int P(A_1 \mid C) \cdots P(A_m \mid C) P(C) dC}$$

As funções de densidade de probabilidade individuais $p(A_i|C)$ e a priori $p(C)$ têm que ser estimadas a partir de um conjunto de exemplos de treinamento. Para $p(A_i|C)$ há

dois casos a serem considerados: o caso onde o atributo A_i é numérico e o caso onde ele é nominal.

Se A_i é um atributo numérico, ele é normalizado e a probabilidade condicional $p(A_i|C)$ é estimada computando-se uma aproximação para a probabilidade conjunta $p(A_i,C)$, uma vez que:

$$P(A_i \mid C) = \frac{P(A_i, C)}{\int P(A_i, C) dA}$$

Isso é feito por meio de um estimador de densidades por kernels:

$$\hat{P}(A_i = a, C = c) = \frac{1}{n h_{Ai} h_C} \sum_{j=1}^{n} K\left(\frac{a - a_j}{h_{Ai}}\right) K\left(\frac{c - c_j}{h_C}\right)$$

onde a_j é o valor do atributo, c_j é o valor objetivo do exemplo de treinamento j, $K(.)$ é uma dada função kernel, e h_{Ai} e h_C são as larguras dos kernels para A_i e C.

Se A_i é um atributo nominal, com um conjunto de valores não ordenados v_1, v_2, ..., v_o, ao invés de estimar $p(A_i|C)$ diretamente, o problema é transformado no de estimar $p(C|A_i)$, a pdf de uma variável numérica dada uma nominal, e $p(A_i)$, a probabilidade a priori do valor nominal de um atributo:

$$P(A_i = v \mid C = c) = \frac{P(A_i = v) P(C = c \mid A_i = v)}{\sum_{k=1}^{o} P(A_i = v_k) P(C = c \mid A_i = v_k)}$$

$p(C|A_i)$ é estimado por meio de um estimador de densidades por kernels unidimensional:

$$\hat{P}(C = c \mid A_i = v_k) = \frac{1}{n_k h_k} \sum_{j=1}^{n_k} K\left(\frac{c - c_j}{h_k}\right)$$

onde o somatório é sobre todos os n_k exemplos com valor de atributo $A_i = v_k$. Uma estimativa de $p(A_i)$ é obtida simplesmente pela computação da proporção de exemplos com valor do atributo igual a v_k.

A priori $p(C)$ também é estimada com o uso de um estimador de densidades por kernels unidimensional:

$$\hat{P}(C = c) = \frac{1}{n h_C} \sum_{j=1}^{n} K\left(\frac{c - c_j}{h_C}\right)$$

As larguras dos kernels são escolhidas para minimizar a entropia-cruzada, cuja estimativa não enviesada pode ser obtida com validação cruzada leave-one-out. uma escolha comum para $K(.)$ é o kernel Gaussiano:

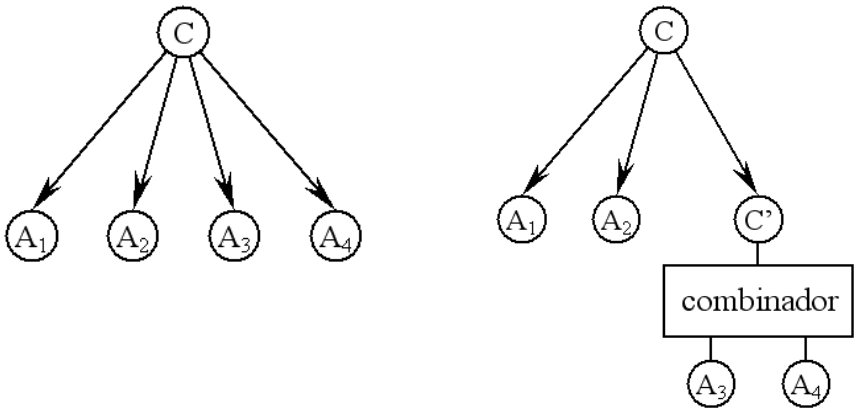
$$K(t) = \frac{e^{\left(\frac{-t^2}{2}\right)}}{\sqrt{2\pi}}$$

Com essas fórmulas pode-se computar uma estimativa de $p(A_i|C)$.

A predição ótima para um exemplo *e* em relação à probabilidade posterior $p(C|A = a)$ depende da função de perda. O NBR pode usar duas funções de perda: o erro quadrático e o erro absoluto. Em qualquer dos casos, o valor predito deve minimizar a perda esperada. O erro quadrático esperado é minimizado se o valor esperado de *c* (isto é, sua média) é predito. Se, por outro lado, o erro absoluto esperado deve ser minimizado, a predição ótima é a mediana.

3. Combinando Atributos

Uma vez que a principal razão da baixa performance do NBR é a suposição de independência, tentamos superar essa situação combinando atributos relacionados. Nossa abordagem é similar a descrita em [Pazzani 1996], na qual atributos nominais são combinados se isso gera aumento da acurácia preditiva. A principal diferença é que em [Pazzani 1996] os atributos originais são substituídos por um novo, cujos valores possíveis são as combinações de todos os valores possíveis para cada um deles (atributos numéricos são discretizados), enquanto que em nossa abordagem os atributos (nominais ou numéricos) são combinados por meio de algoritmos de regressão auxiliares. A princípio, esses algoritmos auxiliares, chamados *combinadores*, podem ser qualquer método de regressão. O valor da combinação é a predição do atributo objetivo fornecida por um combinador baseado apenas nos atributos sendo combinados, isto é, o combinador usa somente aqueles atributos para fazer uma predição. A Figura 1 mostra o algoritmo NBR e um modelo construído para combinar dois atributos: o combinador faz uma predição C' do atributo objetivo C baseado apenas nos atributos A_3 e A_4.



(a) (b)

Figura 1. Naive Bayes para Regressão (a) e o modelo conseguido pela combinação dos atributos A_3 e A_4 (b)

O algoritmo para selecionar todas as combinações que podem ser feitas inicialmente constrói um modelo tratando todos os atributos como condicionalmente independentes. Então ele considera a substituição de cada par de atributos por um novo atributo combinado. Para isso, é gerado um conjunto de instâncias formado apenas pelo par de atributos e pelo atributo objetivo. Um combinador usa esse conjunto de instâncias, chamadas *instâncias parciais*, para fazer as predições que serão os valores para o novo atributo combinado. A performance do modelo gerado por cada combinação é avaliada com validação cruzada, seguindo o procedimento descrito em [Kohavi e John 1995]. A combinação que obtém a melhor performance é mantida. Se nenhuma combinação resulta em melhoria, o modelo atual é selecionado. Caso

contrário, o procedimento continua, considerando combinações no modelo modificado. Esse procedimento e a complexidade do algoritmo usado como combinador determinam o custo computacional da abordagem.

O procedimento para treinar um combinador é descrito na Figura 2. Com o conjunto de treinamento original, são geradas instâncias parciais consistindo apenas dos atributos sendo combinados e o atributo objetivo. Com essas instâncias parciais o combinador é treinado e testado com validação cruzada. Suas predições (C') substituem os atributos sendo combinados no conjunto de treinamento original, gerando assim o conjunto de treinamento final usado para treinar o algoritmo NBR. O processo é análogo para mais de um combinador.

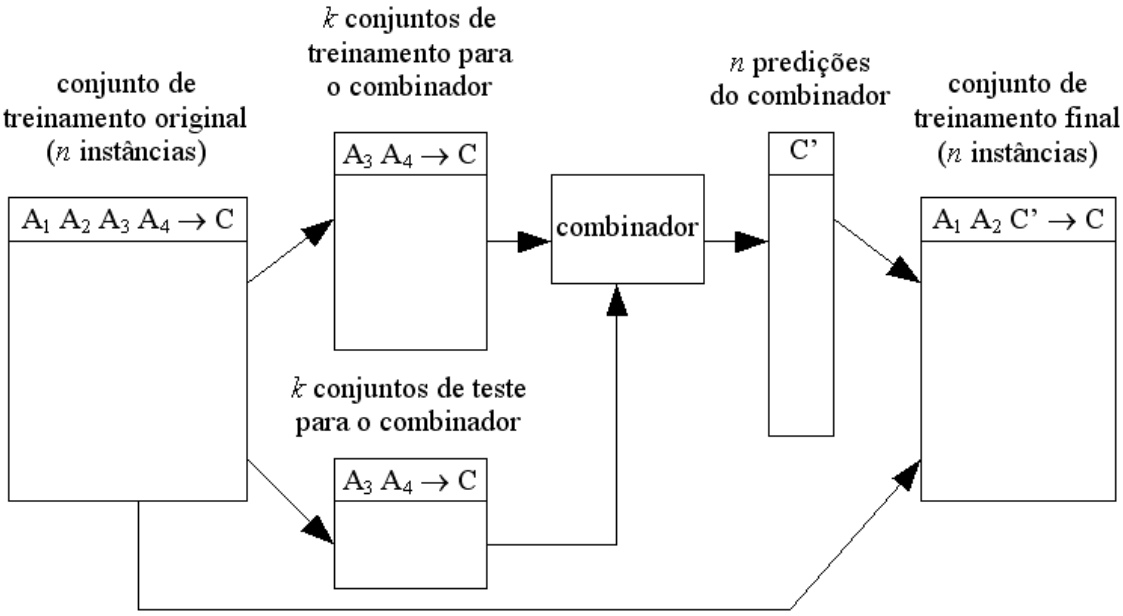


Figura 2. Gerando o conjunto final de instâncias de treinamento

O procedimento de teste é mostrado na Figura 3. Instâncias parciais são geradas e usadas com o combinador, cujas predições substituem os atributos combinados no conjunto de teste original, gerando assim o conjunto final usado para testar o algoritmo NBR. Novamente, o raciocínio pode ser facilmente estendido para o caso de mais de um combinador.

4. Avaliação Experimental

Uma extensa avaliação experimental foi realizada com o objetivo de analisar a performance da nova abordagem. Esta seção descreve os experimentos e apresenta seus resultados.

Uma vez que o NBR é um algoritmo lento (devido a suas características inerentes), escolhemos Decision Stumps [Iba e Langley 1992] para regressão (baseadas no erro médio quadrático) como combinadores, uma vez que elas são rápidas, permitindo a obtenção de resultados em um tempo de execução viável. A função de perda usada nos experimentos com o NBR foi o erro quadrático. A implementação do algoritmo NBR foi fornecida por Eibe Frank, um de seus autores. A implementação da Decision Stump foi obtida a partir do sistema WEKA [Witten e Frank 2005]. Todo o código fonte foi escrito em Java.

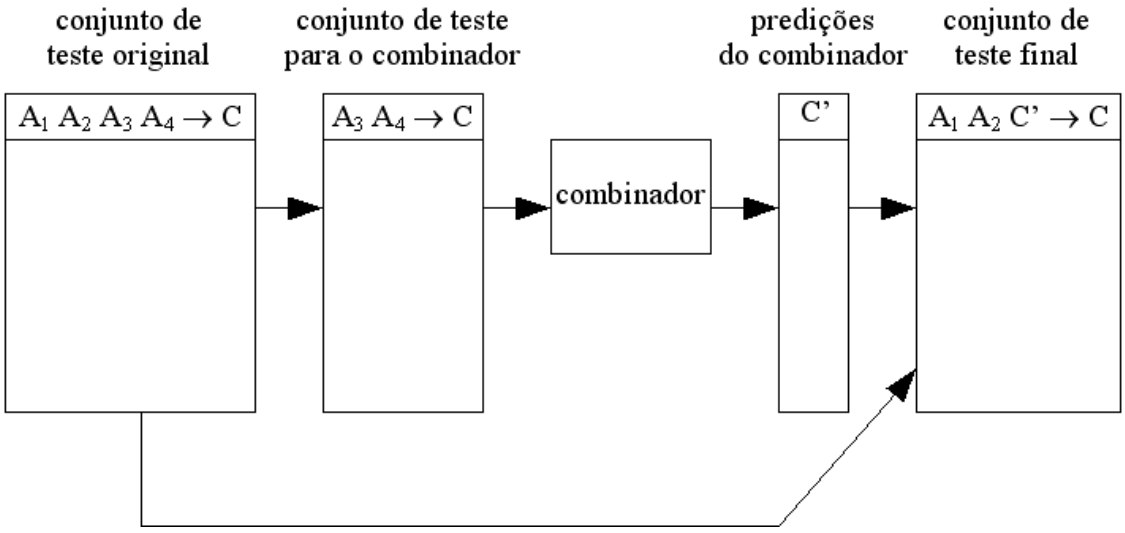


Figura 3. Gerando o conjunto final de instâncias de teste

Nos experimentos, 30 conjuntos de dados foram utilizados a fim de incluir muitos domínios e dificuldades. Os conjuntos de dados foram obtidos a partir do repositório de Aprendizado de Máquinas da Universidade da Califórnia [Blake e Merz 2005] e da página pessoal de Luís Torgo (http://www.niaad.liacc.up.pt/~ltorgo/). Nas primeiras três colunas da Tabela 1, apresentamos cada um dos conjuntos de dados usados, seu número de instâncias e seu número de atributos (excluindo o atributo objetivo). O método de teste usado nesta pesquisa foi o teste *t* emparelhado com validação cruzada de 10 partições [Dietterich 1998],[Mitchell 1997].

A quarta coluna da Tabela 1 mostra o número médio de atributos após o passo de combinação. Analisando os valores pode-se verificar que 2,7 combinações são aceitas em média.

As duas últimas colunas da Tabela 1 mostram para cada conjunto de dados o erro médio quadrático (emq) alcançado pelo algoritmo NBR e pelo modelo com combinadores. Onde está marcado com "**", a performance foi significativamente mais alta com nível de significância de 0,05, e onde está marcado com "*", a performance foi significativamente mais alta com nível de significância de 0,1.

A Tabela 2 apresenta um resumo dos resultados. O modelo construído com combinadores alcançou emq menor em mais da metade dos conjuntos de dados testados. Se forem considerados apenas os resultados estatisticamente significativos nos níveis citados acima, a nova abordagem apresentou melhor performance que o NBR em sete conjuntos de dados (cinco com nível de significância de 0,05 e dois com nível de significância de 0,1) enquanto que o algoritmo NBR venceu significativamente (com nível de significância de 0,1) em apenas um conjunto de dados. Em cinco conjuntos de dados houve empate. Três deles são conjuntos de dados com apenas dois atributos. Para estes conjuntos de dados não houve melhoria com o passo de combinação. Os outros dois empates foram verificados em conjuntos de dados para os quais ambos os modelos alcançaram erro praticamente zero.

Tabela 1. Resultados experimentais

Conjunto de Dados	Inst.	Ats.	Ats. comb.	NBR	NBR + Combinadores
autoMpg[1]	398	7	4,8	15,9308	16,6334
bank8FM[2]	8193	8	4,2	0,0049	0,0042**
baskball[1]	96	4	3,1	0,0090*	0,0103
bolts[1]	40	7	3,9	87,2708	62,7886
breastTumor[1]	286	9	4,2	104,2892	101,8824
cloud[1]	108	6	4,1	0,3373	0,3873
cpu_small[2]	8192	12	7,7	18,9957	18,4744
cpu[1]	209	7	4,8	11353,9600	5669,3247
delta_ailerons[2]	7129	5	3,4	0,0000	0,0000
delta_elevators[2]	9517	6	3,8	0,0000	0,0000
detroit[1]	13	13	7,0	3023,5271	874,9849
diabetes_numeric[2]	43	2	2,0	0,3564	0,3564
echoMonths[1]	130	9	3,6	152,6883	123,2886*
elusage[1]	55	2	2,0	155,0093	155,0093
fishcatch[1]	158	7	4,3	58501,5113	51005,7864
fruitfly[1]	125	4	1,4	297,4220	269,5326*
gascons[1]	27	4	2,4	235,5067	210,1115
hungarian[1]	294	13	9,2	0,2067	0,2528
longley[1]	16	6	3,2	795914,3021	1034660,3298
lowbwt[1]	189	9	4,6	305216,4736	262223,0827
mbagrade[1]	61	2	2,0	0,0960	0,0960
pharynx[1]	195	11	5,5	105828,4724	118369,0570
puma8NH[2]	8192	8	4,0	17,4688	16,1270**
pwLinear[1]	200	10	6,5	533,2886	493,7218**
quake[1]	2178	3	1,8	0,0381	0,0356**
sensory[1]	576	11	5,8	0,7730	0,7368
servo[1]	167	4	3,0	1,4082	0,7372**
strike[1]	625	6	3,5	301324,7891	317903,4610
veteran[1]	137	7	5,4	19574,5786	20325,9729
vineyard[1]	52	3	2,3	9,3684	14,7656

[1] Repositório de Aprendizado de Máquinas da UCI.

[2] Página pessoal de Luís Torgo.

Tabela 2. Resumo dos resultados

Medida	NBR	NBR + Combinadores
Número de vitórias	9	16
Número de vitórias significativas	1	7

5. Conclusões e Trabalhos Futuros

Apresentamos aqui um método para tentar superar a pobre performance alcançada pelo algoritmo Naive Bayes para Regressão devido à suposição de independência condicional entre atributos.

Experimentos demonstraram a eficiência e aplicabilidade da abordagem. Analisando os resultados, pôde-se verificar que o modelo construído pela combinação de atributos obteve melhor performance que o algoritmo NBR. Mesmo assim o modelo não obtém resultados para regressão tão impressionantes quanto o NBC obtém para classificação. Esta pesquisa constitui o primeiro passo na busca por um algoritmo baseado no NBR que seja competitivo quando comparado a outros algoritmos que representam o estado-da-arte para problemas de regressão.

Embora aqui este novo método tenha sido aplicado ao algoritmo NBR, a abordagem pode ser usada com outros algoritmos de aprendizado. Experimentos preliminares com algoritmos de classificação apresentaram resultados pobres, entretanto os resultados apresentados aqui se mostraram promissores para problemas de regressão. Considerando que Decision Stump é um algoritmo de aprendizado fraco, outros algoritmos podem também ser testados como combinadores.

Atualmente estamos realizando testes com regressores lineares como combinadores e com uma nova versão do algoritmo que também verifica se a remoção de conjuntos de atributos pode melhorar a performance.

Agradecimentos

Gostaríamos de agradecer a Eibe Frank por fornecer o código fonte do algoritmo NBR. Os autores são parcialmente financiados pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq.

Referências

Blake, C.L. e Merz, C.J. (2005) UCI Repository of Machine Learning Databases. Machine-readable data repository, Department of Information & Computer Science, University of California, Irvine. [http://www.ics.uci.edu/~mlearn/MLRepository.html]

Dietterich, T.G. (1998) Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, **10**, pp. 1895-1924.

Duda, R.O. e Hart, P.E. (1973) *Pattern Classification and Scene Analysis*. New York, NY: John Wiley & Sons.

Frank, E., Trigg, L., Holmes, G. e Witten, I.H. (2000) Naive Bayes for Regression. *Machine Learning*, **41**, pp. 5-25.

Friedman, N. e Goldszmidt, M. (1996) Building classifiers using Bayesian networks. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 1277-1284. Portland, OR: AAAI Press.

Iba, W. e Langley, P. (1992) Induction of one-level decision trees. In *Proceedings of the 9th International Conference on Machine Learning*, pp. 233-240. Aberdeen, Scotland: Morgan Kaufmann Publishers, Inc.

Keogh, E. e Pazzani, M. (2002) Learning the Structure of Augmented Bayesian Classifiers. *International Journal on Artificial Intelligence Tools*, **11** (4), pp. 587-601. World Scientific Publishing Company.

Kohavi, R. (1994) Feature subset selection as search with probabilistic estimates. In *Procedures of the AAAI Fall Symposium on Relevance*, pp. 122-126. New Orleans, LA: AAAI Press.

Kohavi, R. e John, G.H. (1995) Automatic parameter selection by minimizing estimated error. In A. Prieditis and S. Russell (eds.), *Proceedings of the 12th International Conference on Machine Learning*, pp. 304-312. Tahoe City, CA: Morgan Kaufmann.

Kononenko, I. (1991) Semi-naive Bayesian classifier. In *Proceedings of the Sixth European Working Session on Learning*, pp. 206-219. Porto, Portugal: Springer-Verlag.

Mitchell, T.M. (1997) *Machine Learning*. New York, NY: McGraw-Hill.

Pazzani, M. (1996) Searching for dependencies in Bayesian classifiers. In D. Fisher and H.J. Lenz (eds.), *Learning from data: Artificial intelligence and statistics V*, pp. 239-248. New York, NY: Springer-Verlag.

Wang, Z., Webb, G.I. e Zheng, F. (2003) Adjusting Dependence Relations for Semi-Lazy TAN Classifiers. In *Proceedings of the 16th Australian Conference on Artificial Intelligence*, pp. 453-465. Perth, Australia: Springer.

Wang, Z., Webb, G.I. e Zheng, F. (2004) Selective Augmented Bayesian Network Classifiers Based on Rough Set Theory. In *Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 319-328. Sydney, Australia: Springer.

Webb, G. I., Boughton, J. e Wang, Z. (2005) Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, **58** (1), pp. 5-24.

Witten, I.H. e Frank, E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition. San Francisco, CA: Morgan Kaufmann.

Zhang, H. e Ling, C.X. (2001) An Improved Learning Algorithm for Augmented Naive Bayes. In *Proceeding of the 5th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 581-586. Hong Kong, China: Springer.

Zheng, Z. e Webb, G.I. (2000) Lazy Learning of Bayesian Rules. *Machine Learning*, **41** (1), pp. 53-84.