UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUÍS FELIPE DA SILVA GUARIENTO

UM ESTUDO DE TÉCNICAS DE MINERAÇÃO DE DADOS PARA CLASSIFICAÇÃO COM APLICAÇÃO EM CLASSIFICAÇÃO DE FONTES DE DADOS WEB

Niterói
2011

LUÍS FELIPE DA SILVA GUARIENTO

UM ESTUDO DE TÉCNICAS DE MINERAÇÃO DE DADOS PARA CLASSIFICAÇÃO COM APLICAÇÃO EM CLASSIFICAÇÃO DE FONTES DE DADOS WEB

Monografia apresentada ao Curso Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Aprovada em Julho de 2011

BANCA EXAMINADORA

_____

Orientador: Prof. Luiz André Portes Paes Leme
UFF

_____

Prof º. Alexandre Plastino de Carvalho
UFF

_____

Prof ª. Vanessa Braganholo Murta
UFRGS

Niterói
2011

LUÍS FELIPE DA SILVA GUARIENTO

UM ESTUDO DE TÉCNICAS DE MINERAÇÃO DE DADOS PARA CLASSIFICAÇÃO COM APLICAÇÃO EM CLASSIFICAÇÃO DE FONTES DE DADOS WEB

Monografia apresentada ao Curso Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Luiz André Portes Paes Leme

Niterói

2011

Primeiramente agradeço a Deus por ter tido a oportunidade de desenvolver esse trabalho. Agradeço a meu orientador, a familiares e amigos que sempre entenderam quando eu não podia lhes dar a devida atenção e a todos aqueles que, direta ou indiretamente contribuíram para que eu me aprimorasse mais como pessoa e como profissional

SUMÁRIO

Lista de Equações

Lista de Ilustrações

Lista de Tabelas

ABSTRACT

In this work, we intend to explore data mining techniques to characterize data sources. This requires a detailed study of classification techniques known in the literature and a bibliographic overview of the methodology they use. They should also be characterized methods of evaluating the performance of different classifiers as well as ways to avoid ratings records so wrong. Finally, we analyze an experiment categorization of data sources using the technique of Support Vector Machine classification in order to validate its efficiency in the categorization of sites according with variation of specific parameters for the experiment

Resumo

Nesse trabalho, pretende-se explorar técnicas de mineração de dados para caracterização de fontes de dados. Para isso é necessário um estudo minucioso de técnicas de classificação conhecidas na literatura bem como um apanhado bibliográfico das metodologias por elas usadas. Devem-se caracterizar também métodos de avaliação de desempenho de classificadores distintos bem como maneiras de se evitar classificações de registros de maneira equivocadas. Por fim, analisa-se um experimento de categorização de fontes de dados pela técnica de classificação de Suporte Vector Machine de modo a validar sua eficiência na categorização de sites segundo variação de parâmetros específicos para o experimento.

Palavras-chave: classificação, desempenho, fontes de dados

1 Introdução

1.1 Motivação

A *Deep Web* cresce a cada dia, Madhavan et al. [2] estimam que devem existir cerca de 25 milhões de fontes de dados escondidas na Web. Entretanto, os esforços para a *indexação* desses dados ainda se mostram insuficientes. Problema maior existe na *integração dos dados,* conforme relatado por [18] e [25]. A grande quantidade de fontes, a disponibilidade imprevisível e as constantes diferenças entre os esquemas de dados requerem técnicas automáticas e dinâmicas de integração. Diversos esforços vêm sendo feitos nessa direção, no entanto, atualmente grande parte das fontes é vista como se fossem conjuntos de dados disjuntos.

Mediadores de consultas são aplicações que promovem a integração de fontes de dados. Eles devem ser capazes de identificar novas fontes, eliminar fontes obsoletas e "enxergar" o conjunto completo dos dados como se fossem instâncias de um único esquema integrado de dados. Pode-se dividir o problema de mediação de consulta em cinco subproblemas, conforme Madhavan et al [2]:

i. Identificar novas fontes de dados e excluir fontes obsoletas: *busca de fontes*
ii. Extrair dados de documentos não estruturados: *extração de dados*
iii. Alinhar esquemas de dados: *alinhamento de esquemas*
iv. Gerar esquema de dados mediado: *construção de esquema global*
v. Distribuir consultas de usuários pelas diversas fontes: *mediação de consulta*

O primeiro subproblema diz respeito à disponibilidade imprevisível das fontes. Mediadores não têm controle sobre o ambiente mediado. As fontes podem entrar e sair do ambiente à vontade. Portanto, a lista de participantes deve ser constantemente atualizada.

O segundo subproblema trata da captura de dados estruturados a partir de documentos não estruturados como, por exemplo, páginas HTML.

O terceiro e quarto problemas dizem respeito à construção de um esquema de dados global (subproblema iv), chamado esquema mediado, a partir dos vários esquemas exportados pelas fontes de dados. Nesse processo, conceitos equivalentes em diferentes esquemas devem ser unificados (subproblema iii).

Finalmente, as consultas dos usuários devem ser interceptadas, traduzidas e distribuídas para cada fonte de dados. Os resultados parciais de cada fonte são, então, integrados e devolvidos para os usuários (subproblema v). Nessa última etapa torna-se de vital importância a otimização da distribuição das consultas entre as várias fontes evitando-se encaminhar consultas a fontes de dados que provavelmente não retornariam nenhuma resposta. Isso pode ser alcançado classificando-se fontes de dados e consultas por assunto de modo que o encaminhamento de consultas seria feito somente para fontes de dados que tratam do mesmo domínio de informação que a consulta.

Este trabalho apresenta um estudo de técnicas de mineração de dados aplicadas ao problema de classificação com o objetivo de desenvolver um classificador de fontes de dados. Apresenta também os resultado obtidos por um experimento utilizando uma das técnicas.

Este trabalho está estruturado da seguinte forma: no capítulo 2 são apresentadas técnicas de mineração de dados aplicadas ao problema de classificação. O capítulo 3 descreve algumas funções de similaridade entre objetos usadas por alguns classificadores (como os classificadores baseados nos k-vizinhos mais próximos) bem como os métodos de discretização de atributos contínuos, que servem para melhorar o desempenho de alguns classificadores (como árvore de indução e Naive Bayes). O capítulo 4 descreve Overffiting de modelo e métodos para avaliar o desempenho de classificadores. O capítulo 5 descreve o experimento realizado para classificação de fontes de dados e finalmente o capítulo 6 descreve as conclusões sobre o estudo realizado.

2 Técnicas de classificação

2.1 Introdução

O problema de classificação refere-se à criação de um processo automático de inferência da classe a qual pertence cada um dos registros de um conjunto com base na avaliação de seus atributos. Para isso faz-se uma avaliação dos atributos de registros preliminares, normalmente conhecidos por registros de treinamento. O algoritmo deve formular um modelo adequado capaz de inferir um rótulo de classe apropriado a cada um dos registros de teste cuja classe é desconhecida com uma boa porcentagem de registros classificados corretamente segundo a base de dados utilizada. Os registros de teste são registros de mesma natureza dos registros usados para treinamento, mas que não participaram da geração do modelo. A ilustração a seguir adaptada de [3] ilustra essa idéia:

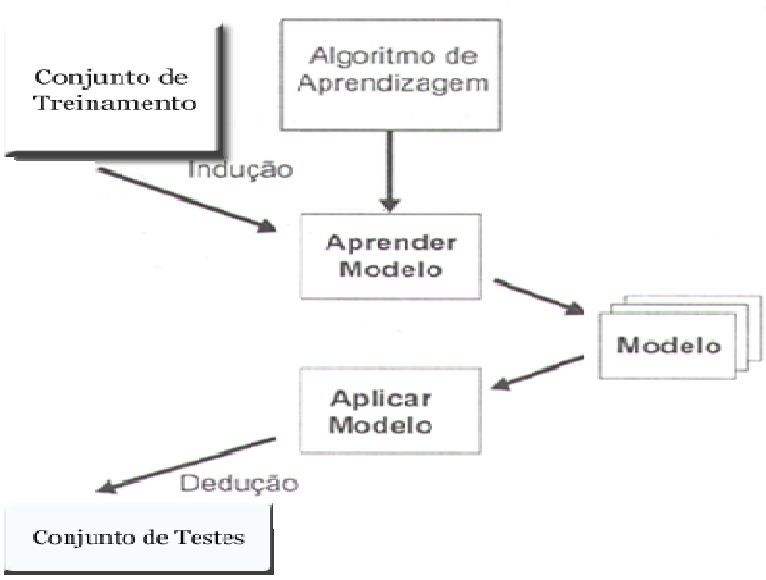


Ilustração 1: Resolução de um problema de classificação

Pelo que foi exposto acima, pode-se então definir classificação como uma tarefa de aprender uma função alvo *f* que mapeie um conjunto de atributos *x* para uma das categorias já definidas *y*.

Os métodos automáticos de classificação aqui descritos podem ser categorizados por alguns paradigmas segundo [11]:

i. *Classificadores Simbólicos:* procuram representar o conhecimento através de símbolos ou expressões lógicas. São representantes desses classificadores as árvores de indução (seção 2.1) e o classificador baseado em regras (seção 2.2).

ii. *Classificadores estatísticos*: o foco é posto nos atributos dos objetos usados pelos classificadores, em que todos os atributos têm valor contínuo ou ordinal. Também com respeito a esse aspecto são obtidos parâmetros da análise dos dados (como o desenvolvimento de um limite linear que separa os objetos classificados). Os representantes desse grupo são os classificadores de bayes (seção 2.5) e o support vector machine (seção 2.6)

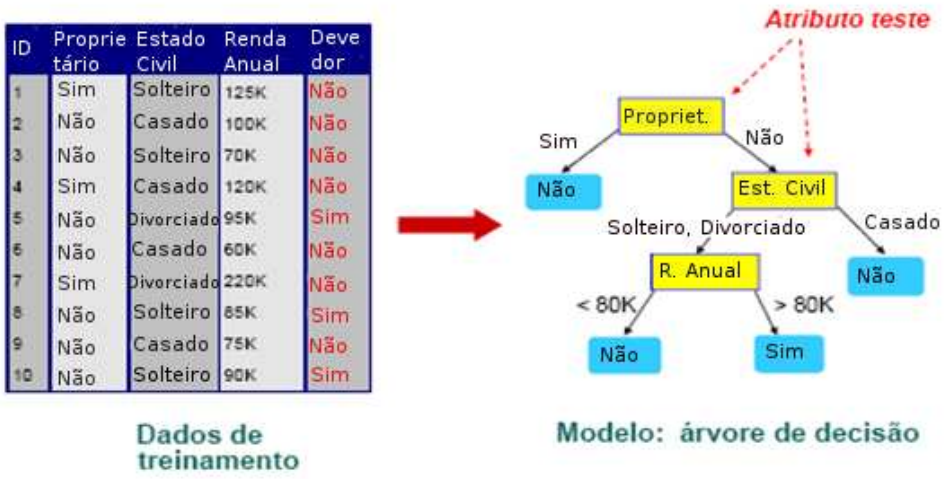
iii. *Classificadores baseados em instância*: baseiam-se em exemplos de classificação anteriores que determinam que para um novo exemplo ser classificado, esse exemplo deve ser comparado aos outros e medido assim um grau de semelhança relativamente a objetos cuja classificação é conhecida. O método dos k-vizinhos mais próximos de um exemplo de teste (seção 2.4) é um representante desse grupo.

iv. *Classificadores conexionistas*: criam um modelo de classificação que tenta simular as conexões neurais do sistema nervoso humano. Esses métodos de classificação baseiam-se no paradigma de redes neurais.

As próximas seções apresentam as seguintes técnicas de classificação: 2.2 - Indução da árvore de decisão, 2.3 - Indução de Regras, 2.4 - k-vizinhos mais próximos, 2.5 - Bayes e 2.6 - Support Vector Machine (SVM).

2.2 Indução da árvore de decisão

Uma árvore de decisão é uma estrutura que representa o conhecimento extraído do conjunto de treinamento sobre um determinado assunto sob a forma de uma árvore, conforme descrito por J. R. Quinlan (1986) [21]. Os nós internos representam testes sobre os valores dos atributos com cada nó representando um atributo. Cada um dos resultados do teste leva a uma subárvore, sendo a estrutura da subárvore a mesma da árvore. Cada um dos nodos folha representa uma categoria ou rótulo inferido a partir da avaliação dos atributos de treinamento. Podem-se ter várias árvores para um mesmo conjunto de treinamento, com testes sobre atributos feitos em ordem diferente. A melhor escolha depende do grau de pureza produzido por cada um dos particionamentos dos registros de treinamento originais, como será discutido mais adiante, e do erro em classificar objetos ainda não vistos. Na ilustração 2, adaptada de [31] e dada a seguir, encontram-se duas árvores produzidas pela avaliação do mesmo conjunto de treinamento, podendo-se visualizar os conceitos aqui expostos.



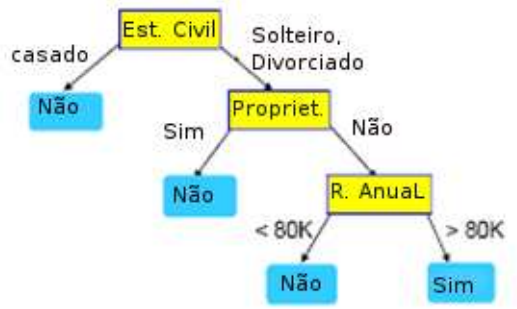


Ilustração 2: Duas árvores diferentes produzidas pela avaliação do mesmo conjunto de treinamento

O processo de classificação de um registro com a árvore formada é definido, ainda segundo J. R. Quinlan (1986) [21], pela avaliação dos atributos do registro de teste segundo a árvore montada e descendo um nível na árvore segundo o valor de cada atributo do registro de teste até que se chegue a um nodo folha. O nodo folha conterá o rótulo ou categoria do registro de teste dado pelo modelo, segundo a ilustração a seguir.

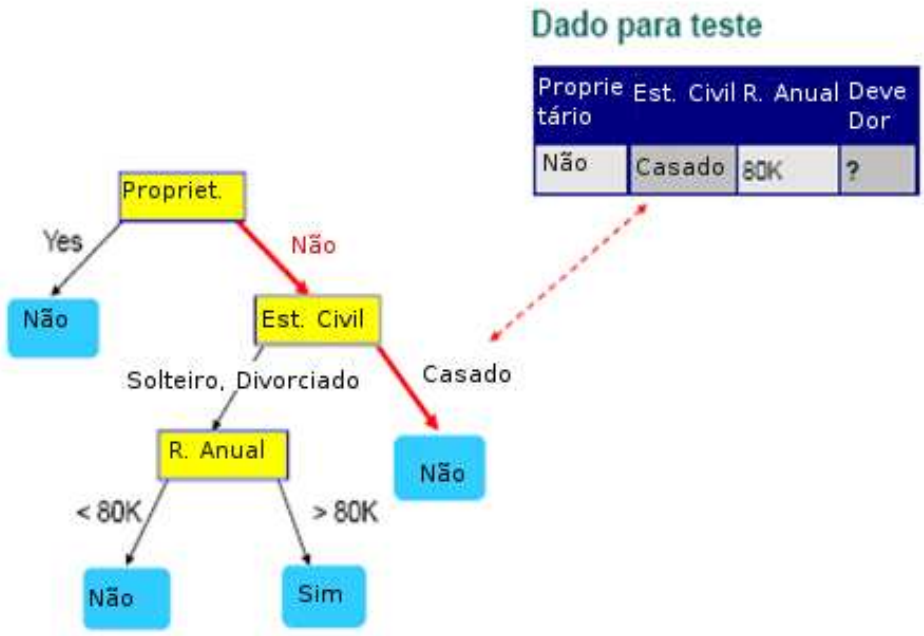


Ilustração 3: como rotular um registro de teste adaptada de [31]

Como encontrar uma árvore ótima para o problema de classificação é inviável (o espaço de pesquisa pode crescer exponencialmente), muitas métricas são utilizadas seguindo uma decisão local sobre quais atributos usar no momento para particionar o conjunto de registros de treinamento (isso será visto na seção seguinte).

Por enquanto, discrimina-se um algotitmo que é a base para muitos algoritmos de árvore de decisão existentes (como ID3, C4.5 e CART [3] e [11]) e é chamado de algoritmo de Hunt, Earl B. Hunt (1962) [21].

2.2.1 Algoritmo de Hunt

O crescimento da árvore é realizado de forma recursiva, pelo particionamento dos registros de treino em subconjuntos mais puros. Suponha-se como D_t o conjunto dos registro de treinamento associados ao nodo t e que $y = \{y_1, y_2,...,y_n\}$ sejam os rótulos das classes desses conjuntos de treinamento. Esse algoritmo é definido recursivamente como a seguir:

1) Caso todos os registros de D_t pertençam à mesma classe y_t, t é um nó folha rotulado como y_t (condição de parada do algoritmo)

2) Caso D_t contenha registros que pertençam a mais de uma classe, uma condição de teste de atributo é selecionada para que os registros sejam particionados em subconjuntos menores. Um nó filho será criado para cada resultado da condição de teste e subconjuntos do conjunto original D_t serão associados aos filhos baseados nos resultados do atributo de teste. O algoritmo é aplicado de forma recursiva a partir daqui a cada filho criado.

A ilustração 4 dada a seguir ilustra esse processo:

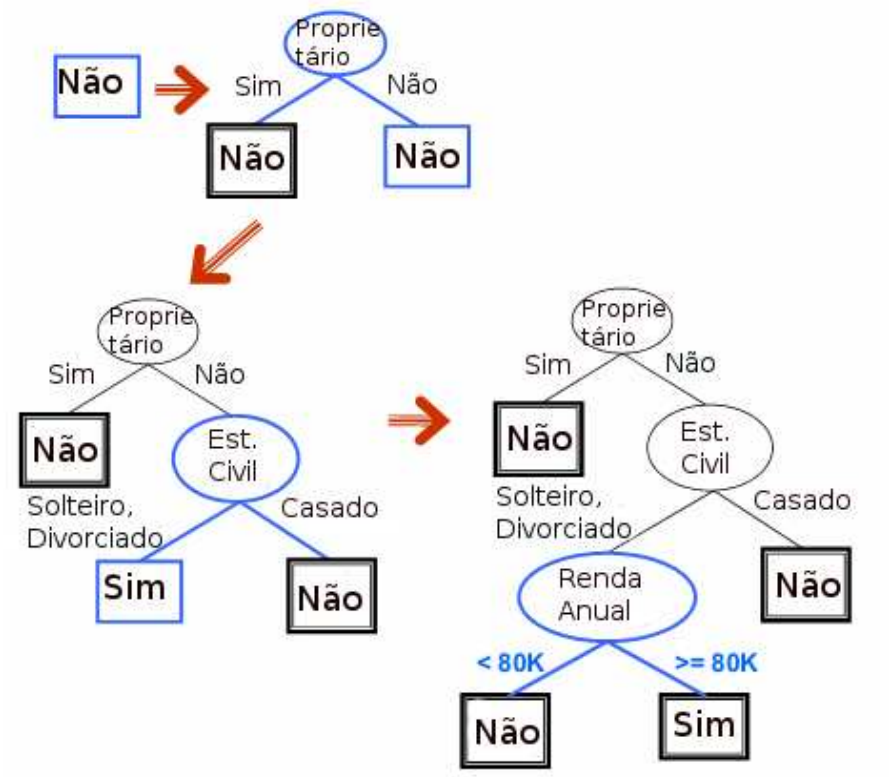


Ilustração 4: Uso do algoritmo de Hunt para montagem da árvore usando os dados de treinamento da ilustração 2, adaptada de [31]

Sendo o algoritmo definido acima, algumas questões de projeto precisam ser definidas. Tais questões serão analisadas mais a fundo nas próximas sessões.

Uma das questões mais pertinentes sobre o tratamento dos registros de treinamento diz respeito à seleção dos atributos a serem analisados a cada passo recursivo. Os testes realizados sobre os atributos serão tratados com detalhes a seguir.

Outra questão chave é o quanto expandir a árvore a partir dos registros de treinamento. Uma estratégia é continuar expandindo até que todos os registros de treinamento sejam do mesmo tipo ou tenham valores de atributos iguais. Tal estratégia pode fazer com que o modelo de classificação se adeque perfeitamente aos dados de treinamento, e generalize mal relativo a atribuição de classes a objetos não vistos previamente . Outros critérios podem ser adotados para evitar tal situação, como a pré-poda [3], que faz com que o algoritmo pare de expandir a árvore antes de se ter uma árvore cujos nodos testem todos os registros de treinamento e adeque as soluções obtidas pela árvore aos dados de treinamento e também a pós-poda [3], que analisa os registros após a formação da árvore completa. Tanto a pré-poda quanto a pós-poda serão discutidas no final dessa seção.

2.2.2 Particionamento de registros

O particionamento de registros segue uma lógica baseada na medida de impureza dos registros, havendo muitas métricas para se medir a impureza de um conjunto de registros a ser particionado. Algumas das quais estão explicitadas abaixo [3].

$$\sum_{i=0}^{c-1} p\left(\frac{i}{t}\right) \log_2 p\left(\frac{i}{t}\right) \qquad (1)$$

$$1 - \sum_{i=0}^{c-1} \left[p\left(\frac{i}{t}\right) \right]^2 \qquad (2)$$

$$erro(t) = 1 - \max_{i=0}^{c-1} p\left(\frac{i}{t}\right) \qquad (3)$$

Nas equações anteriores, p(i/t) é a parcela de registros pertencentes a determinada classe i associada ao nodo t, sendo as classes de registros rotuladas de 0 até c-1. A equação 1 é denominada entropia de um nodo t, a equação 2 refere-se ao índice gini de um nodo t e a equação 3 é o erro de classificação de um nodo t.

O melhor atributo para dividir um conjunto de registros é aquele que produzir a divisão mais "desigual de registros", alocando registros de classes diferentes em nodos diferentes. O exemplo a seguir, retirado de [3] ilustra essa idéia:

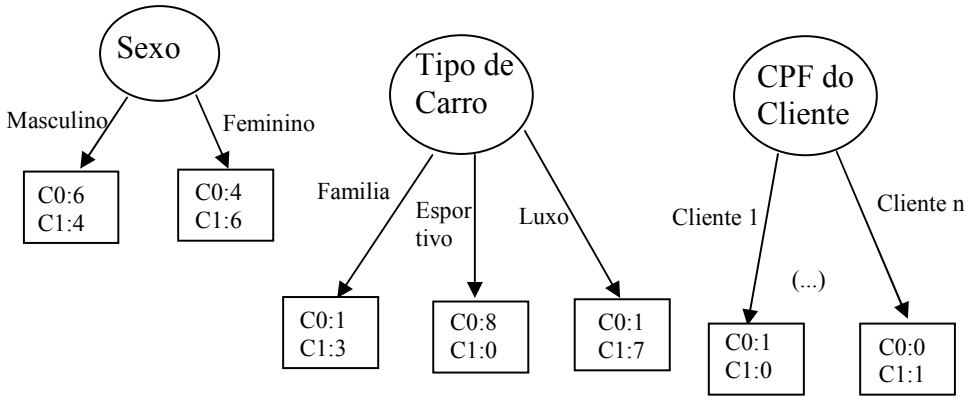


Ilustração 5: várias formas de dividir conjuntos de registros segundo diferentes atributos

Pela equação do índice gini, pode-se perceber que a média ponderada do gini para o atributo "Sexo" é (10/20) * 0,48 + (10/20) * 0,48 = 0,48.

Usando a média ponderada para o segundo atributo "Tipo de Carro" obtêm-se (4/20)*0,375 + (8/20)*0 + (8/20)*0,219 = 0,075 + 0,0876 = 0,1626.

Pelas médias obtidas dos dois ginis calculados, o atributo que produz a melhor divisão desconsiderando a análise do último atributo "CPF", é o tipo de carro (média do gini para os nodos é de 0,1626).

No entanto, o uso do gini como escolha primordial para o atributo de teste pode se mostrar insatisfatório considerando a análise do CPF de um Cliente, visto que o CPF é único para cada cliente, cada particionamento conterá um único registro. Conclui-se que a média do índice gini pode ser bem menor do que para o "Tipo de Carro". Assim, nem sempre um atributo que produz muitas partições é confiável, pois a quantidade de registros em cada

partição pode ser pequena demais. Nesse caso, deve-se penalizar um atributo que produz muitas partições seja restringindo as partições produzidas por um nodo a apenas dois atributos (o algoritmo de árvore de decisão CART utiliza essa abordagem), ou então, considerar as várias partições com penalidade (utilizado pelo algoritmo C4.5), que consiste na taxa de ganho dada a seguir:

$$TaxaGanho = \frac{\Delta \text{Info}}{\text{InfoDiv}} \tag{4}$$

A informação de divisão é dada da seguinte maneira:

$$InfoDiv = -\sum_{i=1}^{k} p(v_i) \log_2 p(v_i) \tag{5}$$

Na equação 5, k simboliza o total de divisões do atributo e $p(v_i)$ é o número de registros pertencentes ao i-ésimo valor do atributo. Isso sugere que, se um atributo produzir grande número de divisões, infodiv será grande e a taxa de ganho diminuirá. Já o ganho de informação (equação 6) é definido pela diferença entre o grau de impureza do nodo pai (antes da divisão) com o de seus filhos (após a divisão).

O que essa equação nos indica é que maximizar o ganho de informação equivale a minimizar as médias ponderadas de impureza dos nodos filhos em relação ao nodo pai.

A equação abaixo resume o ganho de informação (ΔInfo):

$$\Delta Info = I(pai) - \sum_{j=1}^{k} \frac{N(v_j)}{n} I(v_j) \tag{6}$$

Na equação 6, I(pai) é a medida de impureza do nodo pai, $I(v_j)$ é a medida calculada para a impureza do nodo filho v_j, o que pode ser dado por alguma das equações (1), (2) ou (3). $N(v_j)$ simboliza o número de registros associados ao nodo filho v_j e N é o número total de registros testados pelo nodo pai. Para o algoritmo C4.5, usa-se a entropia como métrica de impureza.

Comparando os valores da taxa de ganho de diversos atributos, pode-se escolher aquele que tem a maior taxa de ganho para a divisão dos registros a cada etapa de formação da árvore de decisão.

2.2.3 Pré-Poda:

Na pré-poda o algoritmo de árvore de decisão é parado antes de produzir uma árvore que se encaixe perfeitamente nos registros de treinamento. Tal condição de parada pode ser avaliada por algum dos critérios já vistos, como o índice gini(equação 2) ou a taxa de ganho (equação 4). Compara-se, por exemplo a diferença entre as entropias dos nodos filhos e a entropia do nodo pai. O algoritmo pára quando não for mais possível uma diferença de entropia significativa ou quando o número total de registros para um mesmo nodo também apresenta um valor abaixo do total de seu respectivo nodo pai, por exemplo. Uma das vantagens da parada cedo é o fato de evitar o aumento da complexidade da árvore de decisão. Uma árvore muito complexa pode estar sujeita a overfffiting (seção 4.1). O melhor seria adiar a parada cedo para alguns nodos descendentes posteriores, pois, se por alguma métrica utilizada os nós descendentes apresentarem desempenho também abaixo do esperado, a poda do ramo é necessária.

2.2.4 Pós-poda:

A pós-poda é feita após o algoritmo de obtenção da árvore de decisão ter sido completamente executado e é um complemento a pré-poda. A pós poda realiza então uma avaliação bottom-up das sub-árvores da árvore original considerando alguma medida de desempenho como o ganho de informação de modo a identificar possíveis ramos candidatos a poda. Duas medidas podem ser tomadas. Na primeira a subárvore pode ser substituída por um nodo folha, atribuindo-lhe a classe da maioria dos registros da sub-árvore. Ou então a subárvore pode ser substituída por outra usada com maior freqüência. Isso é feito de modo a replicar partes da árvore com testes sobre atributos que mais caracterizam os dados e assim aumentar o poder de expressão da árvore se o ganho observado com a nova sub-árvore for maior do que com a antiga.

A pós-poda condiciona melhores resultados do que a pré-poda, pois toma suas decisões baseada em uma árvore com erro de treinamento baixo e já completamente desenvolvida. No entanto, os cálculos realizados antes da poda da árvore podem ser

desperdiçados, perdidos. Realizar a pós poda pode ser computacionalmente custoso, pois deve-se analisar cada subárvore da árvore original.

2.3 Indução de regras

Um classificador baseado em Indução de Regras formula regras em torno da categorização dos registros de treinamento de modo que o antecedente de cada regra seja uma sentença condicional (composta de conjunções de cláusulas, sendo cada cláusula um teste sobre algum atributo definido no conjunto de treinamento) e o lado direito, denominado conseqüente, define a categoria da classe. Os principais conceitos aplicados a um classificador de regras podem ser vistos em [3], [27] e [28]. Tais conceitos serão descritos a seguir.

A estrutura básica de uma regra em um classificador de regras pode ser definida pela equação 7.

$$R_i : (S) \rightarrow Y_i \qquad (7)$$

Sendo R_i a regra, S é a sentença condicional e Y_i é a classe da regra dada pela avaliação da sentença. O índice i denota o número da regra. Assim, a sentença condicional é dada como abaixo:

$$S = (<atributo>_1 <Operação> <Valor>_1) and .. and (<atributo>_k <Operação> <Valor>_k) \quad (8)$$

Na equação 8, *atributo* corresponde a algum valor de atributo dos registros de teste propriamente dito e a *operação* corresponde a algum operador relacional dentre {=, <,>,≤,≥,≠}. Em uma sentença pode-se ter k testes de atributos onde k ≤ n, correspondendo n ao total de atributos distintos dos registros.

As regras de classificação também podem ser obtidas a partir de uma árvore de indução. A obtenção de regras a partir de uma árvore de indução foi formalmente definida por Quinlan (1985) [21]. Cada caminho na árvore da raiz até as folhas representando uma regra, o conseqüente da regra é dado pela folha em questão e cada um dos nós internos da árvore juntamente com sua aresta representando uma cláusula. Assim para um mesmo registro deve haver apenas um caminho da raiz às folhas. Considerando os registros de treinamento dado a seguir para a classificação de bons e maus pagadores.

Tabela 1: Registros de Treinamento para um problema de crédito bancário, adaptado de [31]

ID	Proprietário	Estado Civil	Renda Anual	Devedor
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim

A ilustração 3 da seção 2.2 mostra uma das árvores de indução obtidas a partir da tabela 1. Tomando-se essa árvore, podem-se construir as regras de indução como explicado. As regras obtidas a partir da árvore da ilustração estão ilustradas abaixo:

$$R_1 : (\text{Pr}\, oprietário = Sim) \rightarrow Não$$

$$R_2 : (\text{Pr}\, oprietário = Não)\text{^}(Est.Civil = \{Soleteiro, Casado\})\text{^}(R.Anual < 80k) \rightarrow Não$$

$$R_3 : (\text{Pr}\, oprietário = Não)\text{^}(Est.Civil = \{Solteiro, Divorciado\})\text{^}(R.Anual > 80k) \rightarrow Sim$$

$$R_4 : (\text{Pr}\, oprietário\ = Não)\text{^}(Est.Civil\ = Casado\ ) \rightarrow Não$$

Na verdade a extração de regras de indução através de uma árvore é dito um método indireto de extração de regras. O método direto será visto mais adiante.

Levando em conta as regras acima, a maneira usual de medir sua qualidade é através da cobertura e precisão da regra. A cobertura é definida como a fração entre o total de registros cujos valores dos atributos satisfazem ao antecedente da regra R_i (S) e a totalidade dos registros (T). Como pode ser visto abaixo:

$$Cobertura(R_i) = \frac{S}{T} \qquad (9)$$

A cobertura da regra R_1 segundo essa definição é 0,3.

Um fator também importante para definir a qualidade da regra é sua precisão. A precisão pode ser definida como a fração do número de registros que satisfazem ao mesmo tempo o antecedente e o conseqüente de uma regra (S∩y) e o valor de S definido acima. A precisão é definida abaixo:

$$\text{Precisão}(R_i) = \frac{(S \cap y)}{S} \qquad (10)$$

A precisão da regra R_1 é então 100%. Considera-se agora o acréscimo da seguinte regra ao conjunto de regras definidas anteriormente:

$$R_5 : (\text{Proprietário} = Não) \wedge (Est.Civil = \{Solteiro, Casado\}) \rightarrow Sim$$

Calculando a precisão dessa regra obtem-se: Precisão (R_5)=2/6=0, 3333.

Isso quer dizer que essa regra classifica corretamente apenas 2 registros de 6 satisfeitos pelo antecedente da regra. Tem baixa precisão, no entanto a cobertura da regra é maior do que para a regra R_1: cobertura (R_5) = 6/10 = 0,6. O dobro de cobertura de R_1.

Já a regra R_1 tem precisão máxima, mas baixa cobertura.

Dessa discussão conclui-se que considerar os fatores de cobertura e precisão isoladamente pode não produzir resultados aceitáveis. Uma nova métrica de avaliação da qualidade de regras considerando conjuntamente os dois fatores de avaliação descritos acima deve ser considerada. Tais métricas são chamadas métricas estatísticas e probabilísticas [3]. Abaixo são definidas quatro delas:

$$R = \sum_{i=1}^{k} f_i \log\left(\frac{f_i}{e_i}\right) \qquad (11) \qquad avaliacão - m = \frac{f_+ + kp}{n + k} \qquad (12)$$

$$laplace = \frac{f_+ + 1}{n + k} \qquad (13) \qquad Foil = \log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_2}{p_2 + n_2} \qquad (14)$$

Na equação 11 f_i denota a frequência observada dos exemplos da classe i cobertos por R, k é o número de classes do modelo e e_i é a freqüência aleatória para a classe i. Um valor grande para R significa que o total previsões corretas realizadas para a regra R é maior do que a previsão aleatória. Considerando as regras R_1 e R_5:

$R(R_1)= [3 \times \log_2(3/2,1) + 0 \times \log_2(0/0,9)] = 51,46\%$

$R(R_5)= [4 \times \log_2(4/4,2) + 2 \times \log_2(2/1,8)] = 2,26\%$

A cobertura da regra R_1 é baixa, no entanto ela á preferível em relação a regra R_5, porque tem uma precisão de 100% dado que R_5 classifica corretamente somente 2 dos 6 registros de treinamento submetidos a ela.

A equação 12 é dita avaliação-m e a equação 13 é dita métrica de laplace. f_+ simboliza o número de exemplos corretamente classificados pela regra, k é o número total de classes, p é a freqüência da classe que a regra rotula e n é o número total de exemplos cobertos pela regra.

A equação 14 é chamada ganho de informação de Foil. Pode-se verificar se a adição de mais algum teste de atributo em uma regra produz maior ganho na classificação de exemplos positivos (classificados corretamente pela regra) do que a regra em sua condição anterior (o algoritmo de Ripper que será visto adiante usa a métrica de Foil na poda da regra obtida). Na equação p_0 indica a quantidade de exemplos positivos cobertos pela regra antes da adição de mais um teste de atributo na regra, n_0 indica os exemplos negativos cobertos anteriormente pela regra original, p_1 simboliza os exemplos positivos cobertos pela regra após sua expansão e n_1 a quantidade de exemplos negativos da regra aumentada. Os exemplos negativos são aqueles classificados incorretamente pela regra.

2.3.1 Métodos diretos de extração de regras

Outras formas de se obter o conjunto de regras são pelos chamados métodos diretos. Abaixo se descrevem as duas principais abordagens de métodos diretos: por regras ordenadas ou por regras não ordenadas. Basicamente devem-se resolver os problemas de qualidade das regras, como comentado acima e da determinação da classe de um registro por mais de uma regra. A definição dos métodos diretos de extração de regras pode ser vista em [3] e [28].

2.3.1.1 Regras ordenadas

A estratégia de regras ordenadas considera a ordenação decrescente das regras segundo algum critério estabelecido, como a ordem de geração das regras, cobertura, precisão das mesmas, Laplace, avaliação-m (vistos anteriormente), etc. Quando se classifica um registro de teste, ele é submetido à classificação pela regra de mais alta prioridade. Caso o registro sendo testado não corresponda ao teste por tal regra, ele é submetido a segunda regra de maior prioridade e assim sucessivamente. Deve ser definido então um esquema de ordenação de regras. A ordenação pode ser construída baseada em regras ou baseada em classes. Na ordenação baseada em regras, ordenam-se as regras de maneira individual por alguma métrica de qualidade dentre as já citadas. Uma desvantagem para esse esquema é o fato de que regras de menor prioridade vêm sempre abaixo das demais e pressupõem a negação dos atributos testados nas regras anteriores. Como um exemplo, pode-se tomar a ordenação das regras R_1 até a regra R_5 segundo a cobertura das mesmas. Os registros de teste seriam testados pelas regras nessa ordem: R_5, R_4, R_3, R_2, R_1. No caso as coberturas de R_4 até R_1 são idênticas (0,3), mas supondo-se os testes serem realizados na ordem exposta. A interpretação da regra R_5, que testa alguns atributos que também são testados pelas regras cuja avaliação da cobertura é menor, é que todos os registros cujos atributos tenham os valores dados por R_5 serão classificados primeiro (e alguns de forma incorreta), não importando se as regras posteriores comparem mais atributos, as classificações posteriores não irão passar por elas, pois esses atributos já foram validados pela regra de maior prioridade. Como se vê a interpretação das regras de menor prioridade fica confusa conforme a prioridade da regra diminui.

Outra forma de se ordenar as regras é pelas suas classes, no qual regras que pertençam à mesma classe aparecerão juntas nos critérios de teste sobre registros. Assim, uma ordenação das regras sem se considerar as classes é gerada. Após isso, as regras que foram ordenadas serão agrupadas segundo a classe a que pertençam, com a regra da classe que mais satisfaz o critério de ordenação em primeiro lugar. Após ela, virão as demais regras para essa classe. Nesse caso, como o agrupamento é relativo às classes das regras, regras de maior prioridade global podem ser preteridas em relação às de menor prioridade, com os registros testados por uma regra menos prioritária antes sobre essas últimas. A maioria dos classificadores baseados em regras mais conhecidos (como C.4.5 rules e Ripper) empregam esse esquema de ordenação.

2.3.1.2 Regras não ordenadas

A segunda estratégia é por regras não ordenadas: cada registro pode ser classificado por múltiplas regras de classificação. Cada vez que um registro é rotulado por uma regra, essa variável da classe a qual a regra pertence recebe um determinado "ponto" na discriminação da classe do registro. O registro pode receber o valor da classe com o maior número de votos, com o voto tendo um determinado peso dependendo de algum fator como cobertura, precisão, avaliação-m, etc. Podem-se unir vários métodos com determinados pesos no resultado final. Esse tipo de abordagem pode ser menos propenso a erros de escolha da ordenação da regra, já que considera todos os casos em que alguma regra é satisfeita pelo registro. No entanto o processo de classificação é bem mais demorado dado que determinados atributos do registro devem ser comparados aos testes realizados por cada uma das regras.

2.3.2 Aumento de regra

Deve-se encontrar uma determinada regra R que descubra a maior parte dos exemplos positivos do conjunto de treinamento e poucos ou nenhum exemplo que não corresponda a regra a ser descoberta para uma determinada classe Y_i. Dado o espaço exponencial de combinação de atributos para regras, descobrir uma regra ótima pode ser computacionalmente inviável. Então, o que pode ser feito é o aumento gradativo dos valores de atributos testados pela regra. Há duas espécies de aumento de regra: geral-para-específica e específica-para-geral.

2.3.2.1 Geral-para-específica

Nessa estratégia, uma regra inicial R:{}-> Y_i é criada. Essa regra classifica todos os registros de treinamento como sendo do tipo Y_i. Novos testes são adicionados posteriormente para melhorar a qualidade da regra. O melhor dos conjuntos de regras para uma determinada iteração é escolhido, adicionando-se um novo teste sobre atributo ao antecedente da regra. O processo segue até que não seja mais possível aumentar a qualidade da regra pela adição de um novo atributo. Essa estratégia é demonstrada na ilustração 6 abaixo, sobre o problema de classificação da Tabela 1, onde o teste sobre o atributo Est. Civil={Casado} é escolhido para aumentar a qualidade da regra. A partir desse teste, novos atributos serão adicionados no sentido de se aumentar a obtenção de exemplos positivos para a regra.

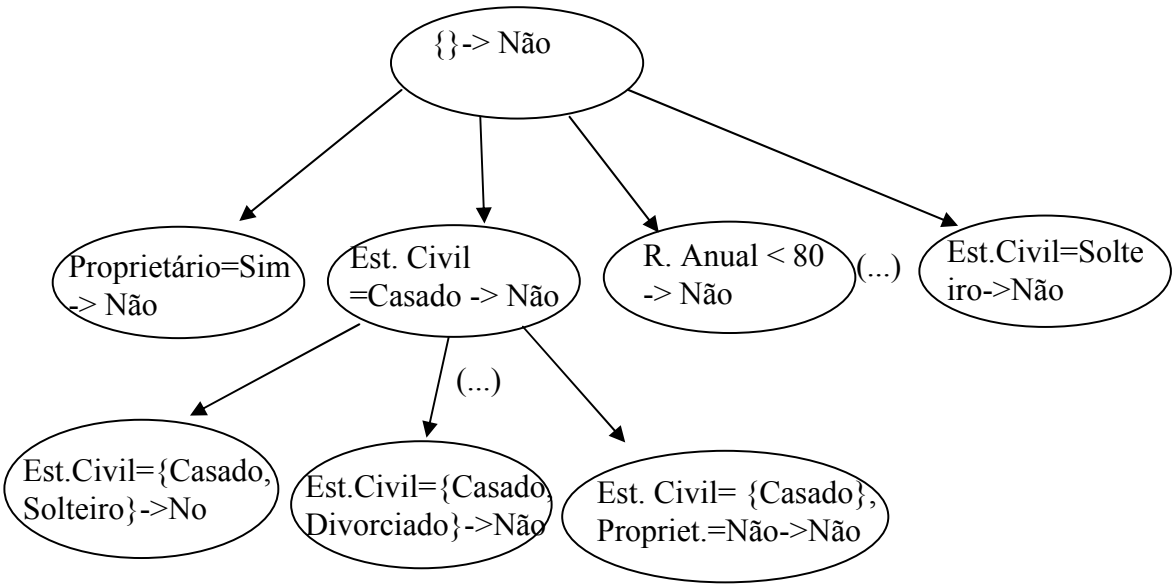


Ilustração 6: Estratégia de geração de Regras Geral para Específica

2.3.2.2 Específica-para-geral

Na abordagem específica-para-geral, escolhe-se aleatoriamente um dos exemplos de treinamento e o rótulo do mesmo como a classe alvo (exemplo positivo) e após, remove-se conjuntos de atributos existentes no exemplo positivo de modo a cobrir mais exemplos positivos. O processo definido na ilustração 7 demonstra essa estratégia começando com o registro 1 da Tabela 1. Continua-se a remoção pelo conjunto que produzir mais exemplos positivos em uma estratégia ávida para encontrar a melhor regra.

O processo terminará quando o critério de parada for estabelecido, ou seja, quando o refinamento de cada um dos nodos passar a cobrir exemplos negativos.

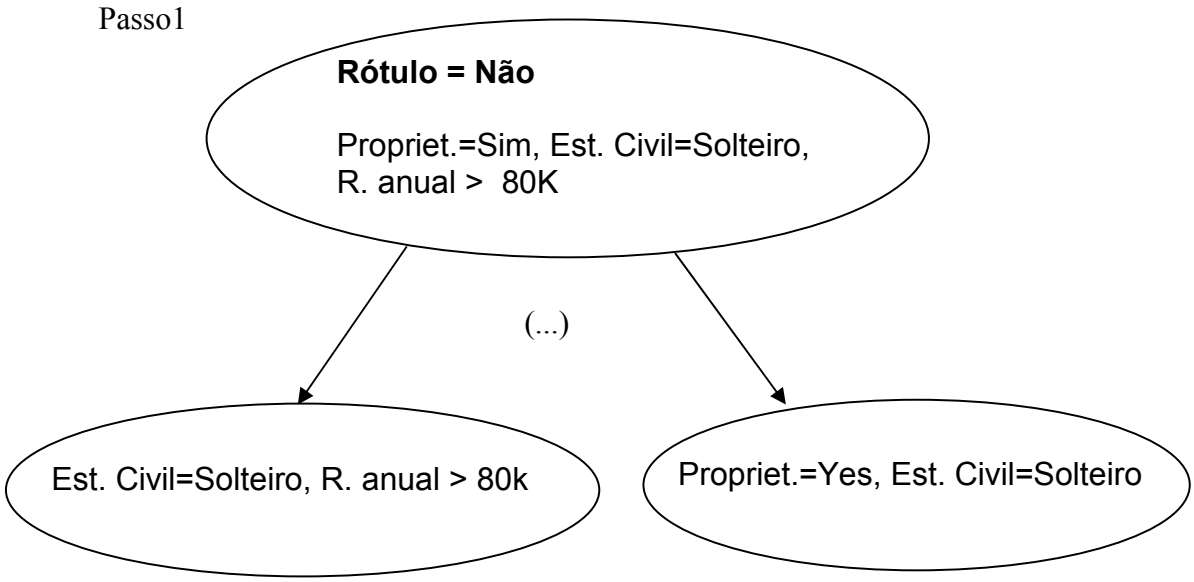


Ilustração 7: Estratégia de obtenção de regras específica para geral

Através das estratégias anteriores, pode-se produzir regras que não maximizem a cobertura de exemplos positivos. O algoritmo poderia então, para efeito de melhoria, manter algumas regras melhores candidatas segundo algum critério. Cada regra candidata poderia ser aumentada separadamente. A qualidade dessas candidatas seria reavaliada a cada iteração.

Após a descoberta das regras as mesmas podem ser podadas para diminuírem seus erros de generalização. As estratégias de poda são descritas na seção 2.7. Caso o erro de generalização diminua após a poda, as regras podadas são usadas em relação às originais.

Abaixo se tem a descrição de dois métodos diretos muito usados para a extração de regras, a saber: cobertura seqüencial e Ripper.

2.3.3 Cobertura Seqüencial

No método chamado cobertura seqüencial estabelece-se critérios de decisão de quais regras relacionadas a uma determinada classe devem ser geradas primeiro. Isto depende de alguns fatores como a predominância de registro da classe no total de registros ou o custo do erro de classificação da classe. Começa-se então com um conjunto de regras R vazias (ou lista de decisão vazia), um conjunto de registros de treinamento $E = \{(A_i, y_j)\}$, que contém pares de atributo-valor e um conjunto ordenado de classes $\{y_0, y_1,...,y_k\}$.

Através de uma função cujo objetivo é de se "descobrir uma regra" (alguma das estratégias específica - para geral ou geral-para-específica anteriores), considera-se cada

conjunto de registros pertencentes à classe ordenada y_i como um "conjunto positivo". O objetivo é então descobrir testes nos registros de treinamento que maximizem o conjunto de registros positivos encontrados (registros cujas classes pertencem a y_i). Os registros pertencentes a outras classes são considerados exemplos negativos. Assim que se encontra tal regra, os registros de treinamento pertencentes à regra em questão são desconsiderados e pega-se a próxima classe y_i, sendo a nova regra adicionada a R. Repete-se o processo para cada um dos y_i's do conjunto de rótulos, permitindo a geração de regras para a próxima classe.

2.3.4 Algoritmo de Ripper

O algoritmo de Ripper [3] é apropriado para inferir regras diretamente de registros com distribuição de classes desequilibradas. O conjunto inicial de treinamento é dividido, usualmente com 70% do conjunto de treinamento podendo ser usado para inferência das regras e a porcentagem restante utilizada como um conjunto de validação para evitar o overfitting de modelo.

De acordo com a freqüência das classes no conjunto geral dos registros de treinamento, as mesmas são ordenadas em ordem crescente em um conjunto $Y_0 = (y_1, y_2,..., y_n)$. A geração de regras é dada pelo algoritmo de cobertura seqüencial. O algoritmo escolhe a estratégia geral-para-específica para aumentar a regra e o algoritmo de Foil (ver equação 14) para escolher o melhor atributo a ser adicionado à regra. O algoritmo de aumento de regra pára quando a regra começar a cobrir exemplos negativos. Após isso, Ripper executa a poda da nova regra utilizando o conjunto de registros de validação de acordo com a métrica a seguir:

$$Poda(r) = \frac{p - n}{p + n} \qquad (15)$$

Nesta equação, p é o número de exemplos positivos do conjunto de validação cobertos pela regra, sendo n o total de exemplos negativos cobertos pela regra testados com o conjunto de validação. Caso a taxa de erro de validação exceda 50%, a regra não é considerada. Caso o valor de Poda(r) seja maior que a regra expandida, a regra simplificada é preferível e o conjunto coberto pela regra na validação é removido.

2.4 Classificadores de vizinho mais próximo

Basicamente, o classificador de vizinho mais próximo usa os exemplos de treinamento na classificação dos exemplos de teste e são conhecidos por “aprendizes diferidos”. Essa estratégia é diferente da estratégia ávida dos classificadores baseados em regras e da árvore de indução que derivam da construção de um modelo de classificação e só depois disso usam os exemplos de teste para testar o modelo criado. Esse classificador foi formalmente definido por Cover TM, Hart PE (1967) [22]. Classificadores de vizinho mais próximo representam cada exemplo de treinamento como um ponto em um espaço n-dimensional em que n é o número de atributos dos exemplos. Após isso, pode-se usar cada um dos exemplos de teste para inferir seu rótulo a partir da sua proximidade com relação ao conjunto dos pontos de treinamento. Os vizinhos mais próximos de um exemplo x no espaço n-dimensional referem-se aos k pontos que estejam mais próximos de x. Para isso, deve-se definir o termo ”proximidade”, assim como ter um meio de transformar um conjunto de atributos de um exemplo de treinamento em um valor que possa ser representado no espaço n-dimensional tal que n é o número de atributos. Algumas métricas de proximidade são discutidas na subseção 3.1.

Os k vizinhos mais próximos de um exemplo x referem-se aos pontos no espaço n-dimensional que estão mais próximos do “ponto” x. Esse classificador, portanto, não faz a inferência de um modelo de classificação a partir dos registros de treinamento como acontece com os classificadores de árvore de indução e de regras, mas usa a noção de proximidade para classificar um registro de teste a partir do treinamento. A idéia básica do vizinho mais próximo é de que “se algo anda como um pato e grasna como um pato então é provável que seja um pato” como é ilustrado pela ilustração a seguir:

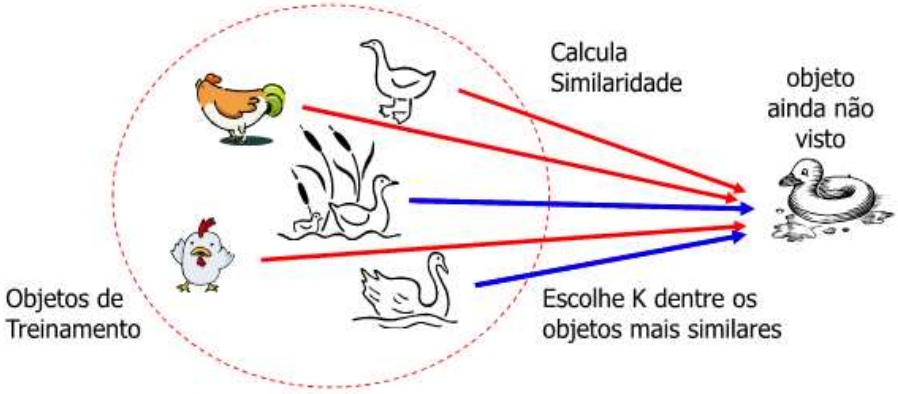


Ilustração 8: Idéia baseada na proximidade para um classificador de vizinho mais próximo

Então, o "ponto de teste" será classificado segundo as classes de seus vizinhos. No caso do conjunto dos k vizinhos serem de classes diferentes, uma estratégia deve ser adotada como o ponto de teste ser atribuído a classe que consta como rótulo do maior número de vizinhos. Entretanto alguns fatores devem ser levados em consideração, como o número dos k vizinhos mais próximos do ponto de teste x na classificação. O problema pode ser evidenciado pela ilustração abaixo:

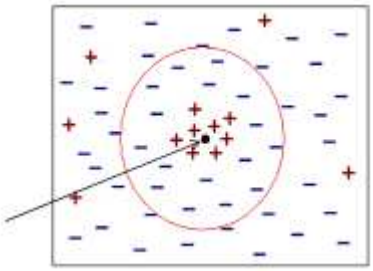


Ilustração 9: Escolha Errada de k grande

Na ilustração 9, k é grande demais para classificar de forma correta os vizinhos mais próximos do ponto de teste indicado pois o alcance para k grande inclui muitos pontos localizados longe da vizinhança do k. Por isso, usando-se a lógica da classe majoritária dos vizinhos de x, esse ponto seria classificado como da classe negativa, no entanto os exemplos negativos da vizinhança localizam-se muito longe de x para terem um peso igual aos pontos mais próximos. Um esquema de escolha da classe que envolve o peso será considerada um pouco mais adiante.

Para k pequeno demais também se tem um problema, pois pode ser suscetível a overfitting (tratado na subseção 4.1) devido ao ruído como exemplificado na ilustração 10:

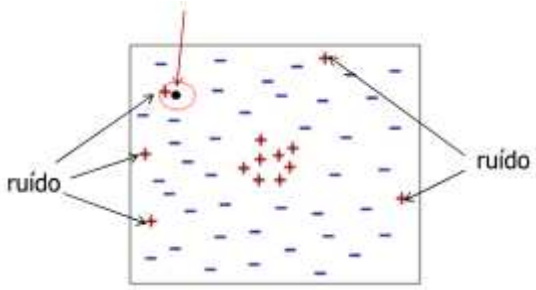


Ilustração 10: Overffiting devido a ruído para um k pequeno demais

Passa-se agora a uma explicação dos passos necessários para inferir a classe de um registro de teste arbitrário a partir dos registros de treinamento. Os passos primordiais são discutidos e explicados abaixo. Vale salientar que o cálculo da distância Euclidiana dada em

2.4 é comumente usado para calcular a distância entre os k pontos mais próximos e o ponto de teste x:

1) Pega-se um exemplo de teste $X=(x'_1,x'_2,...,x'_n)$ em que x'_i representa o conjunto dos atributos do ponto ou registro de teste tal que i varia de 1 até n

2) Para cada um desses exemplos de teste X, calcular a distância $d(X,W_i)$ entre X e o conjunto dos exemplos de treinamento W colocando-os em uma lista ordenada crescentemente Dz pela distância de W_i em relação a X da vizinhança de X. Para o cálculo da distância, escolhe-se alguma função de proximidade capaz de representar o registro X pelo calculo da proximidade de objetos com múltiplos atributos como exposto em na subseção 3.1.

3) Selecione os k primeiros pontos de Dz (como Dz estará ordenado crescentemente, essa etapa selecionará os k vizinhos mais próximos de X)

Ainda assim, usando a "votação majoritária" dos k vizinhos mais próximos para escolher a classe do registro de teste, tem-se que mesmo pontos de treinamento localizados muitos distantes do ponto X terão mesmo impacto para a classificação (isso pode eventualmente levar ao problema da ilustração 9, caso k seja grande). Essa situação pode ser resolvida pela adição de um parâmetro chamado peso p que atribui peso maior aos vizinhos mais próximos cuja distância *d(X,Wi)* calculada for realmente menor [3].

$$P_i = \frac{1}{d(X,W_i)^2} \tag{16}$$

E então, a computação da classe majoritária para o registro de teste X a partir dos pontos de treinamento pode ser computada da seguinte forma:

$$Y_{\max} = \arg\max_v \sum_{w_i \to Y_i \in D_k}^{n} P_i I(v = Y_i) \tag{17}$$

Na equação 17, Y_{max} representa a classe majoritária dos registros. Essa classe majoritária é obtida pelo maior dos somatórios de cada uma das classes de rótulo Y_i multiplicadas pelo peso P_i relativo ao ponto W_i. O peso é calculado pela equação 16 e D_k representa o conjunto dos k registros mais próximos de *X,* sendo *n* o número de pontos de treinamento (cada ponto individual é dado por W_i). A classe *Yi* de maior somatório é então elegida o Y_{max} e classificará o ponto *X*.

Finaliza-se a discussão com algumas das características mais importantes do classificador dos k vizinhos mais próximos:

O vizinho mais próximo não necessita manter um modelo a partir dos dados de treinamento. É assim um classificador baseado em instância. Apesar de não exigir a criação de um modelo, o processo de classificação pode ser bastante custoso, pois requer o cálculo da distância entre cada ponto de teste e todos os pontos do treinamento. Ainda com relação a isso, classificadores dos k vizinhos mais próximos fazem decisões locais sobre o rótulo do registro de teste enquanto os demais classificadores vistos (como árvore de indução e baseado em regras) produzem decisões globais pela própria natureza do modelo obtido. Isso leva a que os classificadores de vizinho mais próximo com k pequeno sejam sensíveis a ruídos nos dados.

Ainda, os limites de decisão do classificador do vizinho mais próximo são arbitrários e não retilíneos, diferentemente dos de árvore de decisão. No entanto, algumas vezes pode se tornar necessário evitar que alguns dados dominem completamente a medida de proximidade posto que seus valores de atributos sejam maiores do que os demais. Como um exemplo, supõe-se registros compostos por três atributos: altura(de 1.6 até 2.2), peso(de 50kg até 120kg) e salário(R$400 até R$50000). Considerando o cálculo das distâncias sem a devida transformação de valores (fazendo com que valores de atributos diferentes sejam mapeados para os mesmos intervalos, como explicado na subseção 3.1 de medidas de semelhança), as medidas de proximidade entre os registros seriam dominados pelo salário (possuirá valores maiores que os dos demais atributos).

2.5 Classificadores Bayesianos

Os classificadores Bayesianos usam probabilidade para inferir classificação adequada a uma instância de teste. Diferentemente de classificadores de árvore de decisão e dos k-vizinhos mais próximos, a natureza do problema resolvido por Bayes é não determinístico, ou seja, a classe correta para uma determinada instância não pode ser prevista com certeza.

2.5.1 Teorema de Bayes

O teorema de Bayes é mais antigo do que se pensa. A natureza probabilística de um problema resolvido por Bayes foi formalmente definida por Thomas Bayes e Richard Price (1763) [23]. Até hoje tal teorema é usado em mineração de dados com aplicação prática para inúmeros problemas inclusive classificação textual [24], com desempenho competitivo com outros classificadores conhecidos [25].

Antes da definição do Teorema de Bayes, deve-se definir a regra de multiplicação da probabilidade condicional ou incondicional de dois eventos. Para isso, pode-se considerar o problema abaixo:

"Em um teste são aplicadas duas questões de múltipla escolha, sendo a primeira de caráter V ou F e a segunda constando dos itens a, b, c, d ou e. Caso um aluno decida chutar todas as respostas, qual a probabilidade do aluno acertar as duas questões?"

Na primeira questão, tem-se 2 alternativas e na segunda questão 5. Pode-se construir duas variáveis aleatórias X e Y tal que X e Y estão relacionados às respectivas probabilidades de acertos de cada uma dessas questões. Seja P(X,Y) a probabilidade de que o aluno acerte ambas as questões, então, P(X,Y) = 1/10 = 0,1.

Pode-se, porém calcular as probabilidades individuais de ambas as questões: P(X) x P(Y) = 1/2 x 1/5 = 1/10.

Com isso pode-se verificar que:

$$P(X, Y) = P(X) x P(Y) \tag{18}$$

No entanto, isso nem sempre é verdadeiro. Há casos em que a probabilidade é condicional, ou seja, o que se quer saber é a probabilidade de ocorrência de um evento X qualquer dado que um evento Y ocorreu ou vice-versa. Isso é verdadeiro quando a ocorrência de um evento depende da ocorrência de outro. Assim, pode-se reescrever a equação acima

considerando probabilidades condicionais P(X | Y) (lê-se probabilidade de X ocorrer dado que Y ocorreu) ou P(Y | X) (representa a probabilidade de Y ocorrer dado que X ocorreu). Assim, reescreve-se a equação acima como:

$$P(X,Y) = P(Y \mid X)xP(X) = P(X \mid Y)xP(Y) \tag{19}$$

Um exemplo disso é ilustrado pelo seguinte problema:

"Retirando-se duas cartas de um baralho de 52 cartas, determinar a probabilidade da primeira carta ser um as e a segunda um rei sem reposição das cartas. "

A probabilidade do segundo experimento é afetado pela ocorrência do primeiro:

P(Ás) = 4/52

P(Rei) = 4/51

P(Ás,Rei) = P(Ás) x P(Rei) = 0,0060

Tomando-se os dois últimos termos da equação 19 encontramos o Teorema de Bayes:

$$P(Y \mid X) = \frac{P(X \mid Y)xP(Y)}{P(X)} \tag{20}$$

Em uma tarefa de classificação, X será o conjunto de valores de atributos e Y o valor da classe para a qual será avaliada a probabilidade. O Teorema de Bayes permite a avaliação dessa probabilidade através da probabilidade condicional P(X | Y), probabilidade anterior P(Y) e a evidência P(X).

P(X) será sempre constante (indica a probabilidade de que o conjunto de valores dos registros de teste seja encontrado no conjunto de treinamento) e poderá ser ignorado.

P(Y) é a fração dos registros de treinamento que pertencem a uma determinada classe, mais especificamente a classe a qual a probabilidade posterior se refere.

A avaliação da probabilidade condicional de classe poderá ser feita por alguma implementação do método de Bayes como Nayve Bayes ou Rede de Crenças de Bayes como será apresentado a seguir.

2.5.2 Nayve Bayes

Nesse caso, considera-se que os atributos de uma instância sejam condicionalmente independentes entre si. A dependência explícita é apenas dos atributos com o valor de classe. Isso traduz-se no fato de que a probabilidade condicional de uma determinada classe calculada sobre cada registro do treinamento é feito isoladamente para cada valor de atributo. Com isso, a equação 20 torna-se:

$$P(Y \mid X) = \frac{P(Y)\prod_{i=1}^{d} P(X_i, Y)}{P(X)} \tag{21}$$

Na equação acima, denominada de equação de Nayve Bayes [3], Y recebe o valor de uma classe conhecida y para ser avaliada. X é um conjunto de atributos sendo cada atributo denotado por X_i,

Cada valor de atributo X_i do registro o qual se quer saber a classe será testado considerando o rótulo de classe denotado por y. Isso é feito para cada y pertencente a Y. Após o cálculo de P(X|Y=y), substitui-se esse valor na equação 20 e calcula-se a probabilidade posterior P(Y|X) de cada classe. Após isso se pode considerar que a maior probabilidade posterior obtida dará o rótulo ao registro de teste.

Nesse sentido, duas abordagens podem ser usadas para o cálculo de $P(X_i,Y)$, considerando atributos nominais ou contínuos.

2.5.3 Atributos Nominais

Para atributos nominais, a probabilidade condicional é avaliada segundo a fração dos registros de treinamento da classe y que recebem um determinado valor X_i de atributo. Como um exemplo, supõe-se que foram analisados perfis de candidatos que procuram emprego na área tecnológica. Do conjunto de treinamento sabe-se que do total de 20 candidatos que conseguem emprego (rótulo da classe é SIM), 12 deles sabem ler inglês fluente. A probabilidade condicional de classe para o atributo “ler inglês” será então:

P(ler inglês = fluente | Sim) = 12/20 = 0,6

2.5.4 Atributos Contínuos

Nesse caso, duas abordagens podem ser utilizadas:

1) Pode-se discretizar (subseção 3.2) o atributo contínuo e fazer com que um determinado valor de atributo esteja associado a um determinado intervalo. Após a discretização a avaliação da probabilidade condicional de classe será a mesma utilizada para atributos nominais.

2) Utilizar uma distribuição de probabilidade qualquer (como a gaussiana abaixo) para avaliar $P(X_i,Y)$:

$$P(X_i \mid Y = y_i) = \frac{e^{-\frac{(x_i - \overline{x_{i,j}})^2}{2\,\sigma_{i,j}^2}}}{\sqrt{2\pi}\,\sigma_{i,j}} \qquad (22)$$

A distribuição gaussiana só será caracterizada por dois parâmetros: a média $\bar{x}$ e a variância σ^2.

Como um exemplo desses dois casos, considera-se a inferência de classes de documentos em economia, saúde e esporte, onde os atributos serão pares de valores palavra-frequência como pode ser visto na tabela 2 adaptada de [3].

Tabela 2: Documentos descritos por freqüências de palavras

Documento	Palavras(p,f)	categoria
1	(dólar,1);(indústria,4);(país,2);(empréstimo,3);(negócio,2);(governo,2),(aumento,1)	economia
2	(maquinário,2);(trabalho,3);(mercado,4);(indústria,2);(emprego,3);(país,1),(aumento,3)	economia
3	(emprego,5);(inflação,3);(trabalho,2);(desempregado,2);(mercado,3);(país,2);(índice,3)	economia
4	(doméstico,3);(previsão,2);(ganho,3);(mercado,2);(venda,3);(preço,2);(trabalho,2)	economia
5	(paciente,4);(sintoma,2);(remédio,3);(saúde,2);(clínica,2);(médico,2)	saúde
6	(farmacêutico,2);(empresa,3);(remédio,2);(vacina,1);(gripe,3),(aumento,1);(ganho,2)	saúde
7	(morte,2);(câncer,4);(remédio,3);(pública,4);(saúde,3);(diretor,2);	saúde
8	(médico,2);(custo,3);(aumento,2);(paciente,2);(saúde,3);(cuidado,1)	saúde

9	(Clube,5);(estádio,3);(previsão,1);(vitória,3);(técnico,6);(jogo,8);(árbitro,4)	esporte
10	(Clube,8);(torcida,5);(vitória,7);(técnico,4);(jogo,3);(jogador,7)	esporte
11	(Clube,4);(técnico,7);(torcida,5);(adversário,4);(estádio,5);(empate,2)	esporte

Calcula-se com base nos dados dessa tabela palavra-frequência a probabilidade condicional de classe para o atributo P(aumento=4 | economia) de um determinado registro de teste. Para isso, obtém-se primeiramente a média e o desvio padrão da amostra com relação a classe economia. Sendo a média obtida de:

$$\bar{x} = \frac{(1+3)}{4} = 1$$

E a variância será de:

$$\sigma^2 = \frac{(1-1)^2 + (3-1)^2 + (0-1)^2 + (0-1)^2}{3} = \frac{6}{3} = 2$$

A probabilidade condicional para nosso atributo contínuo será então:

$$P(aumento = 4 \mid Y = economia) = \frac{e^{\frac{-3^2}{4}}}{\sqrt{4\pi}} = 0.13228$$

Pode-se usar também discretização para transformar atributos contínuos em atributos nominais e usar a equação 21 para predizer a classe de uma instância de teste. No entanto ainda se tem um problema: se algum valor de atributo for nulo para alguma das classes (valor de atributo da instância de teste não for encontrado para alguma das classes das instâncias de treinamento), a probabilidade condicional P(X | Y) será nula e consequentemente P(Y | X) também será nula. Uma nova métrica de avaliação pode ser posta em prática para evitar esse problema. Pode-se usar estimativa-m para avaliar as probabilidades condicionais, como exposto a seguir:

$$P(X_i, Y_i) = \frac{n_c + m_p}{n + m} \qquad (23)$$

A estimativa-m analisa a probabilidade condicional de classe de um determinado atributo Xi para a classe Yi. As variáveis nessa equação podem ser assim explicadas: n_c é o número de registros de treinamento da classe Yi na qual o valor do atributo Xi está presente.

m_p é uma probabilidade especificada pelo usuário e que pode ser contabilizada sobre a probabilidade anteriormente observada para algum valor de atributo X_i na classe Y_i , n é o número total de registros da classe Y_i presentes no conjunto de treinamento e m é uma variável conhecida como tamanho da amostra equivalente.

Considerando uma instância de teste T = {(aumento,3), (médico,4), (previsão,2),(trabalho,3)} e os dados da tabela 2, pode-se calcular a probabilidade de cada classe para a equação 21. A classe com maior valor dado pela equação 21 é dita a classe predita para a instância de teste T.

Usa-se uma discretização de freqüência igual para discretizar as faixas de atributos contínuos (fazer com que cada valor caia em um determinado intervalo). Além disso, apenas discretiza-se os atributos que fazem parte do registro T a ser calculado. As seguintes faixas de atributos contínuos são mapeadas:

$$2 \leq trabalho \leq 5 \rightarrow 1$$
$$2 \leq medico \leq 5 \rightarrow 1$$
$$1 \leq previsão \leq 2 \rightarrow 1$$
$$2 < previsão \geq 4 \rightarrow 2$$
$$1 \leq aumento \leq 2 \rightarrow 1$$
$$2 < aumento \leq 4 \rightarrow 2$$

Supõe-se ainda os seguintes parâmetros:
m = 4 e m_P=1/4, caso Yi=economia
m = 4 e m_P= 1/4 caso Yi = saúde
m = 4 e m_P= 2/4 caso Yi=esporte

O valor de aumento para X é mapeado para o valor 2, pois "cai" nesse intervalo, assim, médico recebe o valor 1, previsão o valor 2 e trabalho o valor 1.

Para economia:

$$P(aumento = 2 \mid economia) = \frac{1 + \frac{1}{4}}{4 + 4} = 0.15625$$

$$P(medico = 1 \mid economia) = \frac{0 + \frac{1}{4}}{4 + 4} = 0.03125$$

$$P(previsão = 2 \mid economia) = \frac{1 + \frac{1}{4}}{4 + 4} = 0.15625$$

$$P(trabalho = 1 \mid economia) = \frac{3 + \frac{1}{4}}{4 + 4} = 0.40625$$

Assim, a probabilidade condicional para economia P(X|economia) é o produto de todas as probabilidades individuais acima, assim:

$$P(X \mid economia) = 0.15625\, x 0.03125\, x 0.40625 = 0.00031018$$

Analogamente, calculam-se as probabilidades condicionais para as demais classes. Após isso se calculam as probabilidades posteriores para Yi de cada classe usando a equação 23. Faz-se também w = 1/P(X), que é um valor constante.

$$P(economia \mid X) = 0.00031018\ x \frac{4}{11} w = 1.12793\, x 10^{-4} w$$

$$P(saude \mid X) = 0.00004294\ x \frac{4}{11} w = 0.15615\, x 10^{-4} w$$

$$P(esporte \mid X) = 0.000026033 x \frac{3}{11} w = 0.071 x 10^{-4} w$$

Agora pode-se afirmar que, como P(economia | X) > P(saúde | X) > P(esporte | X), o registro T = {(aumento,3), (médico,4), (previsão,2),(trabalho,3)} é classificado com o rótulo de economia.

Pode-se, portanto notar que se um determinado atributo Xi for irrelevante, a probabilidade condicional de classe para Xi não terá impacto no total da probabilidade condicional e a probabilidade posterior P(Y | X) não será afetada.

No entanto para atributos que estejam interligados, Naive Bayes pode não ser adequado por não considerar relacionamentos entre atributos. Nesse sentido uma Rede de Crenças Bayesiana pode ser construída conforme o próximo tópico.

2.5.5 Rede de Crenças Bayesiana:

Fornece uma representação gráfica dos relacionamentos probabilísticos entre os atributos das classes pelo mapeamento entre causas e efeitos. Isso é feito por meio de um grafo acíclico cujos nós correspondem a variáveis aleatórias. Nesse sentido, normalmente divide-se os nodos em nodos de causa e de efeito, dependendo da variável sendo representada pelo nodo ser de causa ou de efeito e ligam-se os nodos associados a causas a seus respectivos nodos de efeito. Essa "ligação direcionada" entre pares de nodos mapeia uma relação de dependência imediata entre dois nodos [26], [29].

Cada um desses nodos tem um mapeamento relativo aos estados da variável que ele representa e uma "tabela de relações/probabilidades condicionais" que indica os efeitos dos pais sobre o nodo (probabilidade do nodo estar em um estado específico relacionado com os estados de seus pais). Essa tabela de probabilidades é construída de modo que, caso o nodo X não tenha nodos pais associados, a tabela conterá apenas a probabilidade anterior P(X), no caso de apenas um pai Y, a probabilidade condicional associada será P(X|Y), se X tiver múltiplos pais a tabela conterá a probabilidade condicional $P(X|Y_1, Y_2,..., Y_k)$ [29].

Apesar de se considerar a probabilidade condicional de dependência de alguns atributos em relação a outros e não apenas ao rótulo da classe, como no caso do Nayve Bayes, segundo [29] a inferência probabilística da rede bayesiana poderá ser NP-hard e o tempo computacional da resolução de um problema por uma rede bayesiana pode ser muito alto.

A rede não deve possuir ciclos direcionados, pois um determinado nodo não pode constituir um efeito e ao mesmo tempo uma causa para uma dada variável aleatória de outro nodo, o que inviabilizaria o cálculo da probabilidade posterior. Uma propriedade importante desse tipo de classificador é a independência condicional, isto é, um nodo será condicionalmente independente de seus não descendentes se seus pais forem conhecidos. Um descendente de um nodo X será um nodo Y se e somente se existir um caminho direcionado de X para Y. Por exemplo, analisando o grafo abaixo:

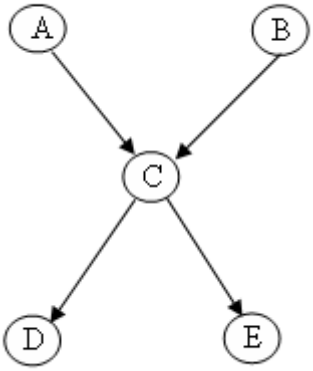


Ilustração 11: Dependência entre atributos em uma rede bayesiana

Vê-se que D é descendente de A e conseqüentemente A é ancestral de D. Esse grafo ilustra a independência condicional, pois D dado C é condicionalmente independente de A, B e E, seus não descendentes, caso a probabilidade condicional calculada for P(D|C).

A metodologia de construção de modelos de redes Bayesianas envolve duas etapas principais: a primeira passa pela construção da estrutura da rede e a segunda passa pela inferência das probabilidades associados a cada nodo. Na etapa 2 usa-se o conhecimento de especialistas para inferir as probabilidades associadas aos nodos. A etapa 1 pode ser formulada como segue:

1) Ordenação de todas as variáveis de atributo ou classe $T=\{X_1,X_2,...,X_d\}$

2) Para cada variável de T denotada por Xi, considerar o conjunto de variáveis predecessoras de Xi denotada por $pred(Xi)=\{X_0,...,X_{i-1}\}$

3) Fazer a remoção de todas as variáveis de pred(Xi) que não afetam Xi

4) Criar aresta saindo de cada um dos nodos restantes em pred(Xi) e chegando em Xi

Uma observação importante é que dependendo da ordenação das variáveis, pode-se chegar a diferentes topologias com algumas produzindo muitas arestas desnecessárias. Uma solução é separar as variáveis em variáveis de causa e de efeito como explicado anteriormente. Após a construção da topologia e da tabela de probabilidades dos nodos, a resolução das probabilidades condicionais é similar ao Bayes simples(equação 21).

Para exemplificar a topologia de uma rede Bayesiana, considera-se o exemplo de rede abaixo [29], cujos atributos representam Cloudy (tempo nublado) , Sprinkley (regador), Rain (chuva) e Wet Grass (grama molhada). Pretende-se resolver através da rede o seguinte

problema: dado que a grama está molhada, qual a evidência de ter chovido ou do regador estar ligado?

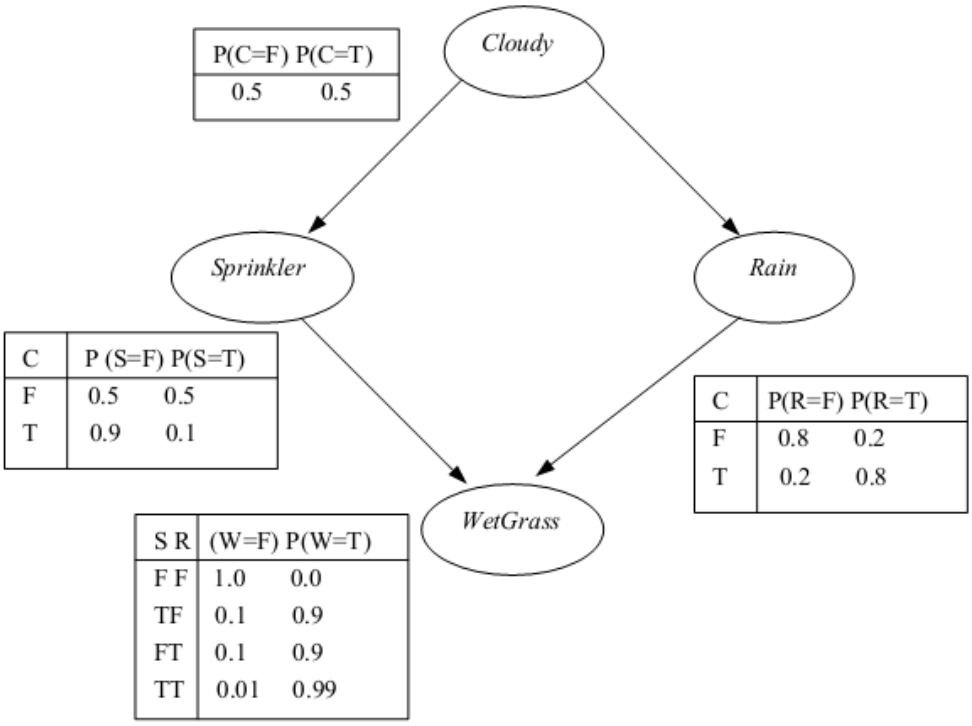


Ilustração 12: Exemplo de uma Rede de Crenças Bayesiana

As probabilidades para as variáveis aleatórias dos filhos é dependente, como explicado, das probabilidades ditadas por especialistas do domínio para os pais. Então a seguinte relação de dependência pode ser verificada:

$$P(x_1,...,x_n) = \prod_{i=1}^{n} P(x_i \mid pais(x_i)) = \coprod_{i=1}^{n} P(x_i, \pi_i) \qquad (24)$$

Dada a estrutura da rede Bayesiana acima, devem ser calculadas duas probabilidades para o problema:

P(R=T | W=T) e P(S=T | W=T)

Pode-se utilizar o Teorema de Bayes (equação 21) para resolver o problema. Torna-se necessário a resolução das duas equações abaixo:

$$P(R = T \mid W = T) = \frac{P(W = T, R = T)}{P(W = T)}$$

$$P(S = T \mid W = T) = \frac{P(W = T, S = T)}{P(W = T)}$$

Como W (evidência) é filho tanto de R quanto de S, R e S tornam-se dependentes. Tanto S quanto R são dependentes de C (Cloudy). O cálculo das probabilidades condicionais acima tornam-se:

$$P(R = T \mid W = T) = \sum_{C,S} \frac{P(C = c_i, S = s_i, R = T, W = T)}{P(W = T)}$$

$$P(S = T \mid W = T) = \sum_{C,R} \frac{P(C = c_i, R = r_i, S = T, W = T)}{P(W = T)}$$

Calculando P(W=T) tem-se:

$$P(W = T) = \sum_{C,R,S} P(C = c_i, R = r_i, S = s_i, W = T)$$

=0.5[(0.9x0.5x0.8)+(0.9x0.1x0.2)+(0.9x0.9x0.8)+(0.99x0.5x0.2)+(0.99x0.1x0.8)]

=0.6471

Calculando para P(R=T|W=T) e P(S=T|W=T):

$$P(R = T \mid W = T) = \frac{0.4581}{0.6471} = 0.70793$$

$$P(S = T \mid W = T) = \frac{0.2871}{0.6471} = 0.44367$$

Chega-se aos resultados do problema para as duas probabilidades.

Algumas considerações finais sobre a Rede de Crenças Bayesiana é que a topologia da rede captura conhecimento através de um grafo acíclico da qual a dependência entre as variáveis aleatórias é dada pela direção da aresta, sendo a probabilidade do nodo "apontado" dependendo da probabilidade de seus pais imediatos caso a probabilidade condicional P(Filho|Pai) seja verificada. A construção requer tempo, experiência e muito esforço para se alcançar um bom resultado ou mesmo aceitável. Também requer a presença e intervenção imediata de um especialista de domínio para ditar as probabilidades de cada nodo e suas dependências. No entanto, o acréscimo de uma nova variável dado que a rede esteja montada é direto, não precisando na maioria das vezes da reavaliação da probabilidade dos ancestrais do nodo criado.

São apropriadas para lidar com dados faltantes ou incompletos. Atributos incompletos podem ser avaliados pela soma das probabilidades para todos os valores possíveis do atributo.

Dado que os dados são combinados probabilisticamente com as probabilidades de variáveis contidas em nodos superiores (conhecimento anterior), o overfitting pode ser modelado e evitado de maneira menos trabalhosa.

2.6 Support Vector Machine

Na classificação por SVM, cada instância de treinamento pode ser vista como um conjunto de atributos onde cada valor de atributo estará associado a uma dimensão diferente em um espaço n-dimensional.

A idéia principal do SVM baseia-se na resolução de um problema de classificação no qual se separam as instâncias de treinamento correspondentes a duas classes distintas através de um hiperplano de máxima margem, o que será discutido a seguir. Os modelos de SVM podem ainda ser divididos em modelos com separação linear, modelos não separáveis linearmente ou modelos não lineares, tratados nas subseções seguintes [3], [5], [9].

Nesse sentido, muitos trabalhos tem tido como base a classificação por SVM's, como [5], [9] e [25], e apresentam para uma variedade de problemas distintos um desempenho superior a outros classificadores.

Finaliza-se a seção com duas técnicas que visam estender o modelo de classificação dual do SVM para mais de duas classes: pairwise ou um-contra-restante.

2.6.1 Hiperplanos de Margem Máxima

A ilustração 13 a seguir apresenta um conjunto de dados compostos por quadrados e círculos e alguns exemplos de separação linear desse conjunto.

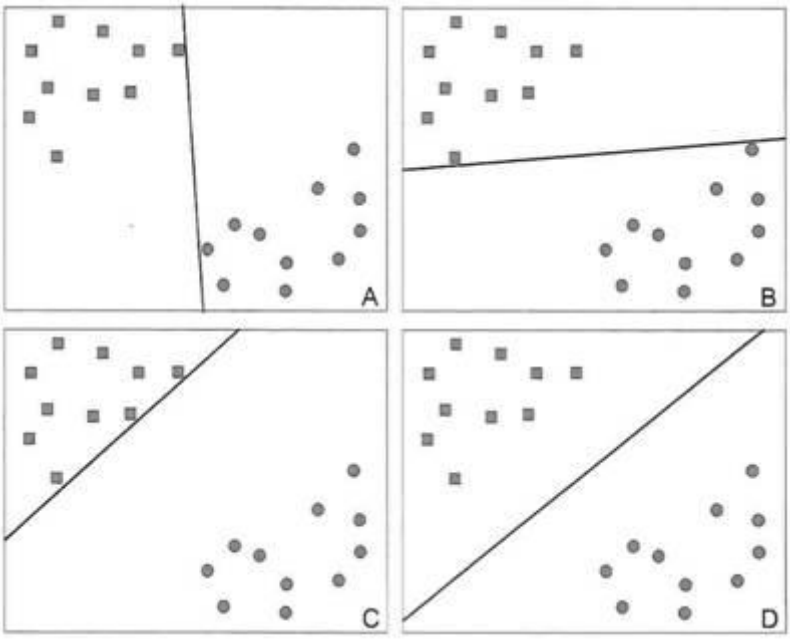


Ilustração 13: Alguns limites de decisão possíveis para um conjunto de dados separáveis linearmente

Como pode ser visto acima, o erro de treinamento de cada modelo, representado por A, B, C ou D é 0 (todos os modelos conseguem separar completamente os objetos pelas suas classes). Contudo, o número de hiperplanos que podem ser encontrados para o exemplo acima com erro de treinamento 0 é muito grande (foram listados apenas 4 deles). Caberá, portanto ao classificador a escolha do melhor modelo que represente as instâncias de treinamento. Considera-se aqui o fato de que essa melhor representação traduz-se na minimização dos erros de generalização para instâncias não vistas previamente e que embora o erro de treinamento de todos os modelos seja 0, seu erro de generalização pode ser alto.

Nesse sentido a verificação do tamanho da margem de um classificador pode ser um fator importante para a comparação de classificadores com erros de treinamento 0.

A margem de um classificador pode ser estimada da seguinte forma: consideram-se dois outros hiperplanos paralelos ao hiperplano de separação do modelo. Esses dois hiperplanos são deslocados a partir do hiperplano que representa o limite de decisão do modelo em direções opostas até que cada um deles toque nos objetos mais próximos do limite de decisão. Os objetos que fazem parte de cada um desses "hiperplanos deslocados" são denominados vetores de suporte e a distância entre esses hiperplanos é a margem do classificador. Isso pode ser visto na ilustração a seguir:

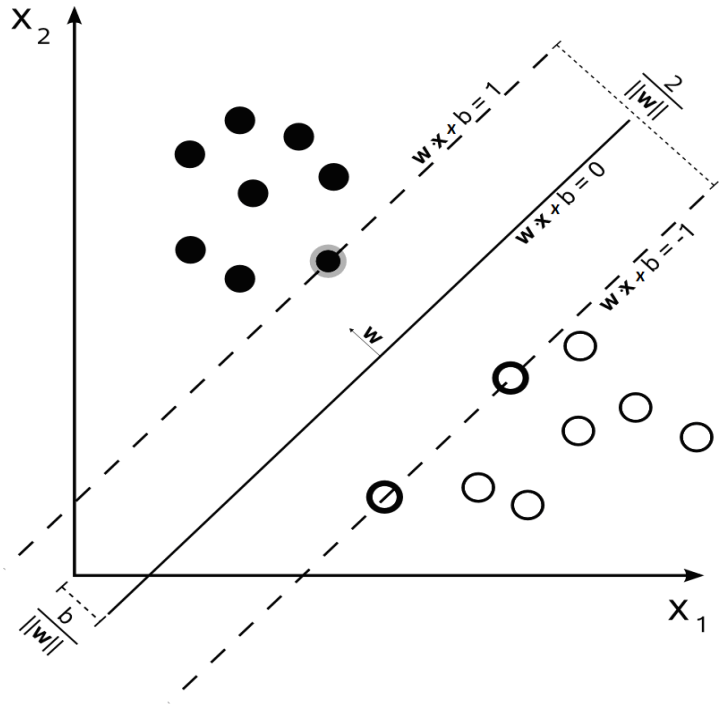


Ilustração 14: Margem de um limite de decisão

Na ilustração acima, a margem do classificador é dada por d = 2/ || w ||.

Observa-se que o número de vetores de suporte para o hiperplano deslocado à esquerda do limite de separação é apenas um (o círculo preenchido cortado pelo hiperplano) e para o hiperplano deslocado à direita do limite de decisão contam-se dois vetores de suporte (as duas instâncias representantes de círculos "brancos" cortados pelo hiperplano wx – b = -1)

Nesse sentido quanto maior a margem de um classificador, maior a tolerância do classificador a erros na classificação de instâncias não vistas pelo modelo, ou seja, se ocorrerem erros na localização dos hiperplanos que passam pelos vetores de suporte, uma margem maior dará menor chance de erros de generalização dado que os objetos não vistos terão semelhança com os objetos usados na confecção do modelo. Intuitivamente qualquer perturbação nos dados por mínima que seja poderá ocasionar maiores erros de generalização em modelos com margens de separação muito próximas entre classes do que naqueles com margens mais afastadas. Voltando à ilustração 13 anterior, pode-se dizer que o hiperplano de margem máxima poderia ser o C ou D. No entanto, D seria preferível a C para indicar a margem máxima do modelo pois qualquer perturbação na margem de decisão de C pode alterar a classificação de registros de teste localizados no limite dos quadrados.

2.6.2 SVM Linear Separável:

Nesse caso o problema de classificação pode ser resolvido pela separação completa das classes através de um hiperplano de margem máxima. Erros de treinamento não são tolerados e o classificador para o problema apresentará sempre erro de treinamento 0. Analisa-se novamente a ilustração 14, w será um vetor perpendicular ao plano de separação, o conjunto dos atributos de cada exemplo de treinamento é conhecido como x_{jk} tal que j representa o objeto o qual o atributo faz parte e k é o k-ésimo atributo do objeto j. O limite de decisão, como pode ser visto pela ilustração, é denotado por:

$$w \bullet x + b = 0 \qquad (25)$$

É possível também demonstrar pela ilustração 14 que qualquer objeto localizado acima do hiperplano de separação (círculo preenchido, x_{cp}) será descrito por uma constante positiva k de modo que:

$$w \bullet x_{cp} + b = k \qquad (26)$$

Analogamente, os círculos vazados x_{cv}, serão descritos por uma constante negativo k' de modo que:

$$w \bullet x_{cv} + b = -k' \qquad (26)$$

Rotulando todos os círculos preenchidos com a classe *y*= +1 e todos os círculos vazados com a classe *y*= -1, pode-se achar o "tipo da classe" de cada um dos objetos localizados acima ou abaixo da decisão pelo seguinte:

$$y = 1, \text{ se } w\,z + b > 0$$
$$y = -1, \text{ se } w\,z + b < 0 \qquad (27)$$

As equações acima podem ser reescritas considerando que todos os objetos localizados acima da margem de separação da ilustração 14 serão da classe +1 e todos os objetos abaixo da mesma margem serão da classe -1. Os hiperplanos de separação podem então ser expressos conforme a ilustração 14 por:

$$w \bullet x + b = +1$$
$$w \bullet x + b = -1 \qquad (28)$$

Pode-se usar então a seguinte equação sendo y_i= +1 ou y_i= -1 na determinação das classes de objetos:

$$y_i (w \bullet x_i + b) \geq 1, i = 1,2,...,n \qquad (30)$$

Sabe-se também pela Ilustração 14 que o tamanho da margem pode ser dado por:

$$d = \frac{2}{|| w ||} \qquad (29)$$

Ainda: essa margem deve ser máxima. Maximizar a margem necessita da resolução da seguinte equação de otimização restrita, onde a função objetiva é quadrática, sujeita às restrições lineares dadas pela equação 30:

$$\min(w) = \frac{|| w ||^2}{2} \qquad (30)$$

Esse problema também é denominado um problema de otimização convexo e pode ser resolvido pelo método dos multiplicadores de lagrange [3],[5]. Basicamente isso envolve a reescrita da equação objetiva com suas restrições onde cada instância será associada a um multiplicador de lagrange conforme abaixo:

$$L_p = \frac{|| w ||^2}{2} - \sum_{i=1}^{n} \lambda_i (y_i [w \bullet x_i + b] - 1) \qquad (31)$$

Essa minimização envolve a busca pelos mínimos locais de equação acima de modo a encontrar w e b. Para isso pega-se a primeira derivada de Lp em relação a w e b, igualando-se os resultados a zero. Explicitar o método de resolução do problema de otimização foge ao escopo desse trabalho. Abaixo se tem um exemplo do cálculo da classe de objetos no R_2

separados por um hiperplano não necessariamente máximo, mas que servirá para ilustrar a teoria de SVM linear separável aqui exposta. Considera-se a ilustração abaixo, onde há exemplos de objetos das classes losango e triângulo:

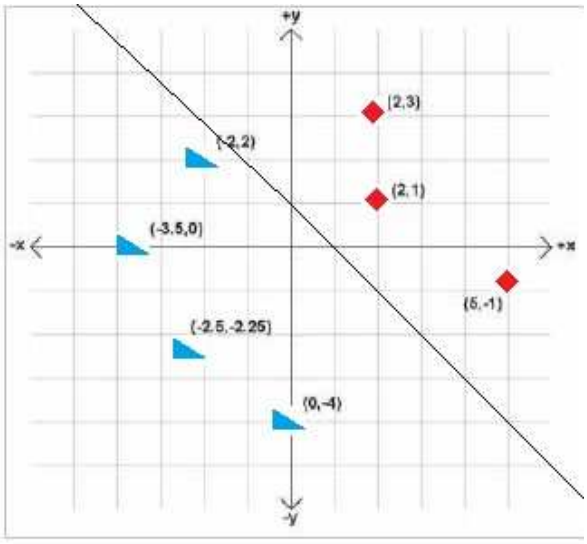


Ilustração 15: SVM linear separável no R_2

Considerando o que foi exposto acima, no R^2 cada instância é composta por dois atributos distintos denotados acima por x e y. O hiperplano de separação entre as instâncias será denotado nesse caso por:

H: $W_1X+W_2Y = 0$

Como o hiperplano denota uma reta, bastam 2 pontos para defini-lo. Pegando-se os pontos (1,0) e (0,1) pertencentes aos hiperplanos pode-se encontrar W_1 e W_2.

A equação que define o hiperplano é dada por y = -x + 1. Pode-se calcular a norma do vetor W desse jeito:

$|| W ||= \sqrt{W_1^2 + W_2^2}$, o resultado será 1.

A partir do cálculo de W e B pode-se usar a equação distância de um ponto a uma reta dado abaixo em termos do vetor W e sua norma. Os pontos cuja distância r for positiva serão classificados como losangos, de outro modo serão triângulo.

$$r = \frac{W \bullet X + b}{|| W ||} \tag{32}$$

Considerando por exemplo o ponto (-2,-2) pode-se inferir a classe:

$$r = \frac{W_1 X + W_2 Y}{|| X ||} = \frac{1(-2) + 1(-2)}{1} = -4$$

Como o resultado é negativo, a instância é classificada como triângulo.

Similarmente para a instância cujos atributos valem (6,-1) tem-se um valor de r de 5, o que classifica a instância como losango.

2.6.3 SVM linear não Separável:

No caso anterior o problema de classificação podia ser perfeitamente separado por um hiperplano de margem máxima sem erros de classificação.

No entanto esse é um cenário idealizado, pois a grande maioria dos problemas de classificação real apresenta ruídos nos dados, instâncias mal classificadas ou que fogem ao padrão real. Portanto uma separação completa de classes por um hiperplano máximo é impossível e ainda que possível, a consideração de erros de treinamento pequenos em benefício de uma margem maior pode ser preferível. Assim o modelo explicado anteriormente serve apenas como base para o entendimento do conceito de máxima margem e não serve para exemplos reais.

Considera-se aqui a construção de um modelo SVM tolerante a pequenos erros de treinamento para a classificação dos dados. Como um exemplo do comportamento desse modelo, considera-se a ilustração 16 dada abaixo.

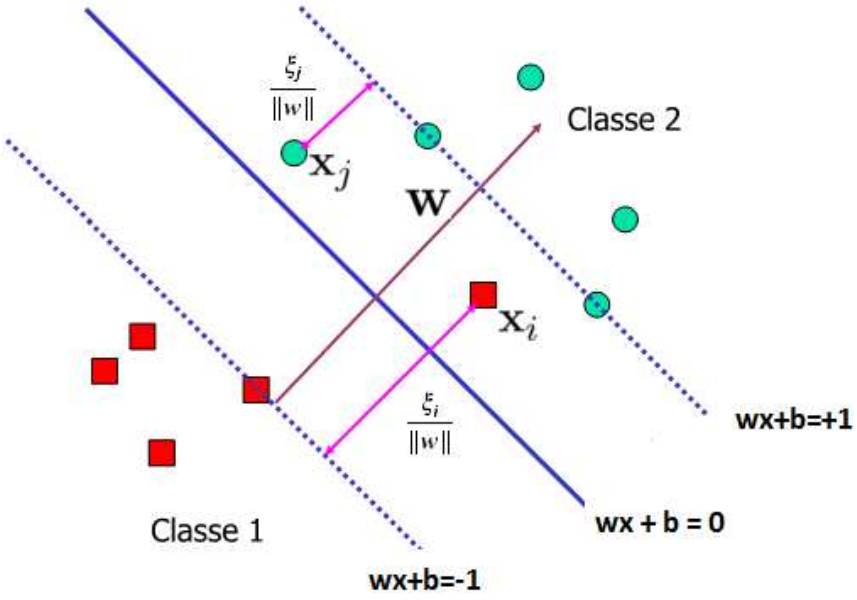


Ilustração 16: margem de decisão para dados não separáveis linearmente

A função objetiva anterior a ser minimizada continua a mesma, porém agora as restrições devem tratar um caso adicional: além da maximização da margem, pode-se aceitar erros de classificação pela introdução de variáveis de folga positivas (Slack ξ) nas restrições do modelo.

O significado dessas variáveis é tornar o modelo tolerante a pequenos erros de classificação. A procura pelo melhor modelo SVM representativo dos dados traduz-se em maximizar a margem e minimizar o número de erros de treinamento gerado.

As restrições do problema de otimização considerando o que foi exposto podem ser então reescritas como abaixo:

$$\begin{aligned} & w \bullet x_i + b \geq 1 - \xi_i, y_i = +1 \\ & w \bullet x_i + b \geq -1 + \xi_i, y_i = -1 \\ & \forall i : \xi_i > 0 \end{aligned} \tag{33}$$

Assim, cada exemplo de treinamento mal rotulado irá se adequar às restrições do problema pela introdução de uma variável Slack para estimar o erro de cada exemplo mal classificado.

Pela ilustração 15 observa-se que o quadrado x_i deveria estar abaixo do hiperplano *wx+b=-1* ou então fazer parte do mesmo.

Pode-se estimar o erro para essa instância através do cálculo da distância entre o quadrado x_i acima do limite de decisão e o hiperplano *wx+b = -1* é dado por $\xi_i / \| w \|$

Como foi mostrado, *w* é fixo e a respectiva variável slack fornece uma boa estimativa do erro individual de treinamento para cada exemplo mal rotulado.

No entanto ainda configura-se um problema: o classificador agora permite que um modelo tenha um erro de treinamento não zero e consegue encontrar uma margem grande para o hiperplano, mas não há restrição sobre o número total de erros de classificação cometidos. Esse problema pode ser resolvido pela introdução de um fator a mais na função objetiva original visando penalizar uma quantidade muito grande de erros de classificação. Na equação abaixo são introduzidas duas variáveis k e C (custo) que permitem especificar a quantidade máxima total de erros de classificação permitidos. A função objetiva modificada pode ser vista na equação abaixo:

$$f(w) = \frac{\| w \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i \right)^k \tag{34}$$

Considerando a nova função objetiva com os máximos erros de classificação permitidos segundo as variáveis slack, o Langrageano para o problema de otimização seria dado por:

$$L = \frac{1}{2} \| w \|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i [y_i (w \bullet x_i + b) - 1 + \xi_i] - \sum_{i=1}^{N} \mu_i \xi_i \tag{35}$$

Nesse equação, os dois primeiros termos são a função objetiva a ser minimizada com o segundo termo (contabilização de erros de classificação) tendo k=1. O terceiro termo refere-se às restrições de diferença dos ξ_i's, ou seja, é permitido erros de classificação dados pela verificação dos valores desses Slacks. O quarto termo refere-se a que os valores dos slacks devem ser maiores do que zero.

2.6.4 Classificadores SVM não lineares:

Nas metodologias anteriores considerava-se que existia um hiperplano capaz de separar as instâncias de treinamento em dois conjuntos distintos onde os dados ou eram separáveis linearmente com erro de treinamento 0 (SVM linear separável) ou com erro de treinamento pequeno segundo variáveis Slack e penalizando erros de treinamento muito grandes (SVM linear inseparável).

Entretanto em outros casos não seria possível encontrar um hiperplano que separasse as instâncias mesmo com erro de treinamento pequeno. Isso pode ser visto na ilustração 17, retirada de [5], onde uma curva seria mais apropriada para separar instâncias de classes diferentes.

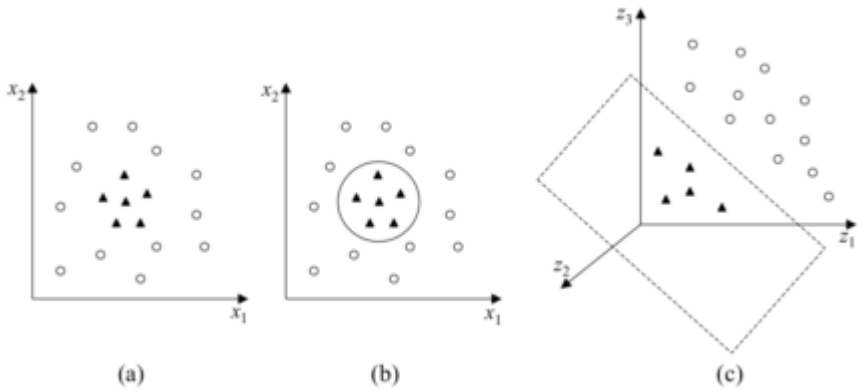


Ilustração 17: (a) Dados não separáveis linearmente, (b) Fronteira não linear, (c) Fronteira linear no espaço transformado.

Nesse sentido algumas abordagens podem ser utilizadas para se resolver o problema. No item c da ilustração acima, considera-se, por exemplo, fazer uma transformação do espaço original x, conhecido como espaço de entradas, para um novo espaço de maior dimensão conhecido por espaço de características (o espaço de características é denotado por Φ(x)). Após a transformação dos atributos das instâncias do espaço original para outro de maior dimensão pode-se usar as técnicas anteriores com a técnica de otimização restrita sendo resolvida para se encontrar um hiperplano no espaço transformado.

Nesse sentido a função objetiva quadrática anterior torna-se:

$$\min_w \frac{\| w \|^2}{2}$$
$$s.a : y_i (w\Phi(x_i) + b) \geq 1, i = 1,..., N \qquad (36)$$

Isso é motivado pelo Teorema de Cover [5] que diz que dado um conjunto de dados não linear no espaço de entradas x, pode-se encontrar uma transformação de x para um novo espaço Φ(x), denominado espaço de características, com alta probabilidade de os dados no espaço transformado serem separáveis linearmente desde que a transformação seja a partir de um espaço não linear e a dimensão do espaço de características seja suficientemente alto.

Pode-se portanto usar a transformação abaixo para transformar os dados em (a) no R^2 para R^3:

$$\Phi(x) = \Phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

Após a transformação, encontra-se um hiperplano capaz de separar os dados linearmente. Na ilustração acima tal hiperplano é dado por:

$$f(x) = w\Phi(x) + b = w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 + b = 0$$

Pode-se observar que o problema é linear em R_3 (ilustração (c)), mas não linear em R_2 (ilustração (b)).

Na prática, no entanto não é tão fácil saber a função de mapeamento correta para cada caso não linear. Outro problema que surge a partir disso é que resolver o problema de otimização restrita no espaço não linear pode ser computacionalmente custoso pois envolve o produto de ponto (tido como o cálculo da similaridade entre dois vetores) no espaço já transformado (multidimensional).

Seguindo os mesmos passos adotados no caso linear, será obtido uma equação langrageana conforme abaixo:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i\lambda_j \Phi(x_i) \bullet \Phi(x_j) \qquad (37)$$

A resolução desse problema requer que os termos λi's sejam obtidos por programação quadrática [3],[5] e a derivação dos parâmetros w e b conforme o caso linear. No entanto, diferentemente do caso linear, o cálculo de $\Phi(x_i)$ $\Phi(x_j)$ pode ser resolvido por uma função de núcleo, que calcula a similaridade entre os pontos do espaço transformado mas usando o espaço original. Assim, pode-se usar a seguinte função de núcleo para inferir a separação de classes em (b):

$$\Phi(x)\Phi(y) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \bullet (y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, 1) \qquad (40)$$

As funções de núcleo possuem algumas características importantes, uma delas é o fato de que permite o calculo da similaridade sem a necessidade de se saber a forma da função exata de mapeamento Φ entre o espaço original e o transformado, pois todas as funções de

núcleo obtidas devem satisfazer ao teorema de Mercer. Esse princípio matemático determina que a função de núcleo calculada para um par de vetores deve corresponder ao produto de ponto desses vetores no espaço de transformação.

Sendo assim o cálculo da equação anterior equivale a usar a seguinte função de semelhança:

$$k(x, y) = (x \bullet y + 1)^2 \tag{38}$$

Esse espaço de transformação é chamado espaço Hilbert de núcleo de Reprodução (RKHS) e realizar esse cálculo é menos custoso do que usar uma função de mapeamento para a produção do espaço de características, pois ainda era necessário a obtenção da melhor função para cada caso SVM não linear. Com a função de núcleo o calculo já pode ser realizado diretamente com a vantagem dos cálculos serem executados no espaço original e a complexidade de se calcular o produto vetorial em dimensões mais altas: R^4, R^5, etc, não existe com o núcleo, que constrói um hiperplano no espaço de características não estando nele.

2.6.5 SVM para múltiplas classes

Na sua concepção original, SVM apenas faz classificações binárias. No entanto para muitos problemas práticos como reconhecimento de dígitos, análise digital, classificação de textos, etc, que requerem conjuntos de objetos de mais de dois tipos de classes a abordagem clássica do SVM precisa ser refinada de alguma forma.

Uma das abordagens para se fazer classificação multiclasse pelo SVM é pela redução de um problema multiclasse simples em vários problemas de classificação binária.

Nesse sentido duas abordagens de resolução serão consideradas. A primeira é denominada um-contra-restante, no qual sendo y o número total de classes existentes para o problema, são feitas y classificações binárias considerando as instâncias de cada classe em questão a seu turno como um exemplo positivo. Se o classificador escolher para a instância a classe sendo considerada, essa será a classe a ser atribuida. Caso contrário a instância será considerada um exemplo negativo e o classificador prossegue tentando inferir a classe da próxima instância. Depois que todas as instâncias foram avaliadas como sendo da classe yi, as que foram consideradas exemplos negativos prosseguem e o classificador considera como exemplos positivos todas as classes pertencentes a yi+1

A outra abordagem é denominada pairwise ou um-contra-um. Nesse caso são construídos y(y-1)/2 classificadores em que cada classificador é usado para distinguir entre um par de classes (y_i,y_j). Nesse caso a classe a qual determinada instância pertence é atribuída normalmente por votação majoritária no total de classificações obtidas.

3 FUNÇÕES DE SIMILARIDADE E DISCRETIZAÇÃO

3.1 Métricas de Proximidade:

O objetivo dessa seção é enunciar algumas medidas de proximidade entre dois objetos que levam em conta a similaridade entre seus atributos. Isso permite determinar um valor numérico indicativo do grau de semelhança ou diferença desses objetos.

De um modo geral, a natureza dos atributos de dois objetos indica a forma de comparação a ser realizada para se medir o grau de semelhança ou de diferença entre esses objetos. Por exemplo, em atributos binários, a similaridade pode ser indicada por 1 caso os valores dos atributos correspondam e 0 caso contrário. Caso os atributos sejam contínuos, diferenças usando distâncias (como a distância euclidiana) podem ser calculadas.

Antes de se discutir as medidas de semelhanças para atributos múltiplos, transformações de valores podem ser necessárias, como explicado a seguir.

3.1.1 Transformação

Algumas vezes é necessário transformar um valor de atributo para cair em uma determinada faixa.

Isso impede que determinados atributos influam de maneira significativa no cálculo das medidas de semelhança ou diferença, principalmente com atributos contínuos ou então pode ser o caso, o software específico usado para calcular similaridade entre objetos só trabalha com uma determinada faixa de valores [3]. Por exemplo, a transformação de um valor contínuo qualquer para um intervalo de [0,1] pode ser realizado através da equação 42 dada a seguir.

$$k' = \frac{(k - \min(k))}{(\max(k) - \min(k))} \tag{39}$$

Na equação acima, k' representa o novo valor e k o valor antigo, min(k) representa o menor valor do intervalo antigo e max(k) é o maior valor do intervalo antigo.Outras transformações segundo [3] podem ser:

$$k' = \frac{1}{(d+1)}$$

$$k' = 1 - \left(\frac{d - \min(d)}{\max(d) - \min(d)} \right) \tag{43}$$

$$s = e^{-d}$$

A seguir definem-se algumas medidas de semelhança para atributos múltiplos. Uma explicação mais detalhada de cada método pode ser encontrado em [3].

3.1.2 Distância Euclidiana:

É uma medida da distância entre dois pontos que pode ser provada pela sucessiva aplicação do Teorema de Pitágoras e mede a semelhança entre objetos cujos atributos são contínuos. Quanto mais próximo a distância é de um valor 0, mais semelhantes serão os objetos sendo comparados. Essa distância é a generalização da distância de Minkowski escolhendo-se r=2. A equação para distância euclidiana pode ser vista a seguir:

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \tag{40}$$

3.1.3 Distância de Minkowski:

Essa equação representa uma métrica no espaço euclidiano que pode ser considerada uma generalização da distância euclidiana e da distância de Manhattan, dependendo para isso da variação de um parâmetro r. Assim, dependendo do valor de r, obtém-se uma medida diferente da distância entre dois objetos quaisquer pela análise de seus atributos. A distância de Minkowski pode se vista abaixo:

$$d(x,y) = \left(\sum_{k=1}^{n} | x_k - y_k |^r \right)^{\frac{1}{r}} \tag{45}$$

Com r=1, a equação de Minkowski torna-se a distância de Manhattan, em que a distância entre dois pontos é a soma das diferenças absolutas entre suas coordenadas.

Com r=2, tem-se a distância euclidiana.

Com r tendendo a infinito, tem-se uma métrica de distância conhecida por distância de Chebyshev, que resulta na diferença máxima entre os atributos dos objetos. A distância de Chebyshev pode ser vista abaixo:

$$d(x,y) = \lim_{r\to\infty}\left(\sum_{k=1}^{n}| x_k - y_k |^r\right)^{\frac{1}{r}} = \max_{k=1}^{n}| x_k - y_k | \qquad (41)$$

3.1.4 Coeficiente de Correspondência Simples (SMC):

Serve para calcular diferenças entre objetos cujos atributos são binários. Entre atributos binários, as seguintes variáveis podem ser verificadas relativa à comparação entre os atributos par a par:

F_{00} = número de atributos onde tanto x quanto y são iguais a 0(atributos são semelhantes)

F_{01} = número de atributos onde x é 0 e y é 1

F_{10} = número de atributos x é 1 e y é 0

F_{11}= o número de atributos onde x é 1 e y é 1, também se tem como em F_{00}, uma correspondência entre os atributos de x e y.

Com esses resultados, pode-se definir o coeficiente de correspondência simples:

$$SMC = \frac{F_{11}+F_{00}}{F_{00}+F_{01}+F_{10}+F_{11}} \qquad (42)$$

3.1.5 Coeficiente de Jaccard:

Simplificação da correspondência simples para atributos binários assimétricos, onde o número de atributos F_{00} é muito maior ou muito menor do que F_{11}.

Um exemplo é para compras de produtos de uma loja. O número de produtos comprados por cada cliente é muito menor do que os não comprados e analisar esse problema

por SMC produziria resultados muito similares mesmo para clientes com padrões de compra muito distintos.

O coeficiente de Jaccard é descrito abaixo:

$$J = \frac{F_{11}}{F_{01} + F_{10} + F_{11}} \tag{43}$$

3.1.6 Semelhança de Cosseno:

Métrica de distância usada para matrizes de valores esparços, onde o valor da maioria dos atributos é nulo. Ideal para calcular similaridade entre matrizes de freqüências de documentos. Nesse caso o cálculo da similaridade dependerá do número de atributos diferentes de zero. O objetivo é calcular o valor do menor ângulo entre os dois objetos. Assim, se o cosseno entre os dois objetos é 1, eles são completamente semelhantes e o ângulo entre eles é 0. Caso o ângulo entre dois objetos x e y seja 90^o, o cosseno obtido tem valor 0 e eles não tem atributos em comum. Pode-se também normalizar o comprimento entre os dois vetores, de modo que o comprimento máximo de cada atributo seja 1, ou seja, não leva em conta a magnitude dos vetores. A semelhança de cosseno e algumas fórmulas de medidas entre vetores é dada abaixo. A equação 49 é a equação da semelhança de cosseno, já a equação 50 é a semelhança de cosseno normalizada. O produto vetorial entre dois vetores x e y é definido pela equação 51 e a norma de um vetor é dado pela equação 52.

$$\cos(x, y) = \frac{x \bullet y}{\|x\| \cdot \|y\|} \tag{44}$$

$$\cos(x, y) = \frac{x}{\|x\|} \bullet \frac{y}{\|y\|} = x' \bullet y' \tag{50}$$

$$x \bullet y = \sum_{k=1}^{n} x_k y_k \tag{51}$$

$$\|x\| = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{x \bullet x} \tag{52}$$

3.1.7 Correlação de Pearson:

Pode ser usada para vetores de atributos contínuos ou binários. Tenta traçar uma reta com os atributos de dois objetos de dados tal que $x_k = ay_k+b$. Caso a correlação entre os objetos seja +1, os mesmos tem uma correlação linear perfeita positiva e são completamente similares. Caso seja -1, eles tem uma correlação linear negativa perfeita, isto é, não são parecidos e se a correlação for 0, não há relacionamento linear entre os atributos dos objetos, mas pode existir uma dependência não linear a ser investigada por outros meios. A equação de correlação de Pearson é dada a seguir.

$$corr(x,y) = \frac{coVariância(x,y)}{desvioPadrão(x) \cdot desvioPadrão(y)} \tag{45}$$

Os valores de desvio padrão e covariância podem ser calculados a seguir. Antes de se definir o desvio padrão, deve-se calcular uma média dos valores dos atributos dos objetos x e y. A média é definida pela equação 54 abaixo. A equação 55 define o desvio padrão e a equação 56 a covariância de cada atributo para os objetos x e y segundo a média obtida.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{46}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{47}$$

$$coVariância(x,y) = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \bar{x})(y_k - \bar{y}) \tag{56}$$

3.1.8 Utilização de Pesos:

Nem sempre os atributos devem ser tratados igualmente no cálculo da similaridade. Por exemplo, para a classificação de uma pessoa como bom pagador ou mal pagador em um sistema de crédito bancário que analisa o perfil de vários indivíduos, a renda mensal de alguém pode ser um atributo mais importante no cálculo do que o estado civil. Alguns atributos podem, portanto, ser mais importantes que outros para o cálculo da proximidade.

Pode-se então considerar alguma fórmula de proximidade anterior atribuindo um peso ao valor de cada atributo. A fórmula para o cálculo da similaridade pode ser dada pela equação 57.

$$prox\,(x,y) = \sum_{k=1}^{n} \frac{w_k \delta_k f_k(x,y)}{\sum_{k=1}^{n} \delta_k} \qquad (48)$$

Na equação acima, $f_k(x,y)$ denota a similaridade para o k-ésimo valor dos atributos de x e y calculado em termos da função de similaridade escolhida, δ é designado para cada atributo e recebe o valor de 1 (quando não há atributos faltantes ou nulos para uma das instâncias) e 0 caso contrário, w_k é o peso atribuído ao k-ésimo atributo.

3.2 Discretização:

O objetivo primordial da discretização é representar faixas de valores contínuos como intervalos, valores categóricos. Intervalos próximos também podem ser agregados em faixas maiores desde que se preserve a propriedade de ordenação [3].

É usado principalmente quando o algoritmo de mineração de dados sendo considerado só aceita atributos categóricos ou então para melhorar o desempenho do método de mineração mesmo para os métodos que aceitam atributos contínuos. Como um exemplo algoritmos de árvore de indução e de regras devem ter atributos contínuos separados por intervalos, caso contrário o processo de classificação pode se tornar ineficiente .

A escolha de quais são os intervalos contínuos a serem representados como categorias dependerá do critério a ser estabelecido e que tenha um bom desempenho na tarefa a ser feita.

3.2.1 Tarefas da discretização:

A discretização de atributos contínuos envolve, então duas subtarefas, sendo a primeira a definição do número de intervalos considerando a ordenação dos valores do atributo. A segunda diz respeito ao mapeamento de todos os valores de um mesmo intervalo para um valor de categoria. O resultado dos dois passos é, então um conjunto de intervalos $\{(x_0,x_1],(x_1,x_2],...,(x_{n-1}, x_n)\}$, podendo x_0 representar $-\infty$ e $x_{n+\infty}$.

Após esse processo de ordenação e criação de intervalos, passa-se à fase de classificação em si. Dois métodos usualmente adotados para divisão dos valores em intervalos são explicitados abaixo:

3.2.2 Discretização sem Supervisão

A discretização sem supervisão não usa informações sobre classes. Assim, deve-se ter uma maneira de se discriminar quantos intervalos são necessários. Dois métodos comuns são a largura igual e a frequência igual. Na Largura igual, usam-se intervalos de mesma largura especificados pelo próprio usuário. Essa abordagem está condicionada pela visão que o próprio usuário tem do conjunto de dados, podendo levar a avaliações incorretas. Já na frequência igual ou profundidade igual tem-se intervalos sem largura fixa, mas usa-se como parâmetro de criação de intervalos o numero de objetos que se localizam em cada uma das faixas do intervalo. Esse número deve ser necessariamente igual.

3.2.3 Discretização Supervisionada

Na discretização supervisionada usam-se as informações adicionais dos valores das classes de registros. Nesse caso, a divisão dos intervalos pode ser mais bem elaborada, dado que se considera nesse caso cada fração de registros de determinada classe no intervalo considerado. Caso os valores de classes para um mesmo intervalo ocorram com a mesma freqüência, o intervalo será o mais impuro possível (valor 1 de entropia), caso o intervalo tenha apenas valores de uma classe, a entropia é 0. O algoritmo receberá então valores de intervalos ordenados, analisará cada valor como um possível ponto de divisão dividindo os valores em dois intervalos e calculando a entropia de cada intervalo. Cada divisão representa uma partição diferente para o atributo sendo considerado. Pega-se a divisão cuja entropia total seja a menor possível. Após isso, o processo de divisão recomeça com o intervalo dentre os dois considerados que tiver maior entropia (o mais impuro, com mais mistura de classes). O processo continua até que um critério de parada seja satisfeito, por exemplo, um número de intervalos estabelecido pelo usuário foi satisfeito ou quando os intervalos obtidos contiverem um número pequeno de registros e a entropia não melhora com a divisão. A fórmula para o cálculo da entropia de cada intervalo é dada abaixo:

$$e_i = \sum_{i=1}^{k} p_{i,j} \log_2 p_{i,j} \tag{49}$$

Na equação acima, p_{ij} representa a fração de registros da classe j no intervalo i, sendo k o número de rótulos de classe diferentes.

A entropia total (e) da partição para um determinado atributo de divisão será dado por uma média ponderada das entropias individuais dos intervalos produzidos, definido como abaixo, onde w_i representa a fração de instâncias que caem no intervalo i.

$$e = \sum_{i=1}^{n} w_i e_i \tag{50}$$

4 Avaliação de Desempenho e Overffiting de um modelo

4.3 Overfitting de modelo

Inicia-se a discussão de Overffiting considerando os dois tipos de erros encontrados em um classificador: erros de generalização ou erros de treinamento [3] e [30]. Os erros de treinamento são conhecidos por erros aparentes e são caracterizados pelo número de registros mal classificados dentre o total de registros usados para treinar um modelo de classificação. Os erros de generalização são erros cometidos após o treinamento do modelo e são caracterizados por erros cometidos pelo modelo na classificação de registros não vistos anteriormente [3].

É esperado que um bom modelo de classificação tenha um erro de generalização muito baixo ou quase zero e que se adapte bem aos registros de treinamento, mas não deve ser uma cópia dos mesmos, ou seja, pequenos erros de treinamento podem ser tolerados.

Para ilustrar a definição de overffiting, pode-se considerar a ilustração 18.

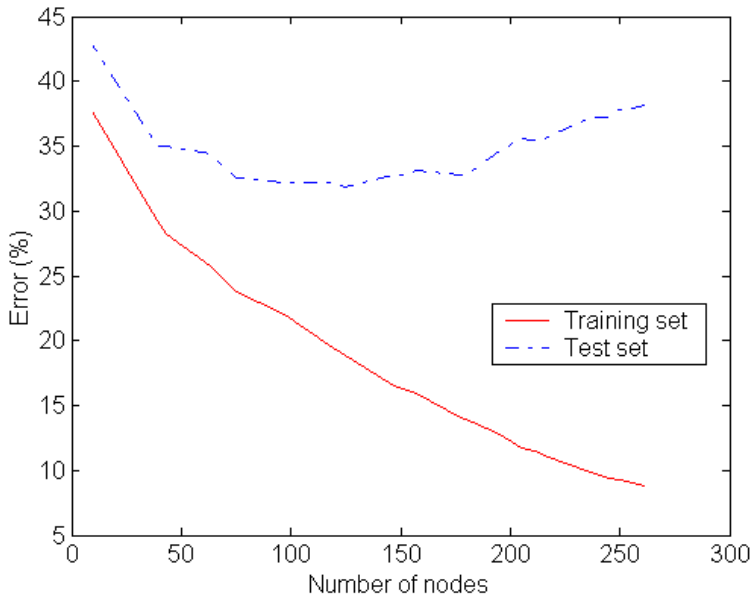


Ilustração 18: Taxas de erro de teste e de treinamento de um conjunto de dados, retirada de [30]

Essa ilustração representa o número de erros de treinamento e teste cometidos por um classificador de árvore de indução conforme aumenta a complexidade da árvore (número de

nodos). No gráfico consideram-se duas classes: uma positiva e outra negativa que possuem 1200 registros cada uma. No experimento acima 30% dos pontos são usados para treinamento com os 70% sendo usados para teste. Usa-se o índice gini como medida de impureza e são aplicados diferentes níveis de poda à árvore inicial.

Os erros de treinamento e teste (ou generalização) começam muito altos para uma árvore com menos de 10 nodos e caem a uma taxa quase linear à medida que o número de nodos aumenta até que a árvore tenha perto de 50 nodos. Desse ponto em diante os erros de generalização aumentam enquanto os de treinamento decrescem até que é alcançado uma taxa de erro de treinamento abaixo de 10% enquanto os erros de teste ultrapassam os 40%.

O comportamento assumido pela árvore com poucos nodos e altas taxas de erro de treinamento e teste chama-se underfitting de modelo e ocorre porque com poucas instâncias de treinamento usadas para treinar o modelo, nem todos os valores de atributos estão caracterizados. O resultado é um fraco desempenho de treinamento e teste.

Com uma árvore com muitos nodos (isso ocorre no exemplo acima quando a árvore ultrapassa os 50 nodos), o modelo se torna cada vez mais adequado aos dados de treinamento até que eles sejam perfeitamente adequados aos dados de treinamento (ou uma cópia dos atributos dos mesmos que indicam o rótulo particular de cada registro). No entanto, a generalização para registros não vistos, com combinações diferentes de atributos tende a ser ruim, pois alguns registros da árvore podem caracterizar valores de atributos incomuns nos dados. Quando essa presença de ruído nos dados de treinamento se torna grande demais, pode produzir generalizações errôneas. Essa é a característica do overffiting de modelo: baixas taxas de erros de treinamento mas o modelo generaliza mal para instâncias não vistas. Analisam-se a seguir algumas causas do overfitting.

4.1.1 Overfitting devido a ruído:

O ruído ocorre devido a registros cujos valores de atributos ou classe estão incorretos. Como um exemplo, são analisadas duas tabelas relativas a registros de treinamento e teste para o problema da classificação de mamíferos em que as instâncias morcego e baleia são classificadas erroneamente como não mamíferos. Os modelos e tabelas discutidos nessa seção foram obtidos de [3]. Uma árvore que se adapta perfeitamente ao conjunto de treinamento é dada pelo Modelo 1 da ilustração 19. Embora o erro de treinamento seja 0 nesse modelo, o

erro de generalização considerando o conjunto de teste é de 30%. Isso ocorre porque os valores de atributos de humanos e golfinhos são iguais aos valores para baleia. O tamanduá com espinhos do conjunto de teste receberá pelo modelo 1 a classe equivocada de não mamífero pois, no conjunto de treinamento, todas as instâncias que não originam seus filhotes são classificados como não mamíferos.

O modelo de árvore 2 tem uma taxa de erro de treinamento maior (não classifica corretamente morcegos e baleias, instâncias com classes erradas) correspondente a 20%, mas generaliza melhor considerando o conjunto de teste, apenas erra a classe do tamanduá. Então, como ilustrado, a árvore do modelo 1 tem problema de overfitting devido a ruído.

Tabela 3: Conjunto de Treinamento para classificar mamíferos

Nome	Temperatura	Origina	Quatro Patas	Hiberna	Rótulo
porco-espinho	sangue quente	sim	sim	sim	sim
gato	sangue quente	sim	sim	não	sim
morcego	sangue quente	sim	não	sim	não(Errado!)
baleia	sangue quente	sim	não	não	não(Errado!)
salamandra	sangue frio	não	sim	sim	não
dragã de komodo	sangue frio	não	sim	não	não
píton	sangue frio	não	não	sim	não
salmão	sangue frio	não	não	não	não
águia	sangue quente	não	não	não	não
peixe guppy	sangue frio	sim	não	não	não

Tabela 4: conjunto de teste para classificar mamíferos

Nome	Temperatura	Origina	Quatro Patas	Hiberna	Rótulo
humano	sangue quente	sim	não	não	sim
pombo	sangue quente	não	não	não	não
elefante	sangue quente	sim	sim	não	sim
tubarão	sangue frio	sim	não	não	não
tartaruga	sangue frio	não	sim	não	não
pinguim	sangue frio	não	não	não	não
enguia	sangue frio	não	não	não	não
golfinho	sangue quente	sim	não	não	sim
tamanduá	sangue quente	não	sim	sim	sim
monstro de gila	sangue frio	não	sim	sim	não

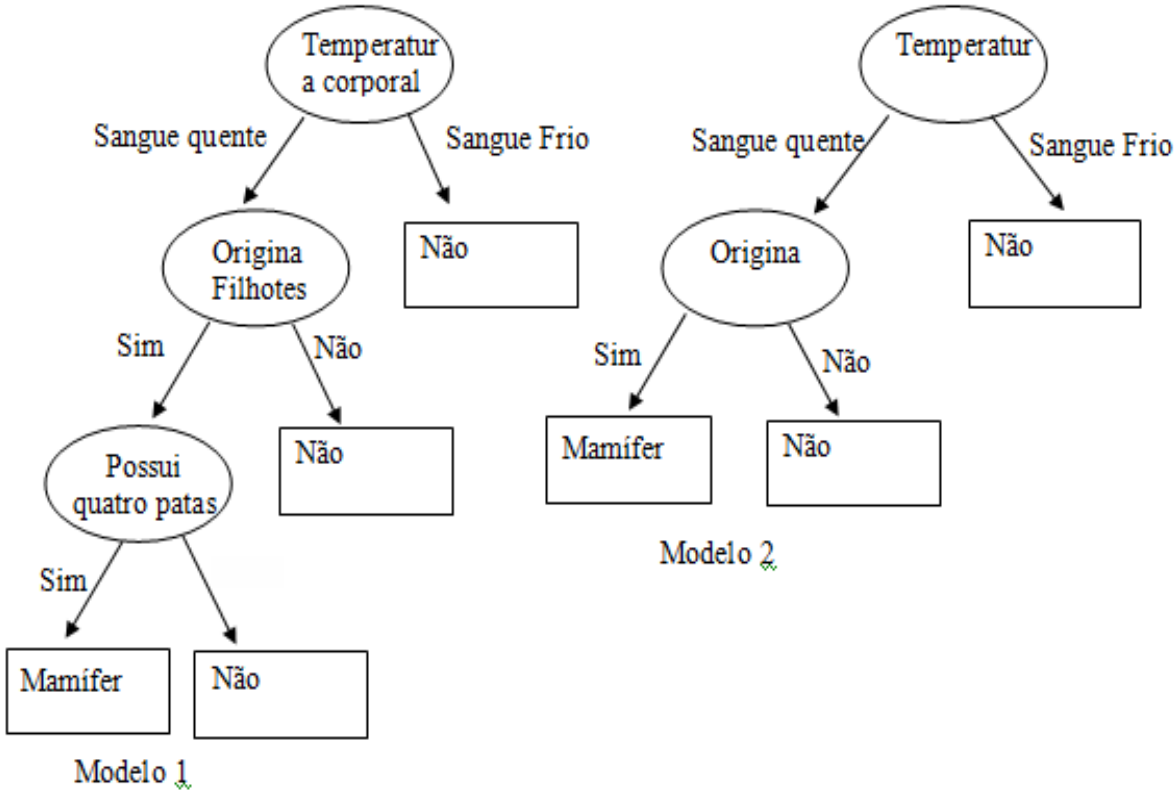


Ilustração 19: Dois modelos de classificação construídos a partir dos registros da tabela 3

4.1.2 Overfitting devido a número pequeno de registros:

Nesse caso, o modelo falha pela falta de exemplos significativos dos grupos principais de classes no conjunto de treinamento. O resultado é a classificação equivocada de vários registros de teste dos grupos principais. O exemplo disso é ilustrado a seguir:

Tabela 5: Poucos dados de treinamento para classificar mamíferos

Nome	Temperatura	Origina	Quatro Patas	Hiberna	Rótulo
salamandra	sangue frio	não	sim	sim	não
peixe Guppy	sangue frio	sim	não	não	não
águia	sangue quente	não	não	não	não
poorwill	sangue quente	não	não	sim	não
platypus	sangue quente	não	sim	sim	sim

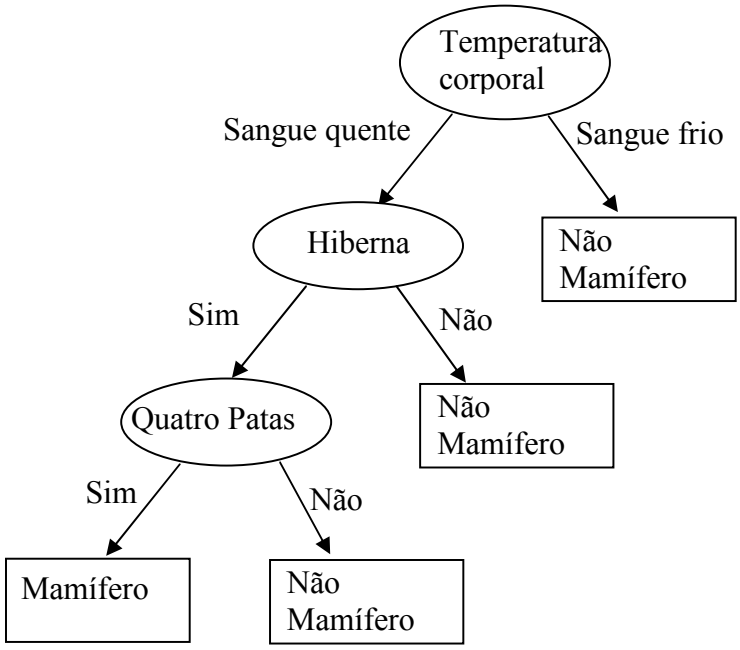


Ilustração 20: Modelo de classificação construído a partir da tabela 5

No exemplo da árvore montada acima, todos os principais grupos de mamíferos são classificados de forma errada como não mamíferos (como por exemplo humanos e golfinhos). Pois o único representante dos mamíferos nos registros de treinamento é um animal chamado "platyplus"que hiberna, tem quatro patas e não origina seus filhotes. Mas a quase totalidade dos mamíferos não hiberna, e origina seus filhotes. Os registros de treinamento adequam-se bem à árvore montada (o erro de treinamento novamente é zero), mas o representante

mamífero do grupo, além de ser um caso raro de mamífero que foge à regra é o único representante da qual se faz a classificação. Novamente se tem overfitting nesse caso.

4.2 Avaliação do desempenho de classificadores

Algumas medidas para se evitar o overffiting são: não permitir que o modelo se adeque perfeitamente aos dados do treinamento e avaliar o desempenho de vários modelos considerando diferentes parâmetros de treinamento [9], [8] e [25]. No caso de árvores de indução, por exemplo, seria interessante ter alguma métrica para avaliar o desempenho dessas várias árvores sobre o mesmo conjunto de treinamento mas com diferentes níveis de poda e considerando os erros de generalização dos classificadores, pois como visto na seção de Overffiting, a avaliação apenas dos erros de treinamento é uma medida pobre de avaliação de modelos. Do mesmo modo, em classificadores SVM pode-se analisar o desempenho de vários modelos sobre o mesmo conjunto de treinamento, no entanto com a consideração de várias funções de kernel e valores de custo C associados. Considerar mais exemplos de treinamento também seria interessante para diminuir a influência de ruidos. Intuitivamente também pode-se afirmar que quanto maior o conjunto de teste, mas exata a estimativa do erro. A visão de um modelo mais simples que não se adeque perfeitamente aos dados de treinamento também é referido como Lâmina de Occam [7], que apresenta uma visão intuitiva: “Dados dois modelos que consideram o mesmo conjunto de treinamento e que tem mesmo erro de generalização, o mais simples é preferível”.

A seguir são listadas algumas técnicas de como avaliar o desempenho de dois classificadores.

4.2.1 Hold-out

Nessa estratégia, separa-se o conjunto de dados originais em treinamento e teste (cerca de 2/3 dos dados para treinamento e o restante para teste).

Para uma idéia mais clara dessa estratégia, considera-se a ilustração 21 dada abaixo obtida de [30], onde diferentes modelos classificatórios são obtidos por meio de funções lineares, quadráticas ou por regressão linear não-paramétrica pela junção dos “pontos de

dados" sob um mesmo conjunto de validação e treinamento. A inferência da função ou modelo que tem a taxa de acerto mais aproximada é feita por essas regressões, determinando o quanto o ponto de teste está distante da função considerada. Por simplicidade, são mostrados 10 pontos representando os dados, onde sete deles são separados para treinamento (montagem da função de regressão) e os restantes serão usados como um conjunto de teste:

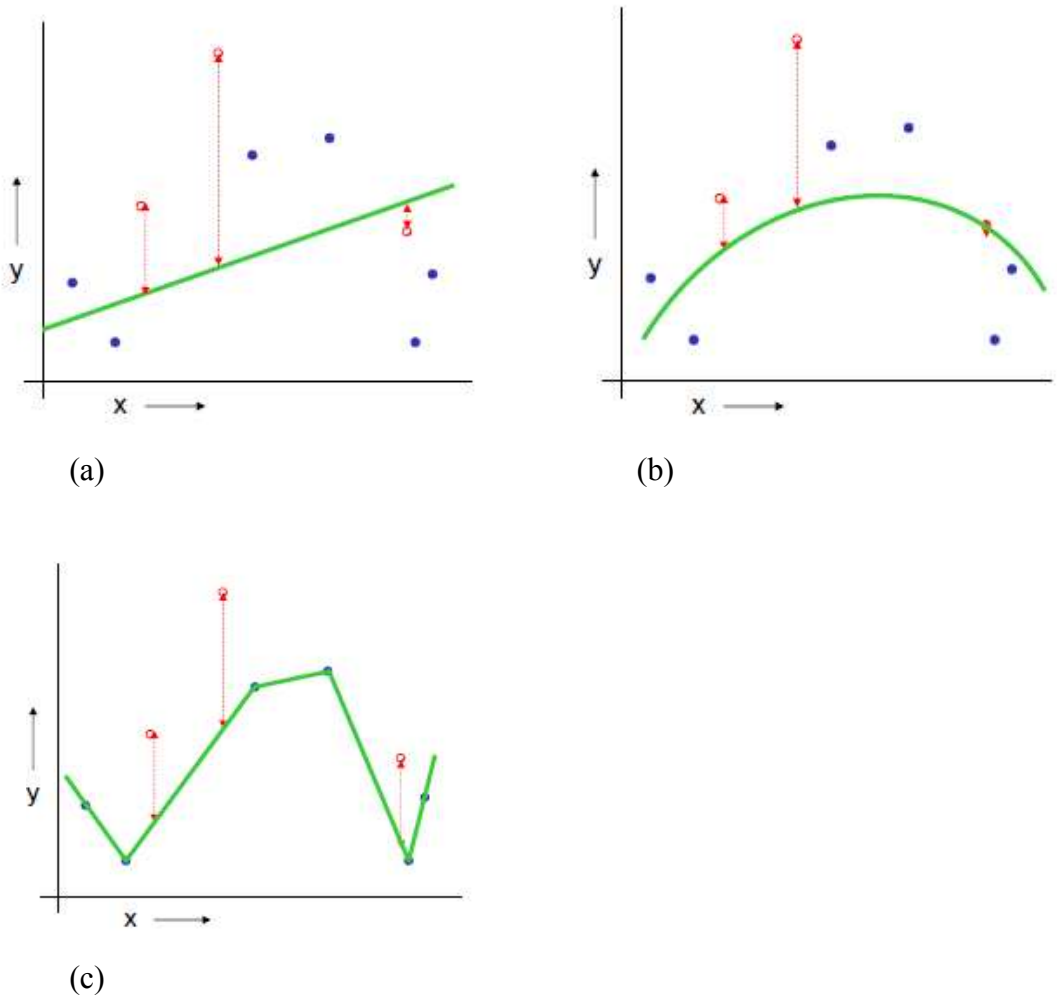


Ilustração 21: Separação de valores em treinamento e teste. (a) Regressão linear, (b) Regressão quadrática e (c) Por junção de pontos. Disponível em [30]

Na ilustração 21, o desempenho de cada função acima treinada com 7 de 10 pontos é obtido pela análise do quão distantes os três pontos restantes estão da função f. Assim, a média do erro quadrático principal medido para a função linear é de 2.4, para a função quadrática o erro é de 0.9 e para a junção de pontos o erro é de 2.2. A junção de pontos significa um modelo que se adequa perfeitamente aos dados de treinamento e é por isso sujeito a overfitting de modelo como explicado, generaliza mal para objetos ainda não vistos. Com a análise acima, infere-se que o modelo mais adequado é aquele de menor erro

quadrático, cuja generalização para pontos não vistos na formação da função apresenta melhor aproximação da mesma, portanto, o modelo delineado pela função quadrática é o melhor nesse caso. O erro médio quadrático pode ser calculado como:

$$EMQ = \frac{1}{n}\sum_{i=1}^{n}(x_{i[est]} - x_{i[real]})^2 \quad (60)$$

Na equação 60, $x_{i[est]}$ é o valor estimado para a variável x_i e $x_{i[real]}$ seu valor real sendo n o número de instâncias preditas.

É também usual para classificadores que inferem um valor para a classe de cada uma das instâncias de teste, o cálculo a acurácia e da taxa de erros desse classificador:

$$accuracy = \frac{acertos}{n_{teste}} \quad (61)$$

$$tErros = \frac{erros}{n_{teste}} \quad (62)$$

Nas equações acima, erros é o número total de erros cometidos no conjunto de teste, acertos é o total de acertos do conjunto de teste e n_{teste} é o número total de instâncias de teste.

O hold-out pode apresentar problemas com um número menor de registros já que menos registros serão usados para treinamento. Os modelos podem ser restritivos e se os dados forem formados por muitos pontos de ruído, pode-se ter overffiting.

4.2.2 Hold-out repetido

Uma outra estratégia é repetir o hold-out para diferentes sub-amostras, o que também pode ser chamado de Sub-amostragem Aleatória. Nesse caso os registros selecionados para treinamento e para teste mudam a cada execução da sub-amostragem, sendo a precisão geral dessa sub-amostragem dada pela equação 63.

$$P_G = \sum_{i=1}^{k}\frac{P_i}{k} \quad (63)$$

Na equação acima, Pg indica a precisão geral do método de sub-amostragem que é dado pelo somatório das precisões calculadas para cada iteração i (a precisão na iteração é Pi), dividindo pelo total de iterações k. É uma maneira mais confiável de se avaliar o desempenho de vários classificadores sobre o mesmo conjunto de treinamento e teste a cada iteração. Porém o método não provê uma maneira confiável para saber quais registros já fizeram parte de teste e treinamento e isso implica que um mesmo registro pode ser usado muitos vezes seguidas para teste.

4.2.3 Validação Cruzada

A validação cruzada é uma alternativa ao Hold-out, podendo ser feita de diversas maneiras. Serão explicadas três formas bem conhecidas.

Uma delas é a divisão dos conjuntos para treinamento (conjunto 1) e para teste (conjunto 2) com 50% dos registros originais para cada um. Primeiro o desempenho de cada um dos classificadores é avaliado através da montagem do modelo com o primeiro conjunto, e a avaliação do modelo é feita com o segundo conjunto. Na segunda etapa o conjunto que era usado para treinamento vira o de teste e o que era usado para teste será agora o de treinamento e a avaliação do desempenho dos classificadores é novamente calculada. A taxa de erro para um determinado classificador é dado pela média dos erros de ambas as execuções. Essa abordagem é denominada validação cruzada de duas partes.

Em uma segunda abordagem denominada *leave one out*, o procedimento de avaliação é repetido várias vezes com um dos registros apenas sendo deixado para teste e todos os demais sendo usados para treinamento. Assim, caso sejam usados 10 registros no total, serão feitas dez execuções dessa validação cruzada, onde cada um dos registro é usado para teste exatamente uma vez. Tem a vantagem de maximizar os registros de treinamento e verificar mais facilmente anomalias nos dados no caso de um registro fugir à regra daqueles que possuem a mesma classe. Sua desvantagem reside na alta variância que pode ser obtida nos teste e a execução ser computacionalmente custosa (se forem 1000 registros, serão 1000 iterações, caso sejam 10000, serão 10000 iterações, etc). Nesse caso o erro geral novamente é obtido pela média dos erros de todas as execuções conforme mostrado anteriormente. Esse

procedimento é ilustrado abaixo para o caso do classificador linear e mesma disposição de registros vista na ilustração 21.

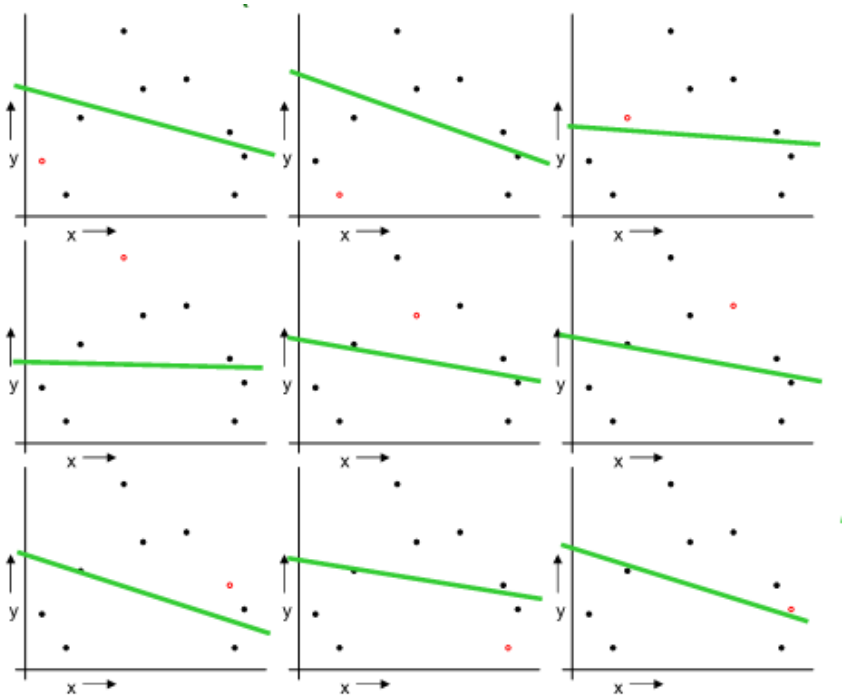


Ilustração 22: Leave one out cross validation disponível em [30]

A terceira e última abordagem é denominada de validação cruzada de k partes. Nessa abordagem, divide-se o conjunto de dados original em subconjuntos de tamanho igual. No exemplo dado pela ilustração 23 os dados são divididos em três partes contendo três instâncias cada. Durante cada iteração da avaliação de desempenho, um dos grupos é escolhido para teste enquanto os demais são usados para treinamento. Sendo k o número de conjuntos obtidos, o experimento é realizado k vezes o que significa que cada partição é usada para teste apenas uma vez e assim, cada um dos registros é também testado apenas uma vez, não havendo o problema do registro ser usado mais de uma vez para teste que ocorria com o holdout. A média dos erros é avaliada para cada classificador como anteriormente explicado e o classificador com menos erros de generalização será o mais adequado ao problema em questão. Então no procedimento da ilustração 23, as instâncias são separadas em três grupos: vermelhos, verdes e azuis, sendo k=3. Primeiramente o classificador é treinado com todos os pontos menos os vermelhos (curva vermelha). O modelo obtido é testado nos vermelhos e o erro médio quadrático é computado. O mesmo ocorre para os demais conjuntos azuis e verdes. No final o classificador com o menor erro é eleito o melhor para o problema em questão.

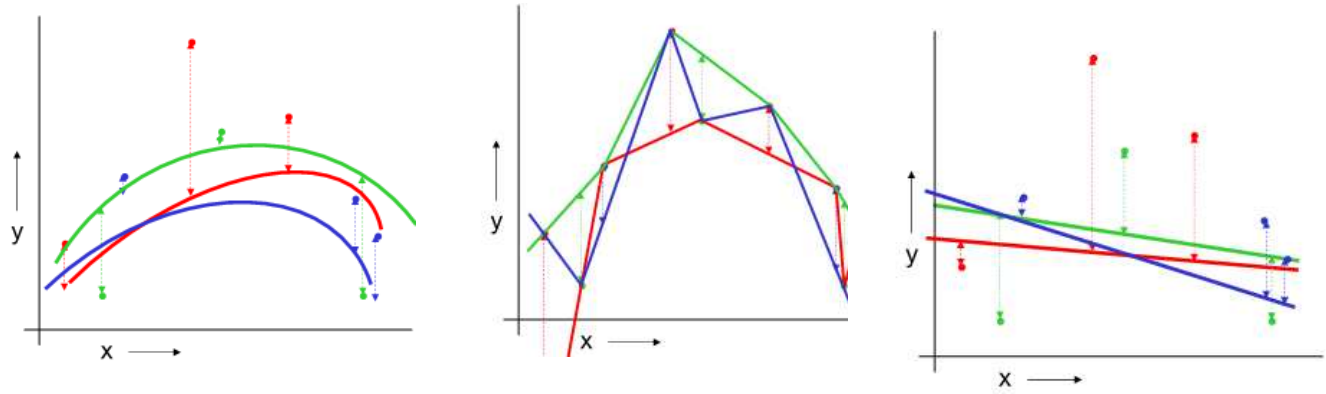


Ilustração 23: K-fold cross validation onde k=3, disponível em [30]

Na ilustração acima, o erro encontrado para a média quadrática das instâncias de teste nas três execuções do algoritmo da validação cruzada de k-partes resulta em um erro de 2.05 para o classificador linear, 1.11 para o classificador quadrático e 2.93 para o classificador que tem erro de treinamento 0 (dado pela junção de todos os pontos que representam instâncias de treinamento em uma determinada iteração da validação cruzada).

Portanto o classificador mais adequado é aquele representado pela regressão quadrática pois generaliza melhor (tem menos erros de teste, os pontos de teste ficam mais próximos da função considerando a média de todas as iterações).

4.2.4 Avaliação por uma Matriz de confusão

As abordagens anteriores não consideram os custos associados aos erros cometidos por um classificador para cada uma das classes. Dependendo da natureza do problema isso pode ser necessário. Por exemplo: em um programa que detecta cogumelos comestíveis e não comestíveis, pode ser interessante considerar que, se o programa classificar um cogumelo comestível como não comestível é um erro menos grave do que o oposto: classificar um não-comestível como comestível. Pode-se, portanto atribuir um peso para cada classificação equivocada de instâncias relativas à classe real das mesmas. No exemplo dado pode-se considerar que a associação de um cogumelo não comestível como comestível terá um peso 3 na computação dos erros. O programa poderá associar peso 1 no caso oposto.

Nesse sentido cria-se uma matriz denominada matriz de confusão que mostra o número de classificações corretas em oposição às classificações preditas para cada uma das classes. Cada uma das classificações erradas pode ser associada a uma matriz de pesos, por exemplo e a taxa de erros pode ser computada considerando essa matriz. Abaixo encontra-se uma matriz de confusão para classificação binária junto com algumas medidas que podem ser definidas através da matriz de confusão:

Tabela 6: Matriz de confusão

Classe	Preditos como C+	Preditos como C-	Precisão da classe	Recall	Precisão Total
C+	Tp	Fn	$\frac{T_p}{T_p + F_n}$	$\frac{T_p}{T_p + F_p}$	$\frac{T_p + T_n}{n}$
C-	Fp	Tn	$\frac{T_n}{F_p + T_n}$	$\frac{T_n}{T_n + F_n}$	

Na tabela acima, T_p é o número de exemplos corretamente classificados como positivos e F_p é o número de exemplos erroneamente classificados como positivos. De maneira similar definem-se os outros. Na tabela, a precisão da classe indica dentre os elementos classificados como positivos, uma taxa de acerto. Da mesma forma define-se a precisão da classe negativa. O Recall da classe positiva indica a porcentagem dos elementos efetivamente positivos dentre todos aqueles classificados como positivos. De maneira análoga define-se o Recall da classe negativa. Finalmente a precisão total define a acurácia do classificador com $n = T_p+F_n+F_p+T_n$.

5 RESULTADOS DO EXPERIMENTO

5.1 Introdução

Fontes de dados escondidas são bancos de dados subjacentes aos sites Web que podem ser acessíveis via interfaces de consulta disponíveis na Web. Madhavan et al.[2] estimam que devem existir cerca de 25 milhões de fontes de dados escondidas na Web. A grande quantidade de fontes, a disponibilidade imprevisível e as constantes diferenças entre os esquemas de dados requerem técnicas automáticas e dinâmicas de integração.

Mediadores de consultas são aplicações que promovem a integração de fontes de dados. Eles devem ser capazes de distribuir consultas de usuários entre diversas fontes de dados e mesclar os resultados obtidos de modo a permitir uma visão integrada e uniforme das fontes consultadas.

Nesse contexto, um dos problemas a serem resolvidos é a distribuição da consulta original entre fontes de dados disponíveis. Sem dúvida, devido ao grande número de fontes torna-se necessário selecionar, dentre todas, aquelas que mais provavelmente contém dados relevantes. Uma possível estratégia poderia ser classificar o conteúdo das fontes por domínio de informação e selecionar somente aquelas que coincidem com o domínio de informação da consulta formulada.

O experimento apresentado nessa seção visa explorar métodos de classificar o conteúdo das fontes de dados escondidas na Web.

5.2 Método

Métodos de classificação de fontes Web podem ser de dois tipos, aqueles que sondam o conteúdo das fontes de dados e aqueles que utilizam características aparentes dos sites que hospedam as fontes de dados. Nesse trabalho busca-se explorar os métodos do primeiro tipo.

Como conjunto de dados de teste utiliza-se o *UIUC Web integration repository* [13] que foi produzido especificamente para classificação de bancos de dados. Esse repositório é encontrado em grande número de trabalhos com deep web [8], [25], [32] e [33], o que mostra que o dataset é reconhecido pela comunidade acadêmica. Ele contém dados extraídos de 447 sites Web classificados em oito categorias diferentes, tais como, automóveis, livros, hotéis,

etc. Os dados consistem de páginas HTML e resultados de consultas submetidas às interfaces de busca de cada site. A vantagem de utilizar esse conjunto de teste é que ele é largamente utilizado, o que permite a comparação de resultados.

Escolheu-se utilizar a técnica Support Vector Machine (SVM) de classificação, pois apresenta resultados na maioria das vezes superiores a outros classificadores na categorização de informações textuais, de acordo com alguns autores como [5], [8], [9], [10], [17], [25]. Segundo essa técnica os objetos a serem classificados devem ser representados por vetores de características (seção 2.6). Cada dimensão do vetor representa uma característica diferente do objeto e cada coordenada representa uma medida da característica associada ao objeto por algum critério. Sendo assim, pode-se representar os sites de várias formas diferentes como, por exemplo, pelo conjunto de palavras que ocorrem nas páginas HTML capturadas. Cada dimensão representa um peso associado a uma palavra diferente e as coordenadas representam a freqüência com que cada palavra ocorre no site. Essa representação provém da intuição de que domínios de informação diferentes possuem conjuntos de palavras característicos, como por exemplo, o jargão utilizado em medicina, engenharia, comércio de livros, filmes, etc.

A representação anterior, no entanto, pode apresentar um problema. A frequência com que a palavra ocorre no site pode não determinar nenhuma correlação com a categoria à qual ele pertence. Pode, sim, estar relacionada com a quantidade de páginas capturadas. Pode-se, então, utilizar uma variação da representação anterior onde as coordenadas são valores binários correspondentes a presença ou não da palavra no site.

Pode-se ser ainda mais específicos quanto à medida associada às características dos sites. Pode-se recorrer às técnicas de recuperação de informação [6] que representam documentos e consultas por vetores de tokens e onde as coordenadas são calculadas segundo o *term frequency – inverse document frequency* (TF-IDF) de cada token. O TF-IDF exprime a intuição de que quanto mais frequente um token em um documento, mais discriminador é o token para o documento e quanto mais frequente um token no conjunto de todos os documentos menos discriminador é o token para o documento. O cálculo do peso de cada token em um documento é, portanto, uma medida relativa ao balanceamento entre esses dois fatores discriminados acima e a definição de cada fator é dado pela análise das equações 64, 65 e 66.

$$f_{i,j} = \frac{frequency_{i,j}}{\max(frequency_{i,j})} \tag{64}$$

Na equação acima, $frequency_{i,j}$ é a freqüência do termo k_i no documento DOC_j. O denominador refere-se à normalização realizada onde divide-se a freqüência do termo considerado pela freqüência máxima do termo encontrado no documento. Assim, quanto mais próxima é a freqüência do termo corrente comparada à máxima, mais próximo $f_{i,j}$ será de 1.

O fator IDF calculado sobre cada termo de cada documento é dado abaixo:

$$idf_i = \log \frac{N}{n_i} \tag{51}$$

Na equação do fator IDF, N é o total de documentos do sistema e n_i o número de documentos no qual o termo de índice k_i aparece. Esse é dito um fator de dissimilaridade entre um conjunto de documentos relevantes e o restante. Assim, se um token ocorre em um número muito grande de documentos, n_i se tornará aproximadamente igual a N e o IDF será próximo de zero. Isso indicará que o token não agrega valor na caracterização dos documentos.

O cálculo do peso de um determinado token i para o documento j será, por sua vez uma combinação entre esses dois fatores:

$$weight_{i,j} = TF \cdot IDF \tag{66}$$

Por fim, se pode indagar se as páginas HTML de cada site são suficientes para classificá-lo ou se são necessários os dados extraídos dos bancos de dados subjacentes às fontes. Todas essas possibilidades foram exploradas no experimento apresentado nessa seção.

5.3 Execução

Para a realização do experimento utilizou-se a linguagem Java para percorrer o dataset citado e realizar as tarefas de classificação. Em um primeiro momento separa-se 75% das fontes de dados de cada domínio para serem utilizadas no treinamento do modelo (dataset de treinamento). Os diretórios restantes representativos de cada domínio de site (25% restantes) serão usados como um dataset de teste para testar o modelo gerado no treinamento.

Cada site é representado por um diretório e está associado a uma categoria ou domínio, como explicado e é composto por páginas HTML oriundas de dois diretórios: homecached e interfacecached.

Para realizar as classificações, é necessário o percurso de cada um desses diretórios e para cada site apenas a pasta homecached ou apenas a pasta interfacecached ou ambos. Essa opção é configurável através do programa em Java.

Para cada página pertencente a um site devem ser obtidos termos ou tokens representativos para se realizar a classificação pelo domínio daquele site.

Nesse sentido, utilizou-se a biblioteca Java Jericho HTML Parser [14], o qual permite manipulação de partes de um documento HTML, assim como identificação de tags e extração de termos entre tags. Nos experimentos foram obtidos termos da tag Body das páginas. No entanto nem todos os termos obtidos agregam significado relevante para categorizar fontes de dados.

Após a obtenção dos tokens uma lista de stop words pode ser usada para ignorar termos irrelevantes como artigos, pronomes, advérbios, interjeições, etc. Além disso para todos os testes realizados se o termo é um verbo ou substantivo poderá ser necessário lematizá-lo, ou seja, considerar apenas seu radical (parte invariável do verbo ou substantivo e que exprime a idéia geral do termo). Informações numéricas (como tais como anos, número de clientes etc) também serão desconsideradas. A lematização e remoção de stop words é realizada através da framework Java Dragon Toolkit [15].

Após o percurso de cada uma das páginas de um site é necessário a realização da obtenção dos pesos de cada um dos tokens por uma das técnicas já citadas (Vetorial TF-IDF ou Binário). O resultado dos pesos gerados para cada um dos tokens de cada site será armazenado em memória.

Após a obtenção dos pesos de todos os tokens de cada site é necessário a escrita de um arquivo texto (para treinamento) que possa ser entendido pela biblioteca libsvm [16], que tem por objetivo fornecer um conjunto de funções e executáveis para se realizar tarefas de classificação por Support Vector Machine (SVM). Nesse caso, usa-se 75% das fontes de cada domínio para o treinamento do modelo e o restante para predição do modelo. A idéia é que um bom modelo de classificação terá grande capacidade de generalização para fontes de dados não vistas no treinamento do modelo.

Nos experimentos também variou-se um conjunto de funções de kernel para cada modelo gerado. A função de kernel é necessária ao SVM de modo a realizar o kernel trick (como explicado na subseção 2.6) provendo uma separação entre instâncias de classes diferentes que de outro modo não seriam separáveis linearmente. Nesse sentido, um número muito grande de funções de kernel podem ser usadas e devido à natureza n-dimensional do problema de classificação não se sabe de antemão qual será o melhor kernel para o problema. Ainda é de conhecimento que somente alguns poucos kernels tem tido desempenho satisfatório para um grande número de problemas [17],[9]. Utilizaram-se três desses kernels, a saber: kernel sigmoidal, kernel RBF e o kernel linear.

Tabela 6: Funções de kernel utilizadas

Kernel	Expressão Parâmetros	Parâmetros
Linear	$K(x_i, x_j) = x_i \cdot x_j$	
RBF	$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\left\| x_i - x_j \right\|^2\right)$	σ^2
Sigmóide	$K(x_i, x_j) = \tanh\left(-\frac{1}{2\sigma^2}(x_i \cdot x_j) + \beta_1\right)$	β_0, σ^2

Na tabela acima, cada x_i e x_j correspondem aos atributos das instância i ou j onde i é diferente de j

Além da função de kernel, um fator de penalidade C deve ser arbitrado para impedir um erro de treinamento muito grande. Também variou-se um fator gamma tal que gamma vale $-\frac{1}{2\sigma^2}$ para os kernels RBF e Sigmóide.

Considerou-se a realização de 10-fold cross validation para cada combinação de vários pares de parâmetros C e gamma considerando cada um dos kernels utilizados e cada configuração de treinamento (geração de pesos e local de extração). O par (C,gamma) com a melhor percentagem de cross validation é escolhido e o modelo de classificação será treinado a partir dessa configuração. Com o modelo treinado, ele será usado para predizer a categoria dos sites do dataset de teste anteriormente separado. Em ambos os casos, uma percentagem será obtido relativa a accuracy de 10-fold cross validation e de predição. O intervalo de

valores para geração de pares de (C,gamma) considerado foi ($C = 2^{-5},2^{-3}, \ldots , 2^{15}$) e ($gamma = 2^{-15},2^{-13}, \ldots , 2^{3}$) . Considerou-se uma combinação de todos os valores de C e gamma desses intervalos. Tem-se boas chances de encontrar um par (C, gamma) com acurácia elevada levando em conta esses intervalos para uma variedade de aplicações segundo [17].

Além disso o framework libsvm foi usado com uma estratégia de um-contra-restante para inferir a percentagem de cross validation ou predição para cada modelo gerado (já que o SVM apenas suporta classificação binária) com estimativa de probabilidade, ou seja, a classe com a melhor percentagem de acertos para uma dada instância será considerada a classe predita para a instância.

5.4 Resultados

Os resultados de todos os testes realizados para cada combinação de parâmetros de treinamento considerando os melhores valores de (C,gamma) podem ser vistos na tabela abaixo em ordem decrescente de desempenho de cross validation:

Tabela 8: Resultados dos testes realizados

Tipo de Kernel	Gamma	Custo	Origem Tokens	Método de Representação	Desempenho Cross	Desempenho Predição
Linear	Não usado	0,031	interfacecached	VETORIAL TF-IDF	85,95%	76,47%
RBF	3,05E-05	512	interfacecached	VETORIAL TF-IDF	84,48%	78,15%
Sigmóide	0,03125	2	interfacecached	VETORIAL TF-IDF	83,99%	78,43%
Linear	Não usado	0,031	homecached	VETORIAL TF-IDF	83,66%	78,43%
RBF	4,88E-04	32	homecached	VETORIAL TF-IDF	82,99%	77,31%
Sigmóide	2	0,125	homecached	VETORIAL TF-IDF	82,69%	77,31%
Linear	Não usado	0,125	homecached e interfacacached	VETORIAL TF-IDF	82,18%	72,90%
RBF	1,22E-04	512	homecached e interfacacached	VETORIAL TF-IDF	80,86%	71,03%
Sigmóide	0,5	0,5	homecached e interfacacached	VETORIAL TF-IDF	79,54%	71,03%
Sigmóide	3,05E-05	512	homecached e interfacecached	BINARIO	74,63%	68,07%
RBF	3,05E-05	2048	homecached e interfacecached	BINARIO	74,33%	63,87%
Linear	Não usado	0,5	interfacecached	BINARIO	74,18%	68,63%
RBF	3,05E-05	8192	interfacecached	BINARIO	74,18%	67,65%
Linear	Não usado	0,125	homecached e interfacecached	BINARIO	74,03%	64,71%
Sigmóide	4,88E-04	512	interfacecached	BINARIO	73,86%	65,69%
RBF	3,05E-05	2048	homecached	BINARIO	67,00%	59,81%
Sigmóide	3,05E-05	2048	homecached	BINARIO	66,67%	59,81%
Linear	Não usado	0,125	homecached	BINARIO	66,34%	59,81%

A tabela acima nos mostra alguns resultados interessantes. Em primeiro lugar, a consideração dos pesos como vetores nos proporcionou a obtenção de melhores resultados do que qualquer outra configuração com pesos binários, o que mostra a validade da afirmação de Salton (1988) [19], que reconhece a limitação de pesos binários. Com relação ao método de representação Vetorial TF-IDF, os melhores resultados foram obtidos quando a extração considerou apenas a pasta interfacecached (resultados de busca de palavras-chave) dos sites, o

que demonstra que a categoria de um site pode ser inferida por um classificador apenas considerando os resultados retornados das interfaces de busca do site. A extração de termos apenas da homepage tem resultados ligeiramente inferiores para o método vetorial e a consideração da extração de termos de ambos faz com que os resultados da accuracy obtida por cross validation caiam em até 6% e para predição utilizando um dataset de teste até 7%.

Nesse sentido a utilização dos kernels Linear, RBF ou Sigmóide são muito parecidos em termos de resultados como pode ser visto pelas melhores porcentagens de predição e cross validation no topo da tabela. Além disso o pior resultado de cross validation produzido pela representação vetorial tem uma accuracy de cerca de 4% a mais que o melhor resultado da representação binária. Além disso, para o modelo de representação binário a extração de termos apenas das interfaces de busca ou de ambos (homepage e interface) produz resultados muito similares. A queda de accuracy (tanto da predição quanto da cross validation) é mais evidente quando se considera a extração de tokens apenas das páginas da homepage nesse caso.

Portanto mostra-se pelos resultados a superioridade de desempenho tanto de cross quanto de predição para o modelo vetorial TF-IDF com um desempenho de cross de cerca de 86% e de predição de 77% para o kernel linear. Mas considerando as mesmas configurações de Método de Representação e Origem dos Tokens para treinamento e a variação das funções de kernels, as mesmas terão desempenho muito similar entre si.

6 Conclusões e Trabalhos Futuros

Nesse trabalho foi considerado um estudo da classificação de documentos por máquinas de vetores de suporte (SVM). Foi feito um levantamento da parte dos sites que melhor caracterizam um conjunto de documentos e ainda uma comparação entre um dos modelos de representação de pesos mais utilizados: Vetorial TF-IDF com os pesos binários verificando a superioridade do modelo vetorial na classificação de domínios web. Variaram-se três dos kernels mais utilizadas em pesquisas recentes para cada caso estudado e foi verificado que todos os kernels utilizados para ambos os modelos Vetorial TF-IDF ou binário considerando as mesmas configurações de Método de Representação e Origem dos Tokens proporcionam um desempenho muito similar.

Tendo em vista os bons resultados alcançados com esse experimento, relativo à comparação com outros trabalhos, como [8],[25] e [32] e que o dataset utilizado no trabalho é um dataset antigo (de maio de 2003) e também que muitas referências de sites nele encontrados não existem mais (é um dataset não mais atualizado), seria de interesse obter um dataset mais atualizado, cujas urls encontradas nos sites pudessem ser também consideradas no cálculo da classe do site de modo a aumentar a acurácia do método.

Também seria de interesse a classificação multiclasse, ou seja, classificar sites por categorias e subcategorias, como uma maneira de refinamento de buscas por esses diretórios categorizados de sites, atribuindo à consultas posteriormente realizadas maior poder de representação.

7 REFERÊNCIAS

1. Hagedorn, K., & Santelli, J. (2008). Google still not indexing hidden web URLs. *D-Lib Magazine*, *14*(7/8). Retrieved April 24, 2011, from http://dlib.org/dlib/july08/hagedorn/07hagedorn.html.

2. Madhavan, J., Jeffery, S., Cohen, S., Dong, X., Ko, D., Yu, C., et al. (2007). Web-scale data integration: You can only afford to pay as you go. *Proceedings of CIDR* (pp. 342–350). Citeseer. Retrieved April 24, 2011, The Conference on Innovative Data Systems Research (*CIDR*) , Asilomar, E.U.A

3. Pang-Ning Tan, Michael Steinbach, Vipin Kumar (2009), Introdução ao Data Mining: Mineração de Dados, Editora: Ciência Moderna, pp.172-223,245-327

4. Gar Garofalakis, M., Rastogi, R., Seshadri, S., Shim, K. (1999), Data Mining and the Web: Past, Present and Future, *ACM Workshop on Web Information and Data Management (WIDM)*, pp. 43-47, Kansas City, Missouri, USA

5. Ana Carolina Lorena, André C. P. L. F. de Carvalho (2007), Uma Introdução às Support Vector Machines, Revista de Informática Teórica e Aplicada – RITA, Volume XIV Número 2, pp. 43-67

6. Ricardo Baeza-Yates, Berthier Ribeiro-Neto (2010), Modern Information Retrieval, Second Edition, pp. 25-30

7. P. Domingos (1999). The Role of Occam's razor in Knowledge Discovery, *Data Mining and Knowledge Discovery*, Volume 3, pp. 409--425

8. Raphael do Vale Amaral Gomes (2010), Classificação automática de bancos de dados da deep web, PUC-RJ, ISSN: 0103-9741, Monografias em Ciência da Computação, Vol. 10

9. Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin (2003), A Practical Guide to Support Vector Classification, National Taiwan University, Taipei, Taiwan, disponível em http://www.csie.ntu.edu.tw/~cjlin/papers.html, acesso em 9/12/2011

10. BChristopher J. C. Burges (1998). A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery*, Volume 2, pp. 121-167

11. Wenliang Du and Zhijun Zhan (2003), Using Randomized Response Techniques for Privacy-Preserving Data Mining*, KDD '03 Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery*

and data mining, ISBN:1-58113-737-0 doi: 10.1145/956750.956810, Nova York, E.U.A

12. Andréia de Jesus (2009), Sistemas Tutoriais Inteligentes, *Revista Eletrônica de Sistemas de Informação*, ISSN 1677-3071 doi:10.5329/RESI

13. The UIUC web integration repository. Computer Science Department, University of Illinois at Urbana-Champaign (2003), disponivel em: http://metaquerier.cs.uiuc.edu/repository, acesso em 9/12/2011

14. Jericho HTML Parser (2002). http://jericho.htmlparser.net/docs/index.html

15. Zhou, X., Zhang, X., and Hu, X. (2007), "Dragon Toolkit: Incorporating Auto-learned Semantic Knowledge into Large-Scale Text Retrieval and Mining," In *proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, October 29-31, Patras, Grécia

16. Chih-Chung Chang and Chih-Jen Lin (2011), LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, Software disponível em http://www.csie.ntu.edu.tw/~cjlin/libsvm, acesso em 9/11/2011

17. Chih-Chung Chang and Chih-Jen Lin (2001), LIBSVM: A Library for Support Vector Machines, National Taiwan University, Taipei, Taiwan, tutorial disponível em http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, acesso em 9/12/2011

18. Wei Liu, Xiaofeng Meng, Weiyi Meng (2010), ViDE: A Vision-Based Approach for Deep Web Data Extraction, *EEE Transactions On Knowledge And Data Engineering*, Vol. 22, p. 447-460

19. Gerard Salton, Christopher Buckley, Term-weighting approaches in automatic text retrieval (1988), Information Processing & Management Vol. 24, No. 5, pp. 513-523, Nova York, E.U.A

20. Earl B. Hunt. Concept learning: An information processing problem (1962), *Behavior Science, Vol. 8 pp. 346*, publicado online em 17/01/2007, doi: 10.1002/bs.3830080408, N.J, E.U.A

21. J. R. Quinlan (1986), Induction of Decision Trees, Machine Learning Vol. 1, pp. 81-106

22. Cover TM, Hart PE (1967), Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13 (1): 21–27. doi:10.1109/TIT.1967.1053964

23. Bayes, Thomas, and Price, Richard (1763). An Essay towards solving a Problem in the Doctrine of Chance. By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S. Philosophical Transactions of the Royal Society of London 53 (0): 370–418. doi:10.1098/rstl.1763.0053

24. S. Tan, X. Cheng, Y. Wang, and H. Xu (2009), Adapting Naive Bayes to Domain Adaptation for Sentiment Analysis, Proc. ECIR, pp.337-349.

25. L. Barbosa and J. Freire (2005), Searching for Hidden-Web Databases. In Proceedings of WebDB, pp. 1-6, Baltimore, Maryland, USA

26. J. Von Zuben, Fernando (2008), Inteligência Artificial em aplicações Industriais (Unicamp), tutorial disponível em http://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ea072_2s11/topico9_EA072_2s11.pdf, acesso em 11/12/2011

27. Quinlan, J. R. (1987), Generating production rules from decision trees, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*. Milan, Italy. pp. 304–307

28. W. W Cohen, Fast Effective Rule Induction (1995). Proc. Of the 12th Conf. on Machine Learning, pp. 115-123, Tahoe City

29. R. Marques e Inês Dutra, Redes Bayesianas: o que são, para que servem, algoritmos e exemplos de aplicações. Coppe sistemas - UFRJ, disponível em http://www.cos.ufrj.br/~ines/courses/cos740/leila/cos740/Bayesianas.pdf acesso em 11/12/2011

30. Andrew W. Moore, Cross-validation for detecting and preventing overfitting, School of Computer Science, Carnegie Mellon University, tutorial disponível em http://www.autonlab.org/tutorials/overfit10.pdf acesso em 11/12/2011

31. Álvares, Luís Otávio, Classificação: conceitos básicos e árvore de decisão, UFRGS, tutorial disponível em http://www.inf.ufrgs.br/~alvares/CMP259DCBD/classificacao.pdf, acesso em 11/12/2011

32. Hieu Quang Le, Stefan Conrad (2009), Classifying Structured Web Sources Using Aggressive Feature Selection, Web Information Systems and Technologies - WEBIST , pp. 618-625

33. Chang, Kevin et al (2003). Structured Databases on the Web: Observations and Implications, Journal SIGMOD Record, pp. 61-70, Lisboa, Portugal