Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Introdução aos Classificadores de Margens Largas (*Large Margin Classifiers*)

Ana Carolina Lorena
André C. P. L. F. de Carvalho

Nº 195

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Maio/2003

Introdução aos Classificadores de Margens Largas (*Large Margin Classifiers*)*

Ana Carolina Lorena
André C. P. L. F. de Carvalho

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
Laboratório de Inteligência Computacional
Caixa Postal 668, 13560-970 - São Carlos, SP, Brasil
e-mail: {aclorena, andre}@icmc.sc.usp.br

Resumo. Este relatório apresenta uma introdução aos Classificadores de Margens Largas (*Large Margin Classifiers* - LMCs). Essas técnicas de aprendizado utilizam o princípio de maximização de margens na indução de classificadores a partir de amostras de dados. As margens correspondem a valores que determinam quanto os dados de diferentes classes encontram-se separados. A utilização desta estratégia provê bons desempenhos em termos de generalização, fato embasado pela Teoria de Aprendizado Estatístico.

Palavras-Chave: Classificadores de Margens Largas (*Large Margin Classifiers*), Aprendizado de Máquina, Máquinas de Vetores Suporte (*Support Vector Machines*), Perceptron com Margens Largas, Árvores de Decisão com Margens Largas, *Boosting*, Discriminante de Fisher com Kernels.

Maio 2003

*Trabalho realizado com auxílio da FAPESP.

Este documento foi preparado com o formatador de textos LaTeX. O sistema de citações de referências bibliográficas utiliza o padrão *Apalike* do sistema bibTeX.

Sumário

1 Introdução **1**

1.1 Aprendizado de Máquina 1

1.2 Classificadores Lineares e o Conceito de Margens 2

1.3 Organização do Relatório 5

2 A Teoria de Aprendizado Estatístico **6**

2.1 Conceitos Básicos 7

2.2 Limites no Risco Funcional Baseados na Teoria de Dimensão VC 8

2.3 Limites no Risco Funcional Baseados em Margens 13

2.4 Considerações Finais 15

3 Técnicas **17**

3.1 Máquinas de Vetores Suporte 17

3.2 Perceptron com Margens Largas 23

3.3 Árvores de Decisão com Margens Largas 26

3.4 Boosting 28

3.5 Discriminante de Fisher com Kernels 32

3.6 Considerações Finais 35

4 LMCs para Várias Classes **36**

4.1 Decomposição um-contra-todos 36

4.2 Decomposição todos-contra-todos 37

4.3 Método de Códigos de Correção de Erros 38

4.4 Considerações Finais 39

5 Conclusão **40**

Referências Bibliográficas **41**

Lista de Figuras

1.1 Ilustração de divisão no espaço de entradas por um hiperplano. 3
1.2 Exemplo de conjunto de dados linearmente separável. 4
1.3 Dois possíveis hiperplanos separadores para o conjunto de dados da Figura 1.2. . 4
1.4 Ilustração das margens dos hiperplanos separadores apresentados na Figura 1.3. 4

2.1 Exemplo de conjunto de treinamento binário classificado segundo três diferentes hipóteses. 7
2.2 Exemplo de distribuição de três pontos no estado bidimensional. 10
2.3 Distribuição de quatro pontos no espaço bidimensional. 10
2.4 Princípio de Minimização do Erro Estrutural (Smola and Schölkopf, 2002). . . . 12

3.1 Ilustração da distância d entre os hiperplanos $\mathbf{w} \cdot \mathbf{x}_1 + b = -1$ e $\mathbf{w} \cdot \mathbf{x}_2 + b = +1$. 19
3.2 Transformação do conjunto de dados no espaço de entrada para o espaço de características. 22
3.3 Regiões retângulares formadas por uma AD tradicional no espaço de entradas. 26
3.4 Regiões poliedrais formadas por uma AD Perceptron no espaço de entradas. . . 27
3.5 Ilustração de um comitê de classificadores (Sharkey, 1996). 28

4.1 Exemplo de grafo direcionado utilizado para classificar quatro classes a partir de preditores binários (Platt et al., 2000). 38

Lista de Tabelas

2.1 Ilustração esquemática do exemplo apresentado na Figura 2.2. 10

3.1 Sumário dos principais Kernels utilizados nas SVMs (Haykin, 1999). 24

Lista de Algoritmos

3.1 SVM não linear (Vert, 2001). 23
3.2 Perceptron básico (Smola et al., 1999b). 24
3.3 *Voted-Perceptron* (Freund and Schapire, 1998). 25
3.4 *AdaBoost* (Schapire, 1999) 30

Capítulo 1

Introdução

Trabalhos atuais mostram evidências de que algoritmos de aprendizado que explícita ou implicitamente exploram o conceito de maximização de margens estão entre as abordagens mais promissoras no aprendizado a partir de dados (Barlett and Shawe-Taylor, 1999; Schapire et al., 1997; Cristianini, 2000). Margens correspondem a valores que determinam o quanto duas ou mais classes estão separadas, ou seja, o quanto os exemplos encontram-se distantes das fronteiras de decisão formadas para sua classificação. Os algoritmos que utilizam esse princípio são denominados "Classificadores de Margens Largas" (*Large Margin Classifiers* - LMCs). Entre os principais LMCs tem-se as Máquinas de Vetores Suporte (*Support Vector Machines*) (Cristianini and Shawe-Taylor, 2000) e a técnica de *Boosting*, utilizada na combinação de preditores (Schapire, 1999).

Este relatório apresenta uma introdução aos LMCs. Iniciando as descrições a respeito deste tipo de classificadores, a Seção 1.1 apresenta alguns conceitos de Aprendizado de Máquina (AM), campo de estudo em Inteligência Computacional do qual os LMCs fazem parte. A Seção 1.2 apresenta a definições de classificadores lineares e de margens. A organização deste relatório é apresentada na Seção 1.3.

1.1 Aprendizado de Máquina

Aprendizado de Máquina (AM) é um campo de pesquisa da Inteligência Computacional que estuda o desenvolvimento de métodos capazes de extrair conceitos (conhecimento) a partir de amostras de dados (Mitchell, 1997).

Em geral, os diversos algoritmos de AM são utilizados de forma a gerar classificadores para um conjunto de exemplos. Por classificação entende-se o processo de atribuir, a uma determinada informação, o rótulo de uma classe[1] a qual ela pertence. Portanto, as

[1] A classe de um exemplo (instância) descreve o fenômeno de interesse, ou seja, o que se deseja aprender.

técnicas de AM são empregadas na indução (a partir de um conjunto de treinamento) de um classificador, que deve ser capaz (idealmente) de prever a classe de instâncias quaisquer do domínio em que ele foi treinado.

O conjunto de treinamento é normalmente composto de pares na forma entrada-saída desejada. Uma hipótese h deve então ser induzida de forma a produzir as saídas desejadas apresentadas. Essa hipótese deve também ser capaz de prever saídas de novos exemplos de forma precisa. Portanto, duas medidas importantes na geração do classificador dizem repeito à quantidade de erros cometidos no treinamento do mesmo e sua eficiência na classificação de dados que não pertençam ao conjunto utilizado em seu treinamento. A segunda medida quantifica a capacidade de generalização do classificador.

Se o classificador é eficiente na classificação dos dados de treinamento, mas obtém baixo desempenho quando confrontado com dados novos, diz-se que houve a ocorrência de *overfitting*. Neste caso considera-se que o classificador se ajustou demasiamente à amostra de treinamento.

Já a utilização de uma hipótese muito simples na separação dos dados pode levar ao efeito oposto de *underfitting*, em que uma precisão baixa é verificada também na classificação dos dados de treinamento.

Deve-se buscar então uma hipótese que represente um compromisso entre a precisão obtida no treinamento e no teste (classificação de novas instâncias) do classificador. Os LMCs buscam este compromisso na indução de classificadores através do conceito de maximização de margens, introduzido na próxima Seção.

1.2 Classificadores Lineares e o Conceito de Margens

Seja S um conjunto de dados de treinamento composto por n pares[2] $(\mathbf{x}_i, y_i)$, em que $\mathbf{x}_i \in \Re^m$ e $y_i \in \{-1, +1\}$. A partir de um processo de indução sobre este conjunto, o objetivo é encontrar uma função $g : \Re^m \rightarrow \{-1, +1\}$ capaz de predizer a classe de novos pontos $(\mathbf{x}, y)$ de forma precisa.

Uma forma de realizar esta tarefa é utilizar uma função linear $f : \Re^m \rightarrow \Re$, representada pela Equação 1.1.

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{1.1}$$

em que $\mathbf{w} \cdot \mathbf{x}$ é o produto escalar entre os vetores $\mathbf{w}$ e $\mathbf{x}$, $\mathbf{w}$ é o vetor normal ao hiperplano e b é um termo "compensador" (*bias*).

Esta Equação divide o espaço de entradas em duas regiões: $\mathbf{w} \cdot \mathbf{x} + b > 0$ e $\mathbf{w} \cdot \mathbf{x} + b < 0$, conforme ilustrado na Figura 1.1. A regra 1.2 pode então ser definida (Müller et al., 2001).

[2]Váriáveis vetoriais serão denotadas em negrito neste relatório.

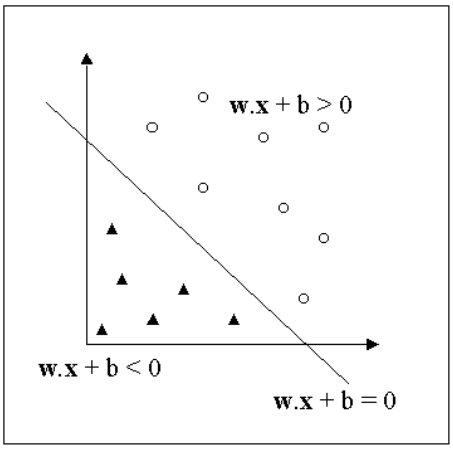

Figura 1.1: Ilustração de divisão no espaço de entradas por um hiperplano.

$$\begin{cases} y_i = +1 \; se \; \mathbf{w} \cdot \mathbf{x}_i + b > 0 \\ y_i = -1 \; se \; \mathbf{w} \cdot \mathbf{x}_i + b < 0 \end{cases} \tag{1.2}$$

e se $\mathbf{w} \cdot \mathbf{x}_i + b = 0$, y_i é indefinido.

A função g, representada na Equação 1.3, sumariza esta informação.

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})), \text{em que } f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{1.3}$$

A função g recebe como entrada a saída de $f(\mathbf{x})$ e computa uma função sinal com centro em 0. O valor de $g(\mathbf{x})$ é $+1$ se $f(\mathbf{x}) > 0$ e -1 se $f(\mathbf{x}) < 0$. A função $f(\mathbf{x})$ define um hiperplano separando as classes $+1$ e -1.

Neste caso, o problema de aprendizado passa a ser encontrar os parâmetros $(\mathbf{w}, b)$ para os quais $g(\mathbf{x})$ apresente erro mínimo na classificação dos dados de treinamento. Os LMCs utilizam como base classificadores lineares deste tipo (Equação 1.3). Na busca por classificadores lineares com erro mínimo em um conjunto de treinamento, porém, várias soluções podem existir.

Considere por simplicidade o conjunto linearmente separável[3] dado na Figura 1.2. Existem diversas combinações $(\mathbf{w}_j, b_j)$ capazes de separar este conjunto nas classes "círculo" e "triângulo". Duas destas são ilustradas na Figura 1.3. Para se escolher a melhor combinação de parâmetros, introduz-se o conceito de margem.

Dada uma função $f : \Re^m \rightarrow \Re$, a margem com o qual um padrão $\mathbf{x}_i$ é classificado é fornecida pela Equação 1.4. Ela mede a distância do padrão $\mathbf{x}_i$ em relação ao hiperplano

[3]Um conjunto é linearmente separável se é possível separar todos dados de diferentes classes presentes no mesmo por (pelo menos) um hiperplano (Russel and Norvig, 1995).

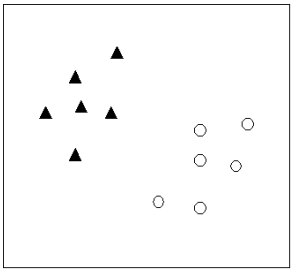

Figura 1.2: Exemplo de conjunto de dados linearmente separável.

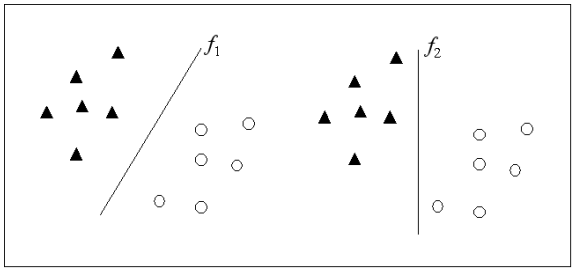

Figura 1.3: Dois possíveis hiperplanos separadores para o conjunto de dados da Figura 1.2.

separador, sendo que uma margem negativa indica uma classificação incorreta.

$$\rho(\mathbf{x}_i, y_i) = y_i f(\mathbf{x}_i) \tag{1.4}$$

A margem ρ_f do classificador linear f é então definida como a margem mínima observada em todo conjunto de treinamento, ou seja,

$$\rho_f = \min_{1 \leq i \leq n} \rho\left(\mathbf{x}_i, y_i\right) = \min_{1 \leq i \leq n} y_i f\left(\mathbf{x}_i\right) \tag{1.5}$$

A Figura 1.4 ilustra as margens dos dois hiperplanos separadores apresentados na Figura 1.3.

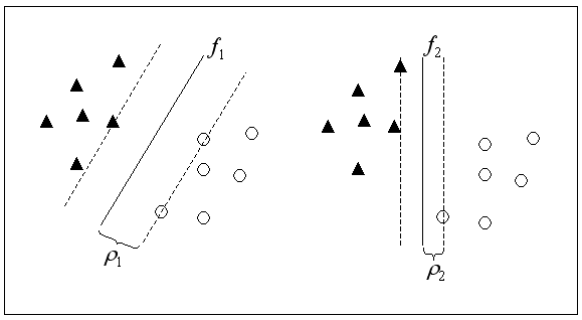

Figura 1.4: Ilustração das margens dos hiperplanos separadores apresentados na Figura 1.3.

Uma possível escolha de classificador é aquele que produz margem máxima de separação entre as classes, ou seja, de uma f' com

$$\rho_{f'} = \arg\max_{f} \rho_f \tag{1.6}$$

Este tipo de função é mais confiável em relação à classificação do conjunto de treinamento, o que pode ser refletido no conjunto de teste, composto de novos padrões. Essa confiabilidade é refletida pelo fato da maximização de margens prover duas propriedades interessantes ao classificador linear: robustez em relação aos padrões e robustez em relação aos parâmetros do modelo (Smola et al., 1999b).

Na robustez em relação aos padrões, tem-se que dado um exemplo $\mathbf{x}$ longe da borda de decisão $f(\mathbf{x}) = 0$, ocasionais pertubações em $\mathbf{x}$ não levarão a uma classificação incorreta $g(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$. Como $f(\mathbf{x})$ é contínua, pequenas variações em $\mathbf{x}$ levarão a pequenas variações em $f(\mathbf{x})$. Logo, dado ε pequeno e positivo, se $\rho(\mathbf{x}_i, y_i) = y_i f(\mathbf{x}_i)$ for um número positivo grande, $y_i f(\mathbf{x}_i \pm \varepsilon)$ também será positivo. A classificação de $\mathbf{x}_i \pm \varepsilon$ permanece então correta.

Para o caso da robustez em relação aos parâmetros, se a margem do classificador é grande, uma pequena perturbação nos parâmetros de f não afeta a classificação dos dados. Como $f(\mathbf{x})$ é contínua em relação a seus parâmetros $\mathbf{w}$ e b, se $y_i f(\mathbf{x}_i)$ é um número muito maior que 0, então $y_i f_{(\mathbf{w},b)\pm\varepsilon}(\mathbf{x}_i)$, também será positivo, para pequenos ε. Desta forma, a classificação dos dados se mantém correta.

Além das motivações apresentadas anteriormente, existem limites no erro de um classificador linear sobre novos dados baseados na margem, confirmando a indicação de que a maximização desta leva a bons desempenhos em termos de generalização. Esses limites são fornecidos através da Teoria de Aprendizado Estatístico, apresentada no próximo Capítulo.

1.3 Organização do Relatório

Como os LMCs têm seus conceitos embasados na Teoria de Aprendizado Estatístico, o Capítulo 2 apresenta os resultados principais desta teoria e como as LMCs fazem uso de seus princípios.

A seguir, o Capítulo 3 apresenta alguns dos principais LMCs descritos na literatura.

Uma vez que grande parte dos LMCs apresentados são propostos para problemas de classificação binária (envolvendo apenas duas classes), o Capítulo 4 mostra como estes algoritmos podem ser utilizados em problemas que envolvam a existência de mais classes.

O Capítulo 5 aponta então as principais conclusões deste estudo.

Capítulo 2

A Teoria de Aprendizado Estatístico

Seja f um classificador, isto é, um mapeamento de um conjunto de padrões $\mathbf{x_i} \in \Re^m$ para o espaço de classes, e F o conjunto de todos classificadores que um determinado algoritmo de AM pode gerar. Este algoritmo, durante o aprendizado, utiliza um conjunto de treinamento S, composto de n pares $(\mathbf{x_i}, y_i)$, em que y_i representa a classe do padrão $\mathbf{x_i}$, para gerar um classificador particular $f' \in F$.

Considere por exemplo[1] o conjunto de treinamento da Figura 2.1. Deve-se encontrar um classificador (ou função) que separe as instâncias de treinamento das classes "círculo" e "triângulo". As funções ou hipóteses consideradas são ilustradas na figura através das bordas, também denominadas fronteiras de decisão, traçadas entre as classes. Na imagem 2.1c, tem-se uma hipótese que classifica corretamente todos os exemplos do conjunto de treinamento. A função utilizada, porém, tem grandes chances de cometer erros quando confrontada com exemplos de teste distintos dos de treinamento, pois é muito específica para os últimos. Esse caso representa a ocorrência de *overfitting*, em que considera-se que o algoritmo "memorizou" os dados do conjunto de treinamento.

Um outro modelo poderia desconsiderar pontos de classes opostas muito próximos entre si, pois esses dados são pouco confiáveis (mesmo uma pequena quantidade de ruídos pode levar a uma mudança do valor de sua classe). A ilustração 2.1a representa esta alternativa, em que se utiliza uma fronteira de decisão linear. A nova hipótese considerada, porém, comete muitos erros, mesmo para casos considerados simples. Pode-se considerar que houve a ocorrência de *underfitting*, pois o classificador não é capaz de se ajustar até mesmo às instâncias de treinamento.

Um compromisso entre as duas funções apresentadas anteriormente é representado em 2.1b, em que o preditor tem complexidade intermediária e classifica corretamente grande parte dos dados.

[1] Baseado em exemplo fornecido em (Smola and Schölkopf, 2002).

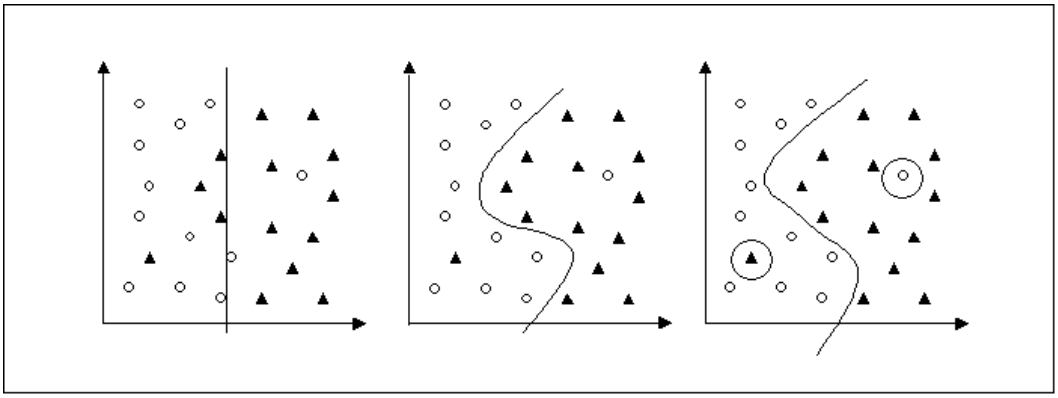

Figura 2.1: Exemplo de conjunto de dados de treinamento pertencentes a duas classes classificado segundo três diferentes hipóteses.

A Teoria de Aprendizado Estatístico (TAE) visa estabelecer condições matemáticas que permitam a escolha de um classificador f' com bom desempenho para os conjuntos de treinamento e teste. Ou seja, busca-se uma função f' capaz de classificar os dados de treinamento da forma mais correta possível, sem dar atenção exacerbada a qualquer ponto individual do mesmo.

A seguir alguns conceitos básicos referentes a esta teoria são apresentados (Seção 2.1), seguidos de limites na generalização do classificador que podem ser utilizados na busca de uma hipótese "ótima" (Seções 2.2 e 2.3).

2.1 Conceitos Básicos

Na escolha de uma função particular f', é natural considerar a função que produz o menor erro durante o treinamento, ou seja, aquela que possui maior habilidade de classificar corretamente os padrões do conjunto de treinamento S. Seja o risco empírico de f ($R_{emp}(f)$), definido pela Equação 2.1, a porcentagem de classificações incorretas obtidas em S, em que $c(.)$ é uma função de custo relacionando a previsão $f(\mathbf{x}_i)$ com a saída desejada y_i (Müller et al., 2001). Um tipo de função de custo é a "perda 0/1", definida por $c(f(\mathbf{x}_i), y_i) = 1$ se $y_i f(\mathbf{x}_i) < 0$ e 0 caso contrário (Müller et al., 2001).

O processo de busca por uma função f' que apresente menor R_{emp} é denominado *Minimização do Risco Empírico*.

$$R_{emp}\left(f\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}c(f\left(\mathbf{x}_i\right), y_i) \tag{2.1}$$

Supõe-se que os exemplos do domínio em que está ocorrendo o aprendizado (incluindo

os de treinamento) são gerados independentemente e identicamente distribuídos (i.i.d.) de acordo com uma distribuição de probabilidade P. O desempenho de generalização de um classificador f pode então ser definido como a probabilidade de que f cometa erro na classificação de um novo exemplo gerado segundo P. A partir desta afirmação, define-se o *Risco Funcional* $R(f) = P(f(X) \neq Y)$ (Equação 2.2), que quantifica a capacidade de generalização de f (se $R(f)$ for igual a 0, pode-se afirmar que f generaliza de forma "perfeita").

$$R(f) = \int \frac{1}{2} c(f(\mathbf{x}), y) dP(\mathbf{x}, y) \tag{2.2}$$

Durante o treinamento, $R_{emp}(f)$ é facilmente observável. O mesmo não pode ser afirmado sobre $R(f)$, pois em geral a distribuição de probabilidade P é desconhecida. A maioria dos algoritmos de aprendizado busca a minimização de $R_{emp}(f)$, esperando que esta leve a um menor $R(f)$. Portanto, na escolha de uma função f, geralmente se opta por uma função $\hat{f}$ tal que $R_{emp}\left(\hat{f}\right) = \min_{f \in F} R_{emp}(f)$. Porém, também é desejável que o classificador possua uma boa capacidade de generalização, ou seja, que se tenha f^* tal que $R(f^*) = \min_{f \in F} R(f)$.

Em geral, $\hat{f}$ leva a uma boa aproximação de f^*. Contudo, nem sempre isto é verdade. Considere, por exemplo, um estimador que memoriza todos os dados de treinamento e gera classificações aleatórias para outros exemplos (Smola et al., 1999b). Este classificador possui um risco empírico nulo, porém seu risco funcional é igual a 1/2.

A TAE provê diversos limites no risco funcional de uma função, os quais podem ser então utilizados na escolha do classificador. As próximas Seções relacionam alguns dos principais limites sobre os quais os LMCs se baseiam.

2.2 Limites no Risco Funcional Baseados na Teoria de Dimensão VC

Seja G o conjunto de funções sinal $g(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$, sendo $f(\mathbf{x})$ uma função contínua qualquer. A TAE provê limites no risco funcional para funções deste tipo que relacionam o número de exemplos de treinamento, o risco empírico obtido na classificação desses dados e a complexidade do espaço de hipóteses. Esta complexidade é medida através do conceito de dimensão Vapnik-Chervonenkis (VC) (Vapnik, 1995).

Dado um conjunto de funções sinal G, sua dimensão VC é definida como o tamanho do maior conjunto de pontos que pode ser particionado arbitrariamente pelas funções contidas em G (Smola et al., 1999b).

Como discutido anteriormente, pode-se considerar que a função $g(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$

divide o espaço de entradas em dois subconjuntos disjuntos, o de padrões $\mathbf{x}_i$ para os quais $f(\mathbf{x}_i) > 0$ e o de padrões $\mathbf{x}_i$ com $f(\mathbf{x}_i) < 0$. Funções deste tipo são também denominadas dicotomias (Haykin, 1999).

Seja $\Delta_G(S)$ o número de dicotomias que um algoritmo de aprendizado tem capacidade de induzir sobre S. Diz-se que S é "fragmentado" por G se $\Delta_G(S) = 2^{|S|}$ ($|.|$ representa a cardinalidade, ou tamanho, de um conjunto), isto é, se as funções contidas em G são capazes de induzir todas as possíveis dicotomias sobre S.

A dimensão VC de um conjunto de dicotomias G é então definida como a cardinalidade do maior conjunto S que é fragmentado por G, ou seja, o maior N tal que $\Delta_G(S) = 2^N$, em que $N = |S|$. Caso este valor não possa ser estimado, N assume o valor ∞.

Logo, um alto valor de dimensão VC indica uma grande complexidade das funções de decisão contidas em G, já que essas se tornam capazes de fragmentar conjuntos de dados também complexos.

O Exemplo 1, extraído de (Haykin, 1999), ilustra de forma mais clara os conceitos apresentados.

Exemplo 1: Seja G' o conjunto de funções sinal com fronteira linear, como representado na Equação 2.3.

$$G' : g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.3}$$

em que $\mathbf{x}$ representa uma entrada m-dimensional, $\mathbf{w}$ o vetor de pesos e b o *bias*.

A dimensão VC do conjunto de funções G' (Expressão 2.3) é dada por 2.4.

$$V(G') = m + 1 \tag{2.4}$$

em que m é a dimensão de $\mathbf{x}$.

Este resultado pode ser analisado através da observação das Figuras 2.2 e 2.3.

Na Figura 2.2, há três pontos bidimensionais, o que gera oito possíveis combinações binárias dos dados, ou seja, oito dicotomias. Um número de oito retas é suficiente para fragmentar estes dados em todas classificações binárias admissíveis. As classificações possíveis e retas utilizadas em sua fragmentação são sumarizadas na Tabela 2.1.

Na Figura 2.3, tem-se os pontos bidimensionais $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$ e $\mathbf{x_4}$. Uma das possíveis distribuições considerada é aquela em que $\mathbf{x_1}$ e $\mathbf{x_4}$ pertencem à classe "círculo" e $\mathbf{x_2}$ e $\mathbf{x_3}$ são da classe "triângulo". Neste caso não é possível separar os dados com o uso de uma reta. É necessária uma função de maior complexidade para realização desta tarefa.

A dimensão VC do conjunto de funções 2.3 no caso bidimensional é, portanto, igual a 3, em conformidade com a Equação 2.4.

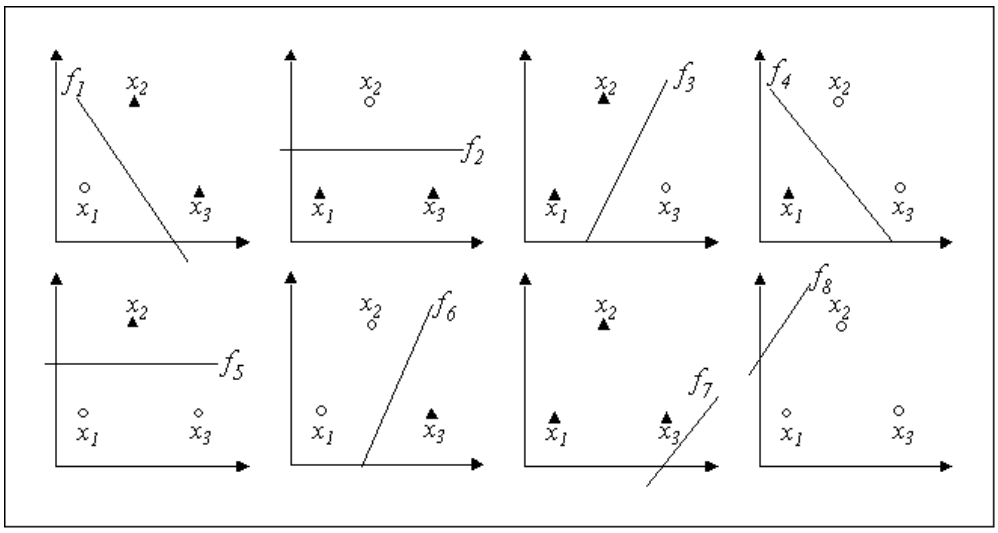

Figura 2.2: Exemplo de distribuição de três pontos no estado bidimensional. Verifica-se que para todas as dicotomias possíveis, é possível traçar retas que separem as classes "círculo" e "triângulo".

x_1	x_2	x_3	**Função**
○	▲	▲	f_1
▲	○	▲	f_2
▲	▲	○	f_3
▲	○	○	f_4
○	▲	○	f_5
○	○	▲	f_6
▲	▲	▲	f_7
○	○	○	f_8

Tabela 2.1: Ilustração esquemática do exemplo apresentado na Figura 2.2.

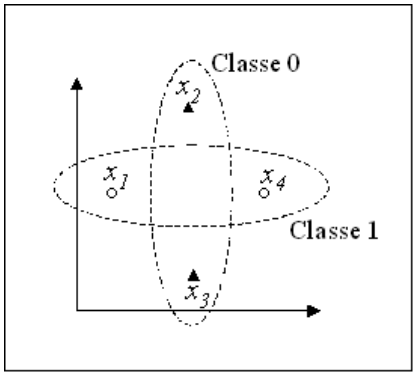

Figura 2.3: Distribuição de quatro pontos no espaço bidimensional. Para este caso, verifica-se que não é possível separar as classes por meio de uma reta.

Os teoremas seguintes provêem limites no risco funcional ($R(.)$) de uma função sinal baseados na dimensão VC do espaço de hipóteses do qual o classificador é extraído (Smola et al., 1999b).

Teorema 2.1 (Limite superior) *Seja G um conjunto de funções de decisão mapeando $\Re^m$ em $\{-1,+1\}$ com dimensão VC igual a h. Para qualquer distribuição de probabilidade P em $\Re^m$x$\{-1,+1\}$, com probabilidade de ao menos $1-\delta$ sobre n exemplos e para qualquer hipótese g em G, o risco funcional é limitado por*

$$R(g) \leq R_{emp}(g) + \sqrt{\frac{c}{n}\left(h + \ln\left(\frac{1}{\delta}\right)\right)} \tag{2.5}$$

em que c é uma constante universal. Se $\hat{g} \in G$ minimiza o risco empírico ($R_{emp}(.)$), então com probabilidade $1-\delta$

$$R(\hat{g}) \leq \inf_{g \in G} R(g) + \sqrt{\frac{c}{n}\left(h + \ln\left(\frac{1}{\delta}\right)\right)} \tag{2.6}$$

Teorema 2.2 (Limite inferior) *Seja G um conjunto de funções de decisão mapeando $\Re^m$ em $\{-1,+1\}$ com dimensão VC finita $h \geq 1$. Para qualquer algoritmo de aprendizado há distribuições tal que com probabilidade de ao menos $1-\delta$ sobre n exemplos, o risco funcional da hipótese $g \in G$ satisfaz*

$$R(g) \geq \inf_{g' \in G} R(g') + \sqrt{\frac{c}{n}\left(h + \ln\left(\frac{1}{\delta}\right)\right)} \tag{2.7}$$

em que c é uma constante universal.

O termo mais à direita das Equações 2.5, 2.6 e 2.7 é também denominado termo de capacidade do conjunto de funções G.

Por meio da observação das equações apresentadas nos Teoremas 2.1 e 2.2, as seguintes asserções podem ser realizadas:

- O risco funcional de uma função g é minimizado se o número de observações n do conjunto de treinamento for suficientemente grande. Quanto maior a amostra de dados apresentada ao algoritmo de aprendizado, mais a minimização do risco empírico se aproxima do risco mínimo que o espaço de funções G pode alcançar;

- O risco médio de g é minimizado se a dimensão VC de G for suficientemente pequena. Ou seja, quanto menor a dimensão VC de G, maior a capacidade de generalização

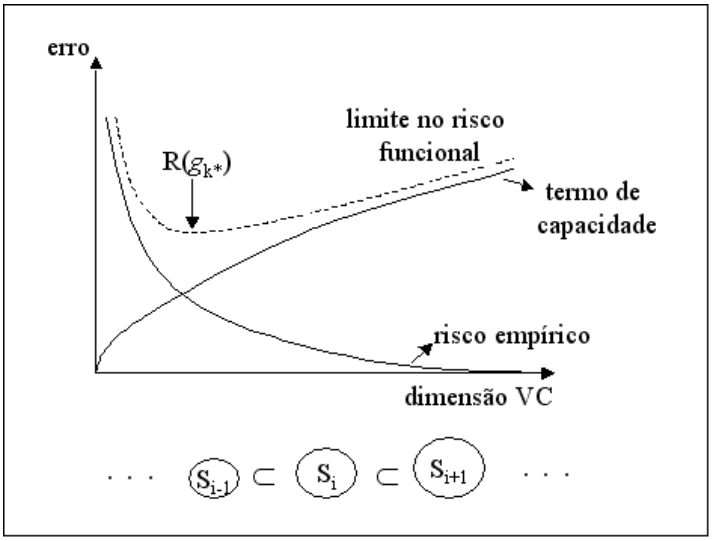

Figura 2.4: Princípio de Minimização do Erro Estrutural (Smola and Schölkopf, 2002).

das funções $g \in G$. Deve-se então escolher uma hipótese $\hat{g}$ que minimize o risco empírico e pertença a uma classe de funções G com baixa dimensão VC.

A contribuição principal dos teoremas apresentados está em afirmar a importância de se controlar o termo de capacidade apresentado nas Equações 2.5 a 2.7.

Como os limites apresentados dizem respeito a uma classe de funções G e não simplesmente a escolhas de funções g, introduz-se uma "estrutura" em G e realiza-se a minimização dos limites sobre escolhas de estruturas. Este princípio é denominado *Minimização do Risco Estrutural* (Smola and Schölkopf, 2002).

Considera-se conjuntos de estruturas S_i, com complexidade crescente ($\ldots S_{i-1} \subset S_i \subset S_{i+1} \ldots$). Como as estruturas são maiores à medida que i cresce, a capacidade das mesmas também é maior com o crescimento deste índice.

Para cada S_k, seja $\hat{g}_k$ o classificador com menor Risco Empírico e g_k^* o que possui menor Risco Funcional. A medida que k cresce, o risco empírico diminui, já que a complexidade do conjunto de classificadores (ou seja, das fronteiras de decisão a serem geradas) é maior. Porém, o termo de capacidade aumenta com k. Como resultado, os limites em $R(\hat{g})$ fornecidos nos Teoremas 2.1 e 2.2 inicialmente decrescem com o aumento de k, e depois crescem. Logo, deve haver um valor ótimo k^* tal que se obtém um erro de generalização mínimo, conforme ilustrado na Figura 2.4. A escolha da função $\hat{g}_{k^*}$ constitui o princípio da Minimização do Risco Estrutural.

Embora os Teoremas 2.1 e 2.2 caracterizem o risco funcional de uma forma razoavelmente completa, na prática esta caracterização tem alguns problemas. Em primeiro lugar, computar os limites apresentados não é uma tarefa fácil (Müller et al., 2001). Além disso,

os teoremas indicam que, dados dois classificadores com mesmo risco empírico R_{emp}, deve-se escolher aquele com menor dimensão VC. Porém, existem resultados experimentais em que funções com maior dimensão VC (algumas vezes infinita) apresentam melhores resultados (Burges, 1998). Podem também haver problemas quando a dimensão VC de uma classe de funções é desconhecida ou infinita (Müller et al., 2001).

Existem limites alternativos relacionando o poder de generalização de um classificador à sua margem (Smola et al., 1999b; Cristianini and Shawe-Taylor, 2000). Esses limites levam em conta, portanto, a distribuição dos dados no espaço de hipóteses, sendo referenciados como "dependentes dos dados" (Shawe-Taylor et al., 1998; Shawe-Taylor and Cristianini, 1998).

2.3 Limites no Risco Funcional Baseados em Margens

Uma primeira análise mais refinada nos limites apresentados pela TAE foi proposta por Vapnik (Smola et al., 1999b). Esta relaciona a dimensão VC de um classificador linear à margem obtida pelo mesmo na separação dos dados. O Teorema 2.3 apresenta este resultado.

Teorema 2.3 *Seja $X_0 \subset \Re^m$ o conjunto de entradas com norma menor que $R > 0$ ($\|\mathbf{x}_i\| \leq R$, $\forall\, \mathbf{x}_i \in X_0$) e F o conjunto de funções lineares definidas em X_0 satisfazendo $\|f(\mathbf{x})\| \geq \rho$, em que ρ é a margem do classificador.*

$$F = \{\mathbf{x} \rightarrow \mathbf{w} \cdot \mathbf{x} |\, \|\mathbf{w}\| \leq 1,\ \mathbf{x} \in X_0\} \tag{2.8}$$

Considerando G o conjunto de funções sinal obtidas à partir de $G = \operatorname{sgn}(F)$ e h a dimensão VC de G, tem-se o resultado representado pela Equação 2.9

$$h \leq \min\left\{\frac{R^2}{\rho^2}, m\right\} + 1 \tag{2.9}$$

Pode-se observar através deste resultado que a dimensão VC de um conjunto de funções sinal $G = \operatorname{sgn}(F)$, com F linear, pode ser menor que o valor $m+1$ calculado no Exemplo 1, em que não se levava em conta a margem ρ (Smola et al., 1999b). Este teorema, portanto, completa a noção de dimensão VC para fronteiras lineares. Segundo o mesmo teorema, a margem do classificador linear e a dimensão VC do espaço de hipóteses do qual este é extraído se relacionam de forma indiretamente proporcional, ou seja, quanto maior a margem de um classificador menor sua dimensão VC. Porém, o limite apresentado ainda

é restrito, pois através dele se requer que todos exemplos, inclusive os de teste, estejam afastados do hiperplano de uma margem ρ.

O Teorema 2.4 provê limites no risco funcional para funções de decisão lineares em função unicamente da margem (Smola et al., 1999b).

Teorema 2.4 *Seja o erro marginal de f ($R_\rho(f)$) a proporção de exemplos de treinamento que têm margem menor que ρ.*

$$R_\rho(f) = \frac{1}{n}\sum_{i=1}^{n} |y_i f(\mathbf{x}_i) < \rho| \tag{2.10}$$

Seja F o conjunto de funções reais na bola de raio R (B_R):

$$F = \{f_w \,|\, f_w(\mathbf{x}) = (\mathbf{w}\cdot\mathbf{x}) \;\text{ com }\; \|\mathbf{w}\| \leq 1 \text{ e } \mathbf{x} \in B_R\} \tag{2.11}$$

Existe uma constante c tal que, para qualquer distribuição de probabilidade, com probabilidade de ao menos $1-\delta$ sobre n exemplos de treinamento, para todo $\rho > 0$ e toda função $f \in F$, com margem de ao menos ρ sobre todos exemplos de treinamento (ou seja, $R_\rho(f) = 0$), a Equação 2.12 é satisfeita.

$$R(f) \leq \frac{c}{n}\left(\frac{R^2}{\rho^2}\log^2\left(\frac{n}{\rho}\right) + \log\left(\frac{1}{\delta}\right)\right) \tag{2.12}$$

Com probabilidade de ao menos $1-\delta$, para todo $\rho > 0$, toda função $f \in F$ tem erro:

$$R(f) \leq R_\rho(f) + \sqrt{\frac{c}{n}\left(\frac{R^2}{\rho^2}\log^2\left(\frac{n}{\rho}\right) + \log\left(\frac{1}{\delta}\right)\right)} \tag{2.13}$$

Pode-se verificar através do Teorema 2.4 que o risco funcional é limitado pela soma do erro marginal a um termo de capacidade. Uma maior margem ρ implica em um menor termo de capacidade. Porém, este procedimento pode levar a um aumento na taxa de erro marginal, pois torna-se mais difícil obedecer a restrição de todos dados de treinamento estarem distantes de uma margem maior do hiperplano separador. Um baixo valor de ρ, por sua vez, leva a um erro marginal menor, porém aumenta o termo de capacidade. Deve-se então buscar um compromisso entre a maximização da margem e a obtenção de um baixo erro marginal. O Teorema 2.5 apresenta limites similares no risco funcional de funções do tipo sinal com fronteiras de decisão lineares (Smola and Schölkopf, 2002).

Teorema 2.5 *Seja G o conjunto de funções $g(\mathbf{x}) = \operatorname{sgn}(f(\mathbf{x})) = \operatorname{sgn}(\mathbf{w}\cdot\mathbf{x})$ com $\|\mathbf{w}\| \leq \Lambda$ e $\|\mathbf{x}\| \leq R$, para algum $R, \Lambda > 0$. Seja $\rho > 0$ e $R_\rho(f)$ o erro marginal de f. Para todas distribuições de probabilidade P gerando os dados, com probabilidade de ao menos*

$1-\delta$ *sobre* n *exemplos, e para qualquer* $\rho > 0$ *e* $\delta \in (0,1)$*, a probabilidade de um ponto de teste amostrado independentemente segundo* P *ser classificado incorretamente é limitado superiormente por*

$$R_\rho(f) + \sqrt{\frac{c}{n}\left(\frac{R^2\Lambda^2}{\rho^2}\ln^2 n + \ln\left(\frac{1}{\rho}\right)\right)} \tag{2.14}$$

em que c *é uma constante universal.*

Deve-se observar que o limite apresentado no Teorema 2.5 não se refere a um único classificador, mas a um conjunto de preditores treinados em diferentes conjuntos de dados gerados com a mesma distribuição de probabilidade P (Smola and Schölkopf, 2002). Porém, sua análise fornece resultados interessantes. Como no Teorema 2.4, o risco funcional é limitado pela soma do erro marginal a um termo de capacidade. Este termo pode ser minimizado mantendo-se R e Λ pequenos e maximizando-se a margem ρ. Fixando R e Λ, o termo de maior importância torna-se a margem ρ. Maximizar ρ leva a um termo de capacidade pequeno. Porém, como já discutido, o erro marginal $R_\rho(\cdot)$ pode se tornar maior.

Como conclusão tem-se que, na geração de um classificador linear, deve-se buscar o hiperplano que tenha margem ρ alta e cometa poucos erros marginais, minimizando-se assim o erro sobre os dados de teste e de treinamento, respectivamente. O hiperplano que possui esta margem é denominado ótimo.

A busca por um classificador que minimize os riscos empírico e funcional considerando a margem é também referenciada por "Minimização do Risco Estrutural baseada nos dados" (Shawe-Taylor et al., 1998).

Existem diversos outros limites no risco funcional reportados na literatura, assim como outros tipos de medida de complexidade de uma classe de funções. Um exemplo é a dimensão *"fat-shattering"*, que caracteriza o poder de um conjunto de hipóteses em separar os dados de treinamento com uma margem ρ (Shawe-Taylor and Cristianini, 1998). Porém, os limites discutidos anteriormente, embora simplificados, provêem a base teórica suficiente à compreensão dos LMCs.

2.4 Considerações Finais

Este Capítulo descreveu a Teoria de Aprendizado Estatístico, proposta por Vapnik e Chervonenkis no final da década de 60 (Vapnik, 1995). Esta teoria apresenta condições matemáticas para a escolha de uma função (hipótese) que separe os dados de treinamento de forma a efetivamente maximizar a generalização do classificador obtido. Para isto, busca-se um compromisso entre a minimização do erro de treinamento e a complexidade

da função escolhida para separação dos dados em classes.

Na geração de classificadores lineares, essas condições são alcançadas através da maximização da margem de separação entre os dados de treinamento e a fronteira de decisão induzida para separação destes em classes. Este é o princípio básico explorado pelos LMCs.

Capítulo 3

Técnicas

Diversos algoritmos foram propostos utilizando os limites providos pela TAE para a obtenção de classificadores com baixo risco funcional. Entre eles incluem-se generalizações de técnicas de AM amplamente difundidas, como o algoritmo Perceptron de Rosenblatt (1962), e algoritmos de AM formulados diretamente à partir dos limites da TAE, como as Máquinas de Vetores Suporte (Cortes and Vapnik, 1995).

Esta Seção apresenta algumas dessas técnicas. Nas considerações realizadas, supõe-se que se deseja classificar objetos m-dimensionais $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ nas classes -1 e +1.

Maior atenção é dada à descrição das Máquinas de Vetores Suporte, apresentadas na Seção 3.1, cujos princípios levaram à formulação de vários dos outros algoritmos discutidos. A Seção 3.2 apresenta uma generalização do algoritmo Perceptron para acomodar o conceito de maximização de margens. A Seção 3.3 apresenta um estudo semelhante em relação às Árvores de Decisão. A Seção 3.4 apresenta o *Boosting*, utilizado na combinação de preditores gerados por outras técnicas. A Seção 3.5 descreve o Discriminante de Fisher com Kernels, que não-lineariza as soluções fornecidas pelo algoritmo linear de Discriminante de Fisher. A Seção 3.6 conclui este Capítulo.

3.1 Máquinas de Vetores Suporte

As Máquinas de Vetores Suporte (*Support Vector Machines* - SVMs) surgiram pelo emprego direto dos resultados fornecidos pela TAE (Boser et al., 1992; Cortes and Vapnik, 1995). Iniciando as descrições a respeito desta técnica, considere um conjunto de treinamento S' cujos dados sejam linearmente separáveis. Na aplicação da TAE, deve-se escolher o classificador $g(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$ capaz de separar os dados pertencentes a este conjunto com o menor risco empírico possível e que também satisfaça a restrição de pertencer a uma família G que minimize o risco funcional.

A primeira condição, no caso de conjuntos de treinamento linearmente separáveis, é satisfeita para pelo menos um par $(\mathbf{w}, b)$. Para satisfazer a segunda restrição, utiliza-se o resultado da TAE relacionando a minimização do risco funcional à margem (ρ) obtida pelo classificador na classsificação dos dados. Portanto, entre os classificadores que minimizam o risco empírico, deve-se escolher aquele que possui a maior margem ρ'.

Como assumiu-se que o conjunto de treinamento é linearmente separável, pode-se reescalar $\mathbf{w}$ e b de forma que os pontos mais próximos do hiperplano separador satisfaçam $|\mathbf{w} \cdot \mathbf{x} + b = 1|$ (Müller et al., 2001). Obtém-se com isto a representação canônica do hiperplano. A partir dessa consideração, a desigualdade 3.1 caracteriza os classificadores lineares que separam o conjunto de treinamento com uma margem positiva. Segundo este sistema, não há pontos entre $\mathbf{w}\cdot\mathbf{x}+b = 0$ e $\mathbf{w}\cdot\mathbf{x}+b = \pm 1$, ou seja, supõe-se que a margem ρ é sempre maior que a distância entre os hiperplanos $\mathbf{w} \cdot \mathbf{x} + b = 0$ e $|\mathbf{w} \cdot \mathbf{x} + b = 1|$ (Campbell, 2000).

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \; se \; y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \; se \; y_i = -1 \end{cases} \tag{3.1}$$

para $i = 1, \ldots, n$.

Sejam $\mathbf{x}_1$ um ponto sobre $\mathbf{w}\cdot\mathbf{x}+b = -1$ e $\mathbf{x}_2$ um ponto sobre $\mathbf{w}\cdot\mathbf{x}+b = +1$ (Expressão 3.2). Supõe-se também que $\mathbf{x}_1$ intercepta a reta perpendicular a $\mathbf{x}_2$, como indicado na Figura 3.1 (Hearst et al., 1998). Tem-se assim que:

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_1 + b = -1 \\ \mathbf{w} \cdot \mathbf{x}_2 + b = 1 \end{cases} \tag{3.2}$$

A partir do sistema 3.2, obtém-se a Equação 3.3.

$$\mathbf{w} \cdot (\mathbf{x}_2 - \mathbf{x}_1) = 2 \tag{3.3}$$

Como $\mathbf{w}$ e $\mathbf{x}_2 - \mathbf{x}_1$ são ortogonais ao hiperplano separador, esses vetores são paralelos entre si. Pode-se desta forma deduzir a Equação 3.4.

$$|\mathbf{w} \cdot (\mathbf{x}_2 - \mathbf{x}_1)| = \|\mathbf{w}\| \times \|\mathbf{x}_2 - \mathbf{x}_1\| \tag{3.4}$$

em que $\|\cdot\|$ representa a norma de um vetor. Substituindo 3.3 em 3.4, obtém-se a Equação 3.5.

$$\|\mathbf{x}_2 - \mathbf{x}_1\| = \frac{2}{\|\mathbf{w}\|} \tag{3.5}$$

A norma $\|\mathbf{x}_2 - \mathbf{x}_1\|$ mede a distância entre os hiperplanos $\mathbf{w}\cdot\mathbf{x}+b = -1$ e $\mathbf{w}\cdot\mathbf{x}+b = +1$. Logo, pela Equação 3.5 pode-se afirmar que a distância entre os mesmos é dada por $2/\|\mathbf{w}\|$.

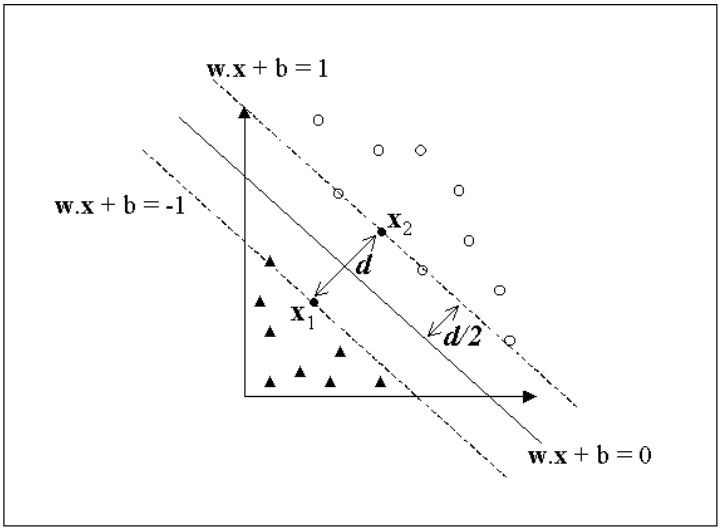

Figura 3.1: Ilustração da distância d entre os hiperplanos $\mathbf{w} \cdot \mathbf{x}_1 + b = -1$ e $\mathbf{w} \cdot \mathbf{x}_2 + b = +1$.

Da mesma forma, pode-se deduzir que a distância entre os hiperplanos $\mathbf{w} \cdot \mathbf{x} + b = 0$ e $\mathbf{w} \cdot \mathbf{x} + b = 1$ (ou $\mathbf{w} \cdot \mathbf{x} + b = -1$) é dada por $1/\|\mathbf{w}\|$. Como foi suposto que a margem é sempre maior que esta última distância, a minimização de $\|\mathbf{w}\|$ leva a uma maximização da margem.

O hiperplano ótimo é então definido para os valores de $\mathbf{w}$ e b que satisfazem as desigualdades 3.1 e para os quais a norma $\|\mathbf{w}\|$ é mínima. Este é um problema de otimização com restrições e pode ser reescrito como:

$$\textbf{Minimizar: } \|\mathbf{w}\|^2$$
$$\textbf{Sob as restrições: } y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 \geq 0, \text{ para } i = 1, \ldots, n$$

Este é um problema clássico em otimização denominado *programação quadrática*, para o qual há uma ampla e estabelecida teoria existente (Hearst et al., 1998). Na resolução do mesmo, introduz-se uma função Lagrangiana, definida em termos de $\mathbf{w}$ e b, apresentada em 3.6 (Campbell, 2000).

$$L\left(\mathbf{w}, b, \alpha\right) = \frac{1}{2}\left\|\mathbf{w}\right\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) - 1\right) \tag{3.6}$$

em que os α_i são denominados *multiplicadores de Lagrange*. O problema passa a ser então a minimização de 3.6 em relação a $\mathbf{w}$ e b e a maximização dos α_i. Os pontos ótimos desta equação são obtidos por meio da resolução das igualdades apresentadas em 3.7 e 3.8.

$$\frac{\partial L}{\partial b} = 0 \tag{3.7}$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \tag{3.8}$$

As Equações 3.7 e 3.8 levam ao resultado representado nas Expressões 3.9 e 3.10.

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.9}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \tag{3.10}$$

Substituindo 3.9 e 3.10 em 3.6, obtém-se o seguinte problema de otimização (denominado dual):

$$\textbf{Maximizar: } \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}$$
$$\textbf{Sob as restrições: } \begin{cases} \alpha_i \geq 0,\ i = 1,\ \ldots,\ n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

Calculado o valor ótimo α^*, $\mathbf{w}^*$ é encontrado por 3.10. Sendo determinada a direção do hiperplano ótimo (ou seja, $\mathbf{w}^*$), b^* pode ser calculado por 3.11.

$$\begin{aligned} b^* &= -\tfrac{1}{2} \left[\max_{\{i|y_i=-1\}} (\mathbf{w}^* \cdot \mathbf{x_i}) + \min_{\{i|y_i=+1\}} (\mathbf{w}^* \cdot \mathbf{x_i}) \right] \\ &= -\tfrac{1}{2} \left[\max_{\{i|y_i=-1\}} \left(\sum_{j=1}^{n} y_j \alpha_j^* \mathbf{x_i} \cdot \mathbf{x_j} \right) + \min_{\{i|y_i=+1\}} \left(\sum_{j=1}^{n} y_j \alpha_j^* \mathbf{x_i} \cdot \mathbf{x_j} \right) \right] \end{aligned} \tag{3.11}$$

Seja α^* o valor ótimo encontrado pela otimização do problema dual. É demonstrado que α_i^* assume valores positivos para exemplos do treinamento que estão a uma distância do hiperplano ótimo exatamente igual à margem. Para os outros padrões, o valor de α_i^* é nulo. Os exemplos para os quais $\alpha_i^* > 0$ são denominados *vetores suporte* (*Support Vectors* - SVs).

Fixados os valores dos parâmetros $\mathbf{w}^*$ e b^* por meio do treinamento de uma SVM, o classificador extraído é definido pela função 3.12.

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn}\Big(\sum_{\mathbf{x_i} \in SV} \alpha_i^* y_i \mathbf{x_i} \cdot \mathbf{x} + b^* \Big) = \begin{cases} +1 \ se \sum_{\mathbf{x_i} \in SV} \alpha_i^* y_i \mathbf{x_i} \cdot \mathbf{x} + b^* > 0 \\ -1 \ se \sum_{\mathbf{x_i} \in SV} \alpha_i^* y_i \mathbf{x_i} \cdot \mathbf{x} + b^* < 0 \end{cases} \tag{3.12}$$

A partir de 3.12, verifica-se que a classificação de um novo padrão $\mathbf{x}$ requer apenas a computação do produto interno entre $\mathbf{x}$ e cada SV. Logo, o hiperplano ótimo é determinado

unicamente pelos SVs, que por este motivo são considerados os exemplos mais informativos do conjunto de treinamento.

O classificador apresentado, porém, não é eficiente quando confrontado com bases de dados mais gerais, podendo se orientar por ruídos presentes nos dados. Supôs-se anteriormente que era possível determinar $\mathbf{w}$ e b tal que houvesse uma região entre $\mathbf{w}\cdot\mathbf{x}+b=0$ e $\mathbf{w}\cdot\mathbf{x}+b=\pm 1$ sem exemplos de treinamento. No caso de conjuntos mais gerais é mais difícil obedecer a esta restrição. Na tentatica de lidar com estes casos, introduz-se o conceito de variáveis de relaxamento ε. As variáveis de relaxamento suavizam as restrições impostas na determinação do hiperplano ótimo, sendo admitida com seu uso a permanência de alguns padrões entre as margens do classificador, assim como a ocorrência de alguns erros de classificação. A inclusão de variáveis de relaxamento é também denominada técnica de margens suaves (Cortes and Vapnik, 1995).

O problema de determinação do hiperplano ótimo é reformulado da seguinte maneira:

$$\textbf{Minimizar: } \|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\varepsilon_i$$

$$\textbf{Sob as restrições: } \begin{cases} \varepsilon_i \geq 0 \\ y_i\left(\mathbf{w}\cdot\mathbf{x_i}+b\right) \geq 1-\varepsilon_i \end{cases}$$

em que C é uma constante que impõe um peso diferente para o treinamento em relação à generalização.

Como no caso de conjuntos linearmente separáveis, chegou-se a um problema de otimização quadrática com restrições. A sua resolução envolve os mesmos passos matemáticos apresentados anteriormente. Os resultados obtidos também são semelhantes aos listados, com exceção das seguintes modificações:

- A restrição do problema dual $\alpha_i \geq 0$ é substituída por $0 \leq \alpha_i \leq C$, para $i = 1, \ldots, n$.
- Os padrões com multiplicador de Lagrange α_i maior que 0, ou seja, os SVs, são aqueles se encontram sobre as margens ou entre elas. Para o caso de SVs sobre as margens, o valor de α_i encontra-se no intervalo $0 < \alpha_i < C$. Para os padrões entre as margens, tem-se $\alpha_i = C$.

Os classificador linear com margens suaves ainda tem utilização limitada. Há muitos casos em que não é possível dividir satisfatoriamente os dados de treinamento por um hiperplano. Um exemplo é apresentado na Figura 3.2a, em que o uso de uma fronteira curva na separação das classes seria mais adequada.

Para generalizar as SVMs para lidar com essas situações, são definidas funções reais $\Phi_1, \ldots, \Phi_M$ no domínio do espaço de entrada. Por meio da utilização destas funções,

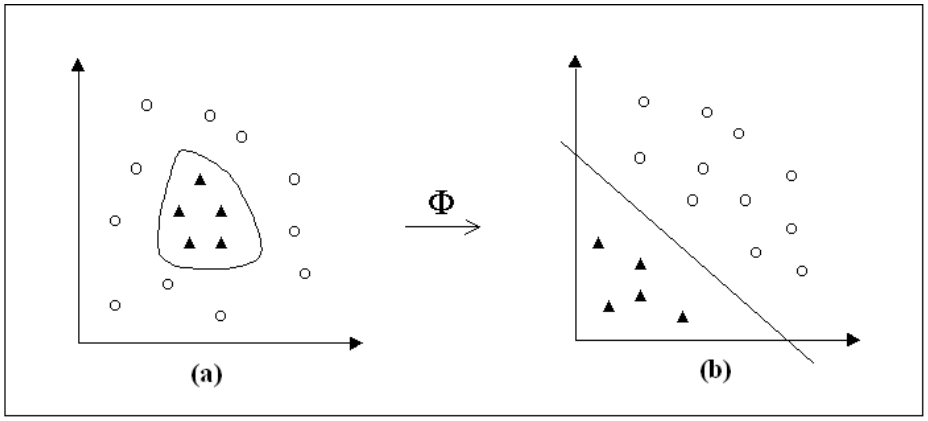

Figura 3.2: Transformação do conjunto de dados no espaço de entrada **(a)** para o espaço de características **(b)**.

mapea-se o conjunto de treinamento S (Equação 3.13) para um novo espaço, denominado *espaço de características*, da maneira apresentada em 3.14.

$$\begin{gathered} S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} \\ \text{em que } \mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \;\; e \; y_i \in \{-1, +1\} \end{gathered} \tag{3.13}$$

$$\begin{gathered} \mathbf{x}_i \;\; (i = 1, \ldots, n) \;\; \mapsto \;\; \mathbf{\Phi}(\mathbf{x}_i) = (\Phi_1(\mathbf{x}_i), \Phi_1(\mathbf{x}_i), \ldots, \Phi_M(\mathbf{x}_i)) \\ \Rightarrow \;\; \mathbf{\Phi}(S) = \{(\mathbf{\Phi}(\mathbf{x}_1), y_1), (\mathbf{\Phi}(\mathbf{x}_2), y_2), \ldots, (\mathbf{\Phi}(\mathbf{x}_n), y_n)\} \end{gathered} \tag{3.14}$$

Estas funções Φ_i podem ser não-lineares. Em particular, M pode ser muito maior que m, a dimensão do espaço de $\mathbf{x}$. Uma característica singular do espaço de características é que a escolha de uma função $\mathbf{\Phi}$ apropriada faz com que o conjunto de treinamento mapeado possa ser eficientemente separado por um hiperplano. As SVMs lineares com margens suaves anteriormente apresentadas podem então ser utilizadas sobre o conjunto de treinamento mapeado nesse espaço (Hearst et al., 1998). Este fato é ilustrado na Figura 3.2.

As adaptações realizadas nas SVMs lineares são apresentadas explicitamente no Algoritmo 3.1.

O problema de classificação passa a ser a avaliação da função 3.15.

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn}(\sum_{\mathbf{x_i} \in SV} \alpha_i^* y_i \mathbf{\Phi}(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x}) + b^*) = \begin{cases} +1 \;\; se \sum\limits_{\mathbf{x_i} \in SV} \alpha_i^* y_i \mathbf{\Phi}(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x}) + b^* > 0 \\ -1 \;\; se \sum\limits_{\mathbf{x_i} \in SV} \alpha_i^* y_i \mathbf{\Phi}(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x}) + b^* < 0 \end{cases} \tag{3.15}$$

Além disso, as considerações sobre os SVs apresentadas para a SVM linear com mar-

Algoritmo 3.1 SVM não linear (Vert, 2001).

1: Para qualquer conjunto de treinamento $\mathbf{\Phi}(S) = \{(\mathbf{\Phi}(\mathbf{x}_1), y_1), \ldots, (\mathbf{\Phi}(\mathbf{x}_n), y_n)\}$
2: Seja $\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*)$ a solução do seguinte problema de otimização com restrições:
3: Maximizar: $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathbf{\Phi}(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x_j})$
4: Sob as restrições: $\begin{cases} \sum_{i=1}^{n} y_i \alpha_i = 0 \\ 0 \leq \alpha_i \leq C, \ i = 1, \ \ldots, \ n \end{cases}$
5: O par $(\mathbf{w}^*, b^*)$ apresentado a seguir define o hiperplano ótimo.
6: $\mathbf{w}^* \leftarrow \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{\Phi}(\mathbf{x_i})$
7: $b^* \leftarrow -1/2 \left[\min_{\{i|y_i=+1\}} (\mathbf{w}^* \cdot \mathbf{\Phi}(\mathbf{x_i})) + \max_{\{i|y_i=-1\}} (\mathbf{w}^* \cdot \mathbf{\Phi}(\mathbf{x_i})) \right]$

gens suaves se mantêm para o caso não linear.

A partir das equações listadas no Algoritmo 3.1, percebe-se que a única informação necessária sobre o mapeamento $\mathbf{\Phi}$ é uma definição de como o produto interno $\mathbf{\Phi}(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x_j})$ pode ser realizado, para quaisquer $\mathbf{x_i}$ e $\mathbf{x_j}$ pertencentes ao espaço de entradas. Isto é obtido com a introdução do conceito de *Kernels*. Um *Kernel* K é uma função que recebe dois pontos $\mathbf{x_i}$ e $\mathbf{x_j}$ do espaço de entradas e computa o produto escalar $\mathbf{\Phi}(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x_j})$ no espaço de características, como descrito em 3.16 (Haykin, 1999).

$$K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{\Phi}(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x_j}) \tag{3.16}$$

De maneira geral, a função Kernel é mais simples que a do mapeamento. Em grande parte dos casos, $\mathbf{\Phi}$ assume formas bastante complexas. Por este motivo, é comum definir-se a função Kernel sem conhecer-se explicitamente o mapeamento $\mathbf{\Phi}$. A utilidade dos Kernels está, portanto, em sua simplicidade de cálculo e capacidade de representar espaços abstratos.

Alguns dos Kernels mais utilizados são os polinomiais, os Gaussianos ou RBF (*Radial-Basis Function*) e os Sigmoidais, apresentados na Tabela 3.1.

3.2 Perceptron com Margens Largas

Um dos primeiros algoritmos propostos para separação linear de um conjunto de dados foi o *Perceptron* de Rosenblatt (1962). O algoritmo Perceptron realiza mudanças incrementais nos parâmetros $(\mathbf{w}, b)$ de uma função linear (Equação 1.1) até que todos exemplos de treinamento sejam corretamente classificados. Um pseudocódigo do Perceptron é apresentado no Algoritmo 3.2.

A convergência do Perceptron é garantida para conjuntos linearmente separáveis. Não

Tipo de Kernel	**Função $K(\mathbf{x_i}, \mathbf{x_j})$ correspondente**	**Comentários**
Polinomial	$\left(\mathbf{x_i}^T \cdot \mathbf{x_j} + 1\right)^p$	A potência p deve ser especificada pelo usuário
Gaussiano	$\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2\right)$	A amplitude σ^2 é especificada pelo usuário
Sigmoidal	$\tanh\left(\beta_0 \mathbf{x_i} \cdot \mathbf{x_j} + \beta_1\right)$	Utilizado somente para alguns valores de β_0 e β_1

Tabela 3.1: Sumário dos principais Kernels utilizados nas SVMs (Haykin, 1999).

Algoritmo 3.2 Perceptron básico (Smola et al., 1999b).

Dados um conjunto de treinamento $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, em que $\mathbf{x}_i \in \Re^m$ e $y_i \in \{-1, +1\}$, e uma taxa de aprendizado η
Inicializar $\mathbf{w}, b = 0$
repita
  para $i = 1, \ldots, n$ **faça**
    computar $g(\mathbf{x}_i) = \text{sgn}(\mathbf{w} \cdot \mathbf{x}_i + b)$
    se $g(\mathbf{x}_i) \neq y_i$ **então**
      atualizar $(\mathbf{w}, b)$ de acordo com:
      $\mathbf{w}' = \mathbf{w} + \eta/2 \left(y_i - g\left(\mathbf{x}_i\right)\right) \mathbf{x}_i$
      $b' = b + \eta/2 \left(y_i - g\left(\mathbf{x}_i\right)\right)$
    fim-se
  fim-para
até que $g(\mathbf{x}_i) = y_i$ para todo $i = 1, \ldots, n$
Retornar $(\mathbf{w}, b)$

há garantia, porém, que o hiperplano obtido será aquele para o qual a margem de separação em relação aos dados é máxima. Este algoritmo, portanto, não leva em conta o princípio de minimização do risco estrutural provido pela TAE.

Freund and Schapire (1998) propuseram uma generalização do algoritmo Perceptron denominada *"Voted-Perceptron"* (VP), em que o erro funcional é limitado pelo inverso da margem do classificador, semelhante aos limites apresentados por Vapnik (1995).

Sejam $f_i(\mathbf{x}) = \mathbf{w}_i \cdot \mathbf{x}$ funções lineares passando pela origem. O algoritmo VP mantém todos os vetores $\mathbf{w}_i$ gerados durante o treinamento do Perceptron. Para cada vetor $\mathbf{w}_i$, mantém-se também um contador c_i, o qual armazena o número de iterações que o vetor é "mantido" até que um erro seja cometido. Este número constitui um peso atribuído à função definida por $\mathbf{w}_i$. O Algoritmo 3.3 apresenta um pseudocódigo do VP.

Dado um novo padrão $\mathbf{x}$, sua classe é fornecida pelas predições de cada função $f_i(\mathbf{x}) = \mathbf{w}_i \cdot \mathbf{x}$, ponderadas por seus respectivos pesos c_i, como definido na Equação 3.17. Esta estratégia reflete o fato de que boas funções $f_i(\mathbf{x})$ tendem a gerar predições corretas por

Algoritmo 3.3 *Voted-Perceptron* (Freund and Schapire, 1998).

1: Dados um conjunto de treinamento $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, em que $\mathbf{x}_i \in \Re^m$ e $y_i \in \{-1, +1\}$, e um número de ciclos T
2: Inicializar $k, \mathbf{w}_1, c_1 = 0$.
3: **repita**
4: **para** $i = 1, \ldots, n$ **faça**
5: computar $g(\mathbf{x}_i) = \text{sgn}(\mathbf{w}_k \cdot \mathbf{x}_i)$
6: **se** $g(\mathbf{x}_i) = y_i$ **então**
7: $c_k = c_k + 1$
8: **senão**
9: $\mathbf{w}_{k+1} = \mathbf{w}_k + y_i \mathbf{x}_i$
10: $c_{k+1} = 1$
11: $k = k + 1$
12: **fim-se**
13: **fim-para**
14: t = t + 1
15: **até que** $t = T$
16: Retornar a lista $\langle (\mathbf{w}_1, c_1), \ldots, (\mathbf{w}_k, c_k) \rangle$

mais iterações do algoritmo e por isso devem ter pesos maiores na votação final.

$$g(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{k} c_i \text{sgn}(\mathbf{w}_i \cdot \mathbf{x})\right) \tag{3.17}$$

O Algoritmo 3.3 tem convergência garantida para conjuntos linearmente separáveis. Para generalizar o VP a problemas não linerares, os autores proprõem o uso de funções Kernel (Equação 3.16), semelhante ao discutido para o caso das SVMs (Cortes and Vapnik, 1995). Para tal, as computações envolvendo os padrões devem ser escritas em termos de produtos escalares. Isto é realizado decompondo cada $\mathbf{w}_k$ como a soma das instâncias que foram utilizadas em sua criação. A Equação 3.18 representa esta informação.

$$\mathbf{w}_k = \sum_{j=1}^{k-1} y_{i_j} \mathbf{x}_{i_j} \tag{3.18}$$

para índices i_j apropriados. A partir da Equação 3.18, o produto interno de $\mathbf{w}_k$ com $\mathbf{x}$ pode ser reescrito como:

$$\mathbf{w}_k \cdot \mathbf{x} = \sum_{j=1}^{k-1} y_{i_j} \left(\mathbf{x}_{i_j} \cdot \mathbf{x}\right) \tag{3.19}$$

Pode-se então substituir os produtos internos $\mathbf{x}_{i_j} \cdot \mathbf{x}$ por funções Kernel $K(\mathbf{x}_{i_j}, \mathbf{x})$, não-linearizando a solução dada pelo algoritmo VP.

Nos diversos experimentos realizados pelos autores Freund and Schapire (1998), o

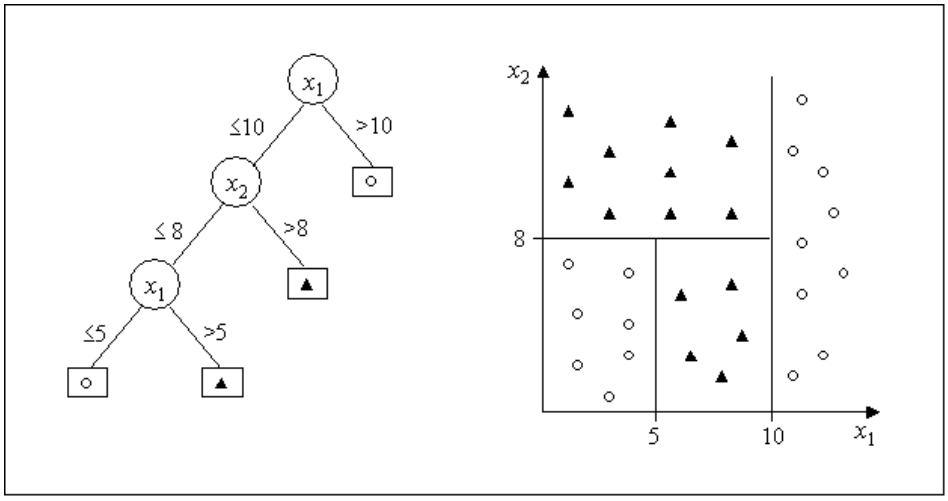

Figura 3.3: Regiões retângulares formadas por uma AD tradicional no espaço de entradas.

desempenho do VP foi superior ao do Perceptron tradicional, porém se mostrou inferior ao das SVMs. Foram testadas também diferentes formas de combinação das funções $f_i(\mathbf{x})$ geradas pelo VP além da representada na Equação 3.17.

3.3 Árvores de Decisão com Margens Largas

As Árvores de Decisão (ADs) são estruturas recursivas compostas de nós e ramificações (Baranauskas and Monard, 2000). Os nós podem ser de dois tipos: de decisão ou folhas. Nós folhas são aqueles pertencentes à base da árvore e representam as classes do problema. Os nós de decisão contêm testes aplicados aos dados. Cada resultado de um teste forma uma ramificação e uma subárvore com mesma estrutura da árvore total.

Nos modelos de ADs mais comuns, os testes realizados em cada nó de decisão são aplicados aos atributos dos dados, possuindo a seguinte forma geral (Baranauskas and Monard, 2000):

$$x_{ij} \text{ op valor} \tag{3.20}$$

em que x_{ij} representa o valor do j-ésimo atributo de $\mathbf{x}_i$, um padrão do conjunto de treinamento, op é um operador do tipo $\{>, \geq, <, \leq\}$ e valor é uma constante válida para o atributo j.

Geometricamente, essas árvores particionam o espaço de entradas em regiões retângulares, por meio de hiperplanos ortogonais ao eixo do atributo em que o teste é aplicado. A Figura 3.3 ilustra um exemplo em que $\mathbf{x} = (x_1, x_2) \in \Re^2$.

Há diversas variações neste modelo. Uma delas se dá no uso de uma combinação linear dos atributos em cada nó de decisão (Bennet et al., 2000). Os testes nos nós de decisão

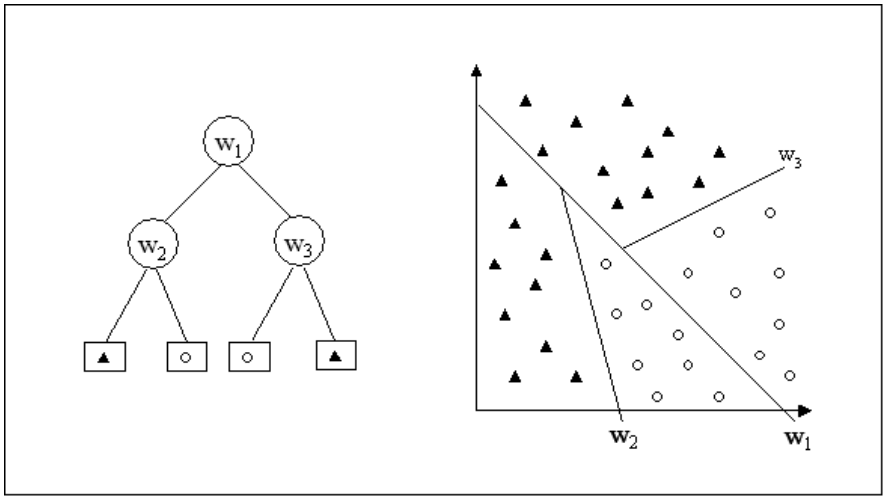

Figura 3.4: Regiões poliedrais formadas por uma AD Perceptron no espaço de entradas.

passam a ter a seguinte forma geral:

$$w_1 x_{i1} + w_2 x_{i2} + \ldots + w_m x_{im} + b \text{ op valor } \equiv \mathbf{w} \cdot \mathbf{x}_\mathrm{i} + b \text{ op valor} \tag{3.21}$$

em que w_j são constantes, x_{ij} são os atributos de $\mathbf{x}_i$, op é um operador do tipo $\{>, \geq, <, \leq\}$ e valor é uma constante.

Estes testes dividem o espaço de entradas através de hiperplanos com orientações gerais, formando partições poliedrais. Este fato é ilustrado na Figura 3.4. Este tipo de AD possui várias nomenclaturas, entre elas Árvores de Decisão Perceptron (ADP), pois pode-se considerar que cada nó da árvore obtida constitui um Perceptron (Utgoff, 1989).

As ADPs são árvores extremamente flexíveis. As ADs "tradicionais" apresentadas anteriormente, por exemplo, podem ser consideradas um caso especial de ADP. Devido a sua flexibilidade, essas árvores são sucetíveis a problemas de *overfitting*. Para lidar com esta situação, uma estratégia comumente aplicada é a poda, que elimina nós redundantes ou pouco expressivos frente ao custo de mantê-los. O controle da capacidade das ADPs, ou seja, de sua complexidade, é então normalmente realizado por meio do controle do tamanho das árvores induzidas. Bennet et al. (2000) propõem que o termo de capacidade dessas árvores também seja reduzido com a maximização das margens dos hiperplanos obtidos em cada nó, baseados nas motivações apresentadas pela TAE. Estes autores mostram resultados teóricos e empíricos comprovando que a estratégia de maximização de margens também é eficiente para as ADPs.

Para obtenção das ADPs com margens largas, Bennet et al. (2000) propuseram três algoritmos. Estes realizam generalizações em um sistema de indução de ADPs denominado OC1 (Murthy et al., 1994). Iniciando pelo nó raiz, o algoritmo OC1 escolhe o hiperplano

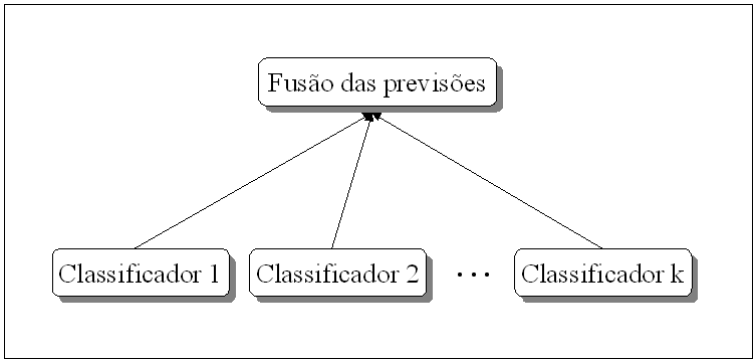

Figura 3.5: Ilustração de um comitê de classificadores (Sharkey, 1996).

que maximiza alguma medida de "impureza" em cada nó de decisão, como, por exemplo, o ganho de informação. Este procedimento procede até chegar a um nó folha.

O primeiro algoritmo desenvolvido para generalizar o OC1, denominado FAT, utiliza a árvore produzida por este algoritmo e maximiza suas margens. Para obtenção do hiperplano de margens largas em cada nó, utiliza-se uma SVM linear.

O segundo algoritmo proposto, denominado MOC1, modifica o critério de impureza utilizado na separação dos dados em cada nó, considerando diretamente o tamanho da margem.

O último algoritmo, denominado MOC2, também altera a medida de impureza utilizada pelo OC1. A exemplo do MOC1, incorpora-se diretamente o conceito de maximização de margens. Porém, neste caso, utiliza-se uma estratégia de margens suaves, a exemplo das SVMs.

Entre os algoritmos descritos para indução de ADPs com margens largas, não houve algum que se destacasse como melhor em todas aplicações. Deve-se ressaltar também que em todos os casos a poda é mantida após a obtenção da ADP.

As ADPs de margens largas, além de obter ganhos em termos de generalização, mantêm outras características atrativas das ADs em geral. Entre elas, destacam-se sua compreensibilidade, facilidade de manutenção, flexibilidade e velocidade.

3.4 Boosting

O *Boosting* é uma técnica utilizada para melhorar o desempenho de algoritmos de aprendizado (Schapire, 1999). Ela é baseada na combinação de classificadores produzidos por outras técnicas, tais como ADs (Drucker and Cortes, 1996) e Redes Neurais Artificiais (RNAs) (Schwenk and Bengio, 2000). O *Boosting* gera, portanto, comitês de classificadores.

Um comitê (Figura 3.5) consiste de um conjunto de preditores cujas saídas são combinadas na previsão da classe de novas instâncias (Baranauskas and Monard, 2000). A motivação desse tipo de classificador está na idéia de que a combinação das saídas de diversos preditores pode melhorar o desempenho obtido por um único preditor. Existem diversas outras técnicas para geração de comitês. Entre elas tem-se o *Windowing* (Quinlan, 1993) e o *Bagging* (Breiman, 1996). O *Boosting* se destaca pelas altas taxas de generalização alcançadas, e também por sua simplicidade.

O princípio básico do *Boosting* está em realizar chamadas a um algoritmo de aprendizado base, também denominado "fraco", sobre um conjunto de treinamento continuamente modificado. A modificação consiste na atribuição de pesos às instâncias do mesmo, segundo uma distribuição de probabilidade que é atualizada a cada iteração do algoritmo, enfatizando exemplos que tenham sido classificados erroneamente. Na geração do classificador final, realiza-se uma combinação ponderada das diversas hipóteses geradas.

A idéia de se combinar preditores "fracos", que se comportam um pouco melhor que uma previsão randômica, para produção de classificadores mais precisos tem suas raízes nos trabalhos de Kearns and Valiant (1988, 1994) sobre o modelo de aprendizado PAC (*Probably Approximately Correct*). O primeiro algoritmo polinomial para realização dessa tarefa foi obtido por Schapire (1990). Posteriormente, Freund (1995) propôs o *AdaBoost*, que resolveu grande parte dos problemas dos algoritmos de *Boosting* anteriores.

O *AdaBoost* (AB) recebe como entrada um conjunto de treinamento S composto de pares $(\mathbf{x}_i, y_i)$, em que $\mathbf{x} \in \Re^n$ e $y \in \{-1, +1\}$ e faz T chamadas a um algoritmo base, em que T é um número máximo de iterações (ciclos), único parâmetro a ser ajustado.

Para atribuição de pesos aos padrões de treinamento, o algoritmo mantém uma distribuição de probabilidade D, conforme pode ser observado no pseudocódigo apresentado no Algoritmo 3.4. O peso de uma instância $\mathbf{x}_i$ no ciclo t é denotado por $D_t(\mathbf{x}_i)$. Inicialmente, a distribuição D é uniforme sobre todo o conjunto de treinamento. A cada ciclo t do algoritmo, o valor do peso associado a exemplos classificados incorretamente é aumentado, fazendo com que o algoritmo base foque nos exemplos mais difíceis de serem classificados.

Para implementação do uso da distribuição D, o algoritmo base deve ser capaz de receber entradas ponderadas. Caso isto não seja possível, pode-se utilizar o recurso de escolher amostras com repetições.

Na geração da função de decisão final, pondera-se as hipóteses h_t segundo constantes α_t, calculadas a partir do erro ϵ_t cometido pelas mesmas ($\alpha_t \geq 0$ se $\epsilon_t \leq 1/2$ e é maior à medida que ϵ_t diminui).

Freund and Schapire (1997) provaram um limite na capacidade de generalização do AB em função do erro cometido para o conjunto de treinamento, do tamanho deste conjunto n, da dimensão VC h da hipótese final e do número de ciclos T do algoritmo. Este limite

Algoritmo 3.4 *AdaBoost* (Schapire, 1999)

1: Dado um conjunto de treinamento $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
2: Inicializar $D_t(\mathbf{x}_i) \leftarrow \frac{1}{n}$ e $t \leftarrow 1$
3: **repita**
4: Treinar o algoritmo base utilizando a distribuição D_t
5: Calcular o erro ϵ_t da hipótese h_t gerada

$$\epsilon_t \leftarrow \Pr_{i \sim D_t}\left[h_t\left(\mathbf{x}_i\right) \neq y_i\right] = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t\left(\mathbf{x}_i\right)$$

6: Escolher $\alpha_t \leftarrow \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ e $Z_t \leftarrow \sum_{i=1}^{n} D_t\left(\mathbf{x}_i\right)\exp\left(-\alpha_i y_i h_t\left(\mathbf{x}_i\right)\right)$, em que Z_t é um termo normalizador, escolhido tal que $\sum_{t=1}^{T} D_{t+1}\left(\mathbf{x}_i\right) = 1$
7: Atualizar D_t:

$$\begin{aligned} D_{t+1}\left(\mathbf{x}_i\right) &\leftarrow \frac{D_t(\mathbf{x}_i)}{Z_t} \times \begin{cases} \exp\left(-\alpha_t\right) & se\ h_t\left(\mathbf{x}_i\right) = y_i \\ \exp\left(\alpha_t\right) & se\ h_t\left(\mathbf{x}_i\right) \neq y_i \end{cases} \\ &= \frac{D_t(\mathbf{x}_i)\exp(-\alpha_i y_i h_t(\mathbf{x}_i))}{Z_t} \end{aligned}$$

8: $t = t + 1$
9: **até que** $t = T$
10: Hipótese Final:

$$H\left(\mathbf{x}\right) \leftarrow sign\left(\sum_{t=1}^{T} \alpha_t h_t\left(\mathbf{x}\right) \Big/ \sum_{t=1}^{T} \alpha_t\right)$$

é apresentado na Equação 3.22.

$$\hat{P}r\left[H\left(\mathbf{x}\right) \neq y\right] + \tilde{O}\left(\sqrt{\frac{Th}{n}}\right) \tag{3.22}$$

em que $\hat{P}r$ denota a probabilidade empírica para o conjunto de treinamento e $\tilde{O}$ é uma notação de ordem em que funções logarítmicas e constantes são omitidas.

Segundo o limite fornecido em 3.22, se um valor muito grande for atribuído ao número de ciclos T, pode ocorrer *overfitting*. Embora isto ocorra em algumas situações, experimentos realizados por Breiman (1998), Drucker and Cortes (1996) e Quinlan (1996) sugerem que o AB não leva a *overfitting* mesmo quando executado por milhares de ciclos. Em certas ocasiões, o erro de generalização continua a cair após o erro de treinamento ser nulo, o que contradiz o que é expresso na Equação 3.22.

O trabalho de Schapire et al. (1997) provê então uma análise alternativa da capacidade de generalização do AB, em termos de margens. No caso do AB, a margem ρ pode ser definida por 3.23.

$$\rho\left(\mathbf{x}, y\right) = \frac{y f\left(\mathbf{x}\right)}{\sum_{t=1}^{T} |\alpha_t|} = \frac{y \sum_{t=1}^{T} \alpha_t h_t\left(\mathbf{x}\right)}{\sum_{t=1}^{T} |\alpha_t|} \tag{3.23}$$

A função $\rho(\mathbf{x}, y)$ assume valores no intervalo $[-1, +1]$, e é positiva se H classifica corretamente o exemplo $\mathbf{x}$. O limite na generalização do AB pode ser então reescrito como 3.24.

$$\hat{Pr}\left[\rho\left(\mathbf{x}, y\right) \geq \theta\right] + \tilde{O}\left(\sqrt{\frac{h}{n\theta^2}}\right) \tag{3.24}$$

para um $\theta > 0$ qualquer. O termo $\hat{Pr}[\rho(\mathbf{x}, y) \geq \theta]$ refere-se ao erro marginal sobre o conjunto de treinamento, definido na Seção 2.3. Satisfazendo os resultados empíricos anteriormente descritos, a Equação 3.24 não depende do número de ciclos T.

Os autores Schapire et al. (1997) também demonstraram que o AB realiza uma maximização implícita das margens, já que ele enfatiza os exemplos mais próximos à fronteira de decisão, os quais em geral são mais difíceis de classificar.

Apesar do AB, como apresentado no Algoritmo 3.4, ter se mostrado eficiente em diversas aplicações práticas, sua precisão é afetada pela presença de ruídos nos dados. Quando aplicado sob bases de dados ruidosas, este algoritmo obtém um baixo desempenho, levando a problemas de generalização. Isto ocorre devido ao artifício de se enfatizar exemplos mais difíceis de serem classificados, já que os ruídos também podem ser caracterizados como tal.

Para minimizar a influência de padrões ruidosos nos resultados do AB, extensões são propostas visando permitir alguns erros durante o treinamento, de forma similar ao que é realizado em relação às SVMs. Introduz-se então o conceito de margens suaves para o AB. Entre os trabalhos que utilizam este princípio está o *AdaBoost Regularizado* de Rätsch et al. (1998). Este algoritmo possui um termo de regularização, que define a influência de um padrão sobre as hipótese combinadas, e é definido pela Equação 3.25.

$$\mu_t\left(\mathbf{x}_i\right) = \sum_{j=1}^{t} \alpha_j D_j\left(\mathbf{x}_i\right) \tag{3.25}$$

Um padrão classificado incorretamente com freqüência (difícil de classificar) tem um valor de μ alto (alta influência).

A partir dessa definição, a margem suave de um padrão $\tilde{\rho}\left(\mathbf{x}_i, y_i\right)$ é definida por 3.26, e representa um compromisso entre a margem ρ e a influência do padrão na hipótese final.

$$\tilde{\rho}\left(\mathbf{x}_i, y_i\right) = \rho\left(\mathbf{x}_i, y_i\right) + C\mu\left(\mathbf{x}_i\right)^p \tag{3.26}$$

em que p e C são constantes fixadas empiricamente. Para $C = 0$, tem-se o algoritmo AB original.

A modificação introduzida no Algoritmo 3.4 visando acomodar o conceito de margens

suaves se dá na atualização de $D_t(\mathbf{x}_i)$. Para $p = 1$, obtém-se 3.27 (Rätsch et al., 1998).

$$D_{t+1}(\mathbf{x}_i) = D_t(\mathbf{x}_i) \exp\{\alpha_t I(y_i \neq h_t(\mathbf{x}_i)) - C\mu_t(\mathbf{x}_i)|\alpha_t|\} \tag{3.27}$$

em que $I(verdadeiro) = 1$ e $I(falso) = 0$.

3.5 Discriminante de Fisher com Kernels

O Discriminante de Fisher com Kernels (*Kernel Fisher Discriminant* - KFD) (Mika et al., 1999) generaliza o algoritmo Discriminante de Fisher (Fisher, 1936) para o caso não-linear, através do artifício do uso de funções Kernel, em uma estratégia semelhante à utilizada pelas SVMs (Boser et al., 1992) e pelo VP (Freund and Schapire, 1998).

O Discriminante de Fisher (*Fisher Discriminant* - FD) busca projeções lineares dos dados tal que a separação entre as classes presentes seja a maior possível (Müller et al., 2001). O FD pode ser considerado, na prática, um método para extração de características dos dados, em que se procura o melhor conjunto de características capaz de discriminar diferentes classes (Mika et al., 1999).

Para realizar esta tarefa, deve-se maximizar o coeficiente de *Rayleigh*, definido pela Equação 3.28.

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{3.28}$$

Este coeficiente mede a separabilidade entre classes por meio de duas quantidades: a dispersão entre as diferentes classes (numerador da Equação 3.28) e a dispersão dentro das classes (denominador de 3.28). A dispersão entre as classes indica quão distantes estão as médias das projeções dos dados de cada classe. Para uma maior separabilidade, ela deve ser maximizada. A dispersão dentro das classes, por sua vez, mede a variabilidade entre as projeções em uma mesma classe e deve ser minimizada. Dado um problema com duas classes, $\mathbf{S}_B$ e $\mathbf{S}_W$ são definidos pelas Equações 3.29 e 3.30, respectivamente.

$$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \tag{3.29}$$

em que $\mathbf{m}_i$ é a média dos dados da classe i, sendo definida pela Equação 3.31.

$$\mathbf{S}_W = \sum_{i=1,2} \sum_{\mathbf{x} \in X_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T \tag{3.30}$$

onde X_i é o conjunto de exemplos da classe i.

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbf{x}_j^i \tag{3.31}$$

em que n_i representa o número de exemplos da classe i.

Para não-linearizar a técnica FD, aplica-se um mapeamento $\mathbf{\Phi}$ aos dados de treinamento, de forma similar ao feito para as SVMs (Cristianini and Shawe-Taylor, 2000). As Equações 3.28 a 3.31 podem ser então reescritas como (Mika et al., 1999):

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B^{\mathbf{\Phi}} \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^{\mathbf{\Phi}} \mathbf{w}} \tag{3.32}$$

$$\mathbf{S}_B = \left(\mathbf{m}_1^{\mathbf{\Phi}} - \mathbf{m}_2^{\mathbf{\Phi}}\right) \left(\mathbf{m}_1^{\mathbf{\Phi}} - \mathbf{m}_2^{\mathbf{\Phi}}\right)^T \tag{3.33}$$

$$\mathbf{S}_W = \sum_{i=1,2} \sum_{\mathbf{x} \in X_i} \left(\mathbf{\Phi}(\mathbf{x}) - \mathbf{m}_i^{\mathbf{\Phi}}\right) \left(\mathbf{\Phi}(\mathbf{x}) - \mathbf{m}_i^{\mathbf{\Phi}}\right)^T \tag{3.34}$$

$$\mathbf{m}_i^{\mathbf{\Phi}} = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbf{\Phi}(\mathbf{x}_j^i) \tag{3.35}$$

A utilização de funções Kernel sobre este problema requer que as Equações 3.32 a 3.35 sejam ainda reformuladas em função de produtos internos entre os dados. Seja F o espaço de características dado pelo mapeamento $\mathbf{\Phi}$. A teoria de Kernels afirma que a solução $\mathbf{w} \in F$ deve ser definida em função dos exemplos de treinamento mapeados neste espaço, como representado na Equação 3.36.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{\Phi}(\mathbf{x}_i) \tag{3.36}$$

Através de manipulações algébricas, o problema de otimização do KFD no espaço de características pode então ser definido como:

$$J(\alpha) = \frac{\alpha^T \mathbf{M} \alpha}{\alpha^T \mathbf{N} \alpha} \tag{3.37}$$

em que $\mathbf{M} = (\mathbf{M}_1 - \mathbf{M}_2)(\mathbf{M}_1 - \mathbf{M}_2)^T$, com $(\mathbf{M}_i)_j = {}^1\!/_{n_i} \sum_{k=j}^{n_i} K(\mathbf{x}_j, \mathbf{x}_k^i)$ (K é a matriz Kernel), e $\mathbf{N} = \sum_{j=1,2} K_j \left(\mathbf{I} - \mathbf{1}_{n_j}\right) K_j^T$, com $(K_j)_{mn} = K(\mathbf{x}_n, \mathbf{x}_m^j)$ (matriz Kernel para a classe j), $\mathbf{I}$ igual à matriz identidade e $\mathbf{1}_{n_j}$ igual a uma matriz com entradas ${}^1\!/_{n_j}$.

Há um problema na definição acima decorrente do fato da dimensão de F ser igual ou maior que o número de exemplos n, o que torna o problema mal condicionado. Para solucionar esta questão, uma regularização é realizada adicionando-se um múltiplo da matriz identidade a $\mathbf{N}$ (Equação 3.38).

$$\mathbf{N}_\mu = \mathbf{N} + \mu\mathbf{I} \tag{3.38}$$

Resolvido o problema de otimização do KFD, a projeção de um novo padrão $\mathbf{x}$ em $\mathbf{w}$ é dada pela Equação 3.39.

$$(\mathbf{w} \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \tag{3.39}$$

Para usar esta projeção na classificação do novo padrão, deve-se estabelecer um limiar que será utilizado para a separação entre as classes. Este pode ser obtido pela média das projeções médias das duas classes ou pelo treinamento de, por exemplo, uma SVM linear sobre as projeções obtidas pelo KFD.

O KFD pode ser solucionado encontrando-se o autovetor α com maior autovalor λ no sistema $\mathbf{M}\alpha = \lambda\mathbf{N}\alpha$ ou computando-se $\alpha = \mathbf{N}^{-1}(\mathbf{M}_2 - \mathbf{M}_1)$ (Mika et al., 1999). Estes métodos, porém, são computacionalmente caros para grandes quantidades de dados, sendo factíveis apenas para pequenos valores de n.

Para lidar com este problema, Mika et al. (2001a) propuseram uma reformulação do KFD como um problema de otimização quadrática. Este problema, definido abaixo, realiza a minimização da variância dos dados projetados enquanto maximiza a distância entre a saída média de cada classe, objetivos primordiais da técnica FD.

$$\min_{\alpha,b,\xi} \|\xi\|^2 + CP(\alpha)$$

$$\textbf{Sob as restrições:} \quad \begin{array}{c} K\alpha + \mathbf{1}b = y + \xi \\ \mathbf{1}_k^T \xi = 0, \ \text{para k} = 1, 2 \end{array}$$

em que $\alpha, \xi \in \Re^n$ e $b, c \in \Re$. P representa o termo regularizador e $(\mathbf{1}_k)_i$ é igual a 1 caso y_i pertença à classe k e 0 caso contrário. A primeira restrição pode ser lida como $(\mathbf{w} \cdot \mathbf{x}_i) + b = y_i + \xi_i$, $i = 1, \ldots, n$ e "força" que a saída de cada exemplo seja sua classe desejada. O termo $\|\xi\|^2$ representa minimização da variância dos erros cometidos, enquanto as restrições $\mathbf{1}_k^T \xi = 0$ asseguram que a média das saídas de cada classe seja o rótulo desejado.

Esta formulação permitiu a derivação de algoritmos de treinamento mais eficientes para o KFD. Um desses métodos, denominado *sparse-greedy* (Mika et al., 2001b), constrói a solução do problema iterativamente. Começando por uma solução vazia, adiciona-se a

cada iteração um novo padrão na expansão 3.36. Escolhe-se este padrão por meio de heurísticas. Um exemplo de heurística constitui em escolher o padrão que proporciona o maior decréscimo na função objetivo. Termina-se o algoritmo quando a mudança na função objetivo se torna menor que um limiar pré-estabelecido.

O KFD possui desempenho similar ao das SVMs, apresentando, em geral, boa capacidade de generalização. Porém, diferente das SVMs, não se pode definir o conceito de margem explicitamente para esta técnica. Este fato deve ser cautelosamente observado, já que o bom desempenho das SVMs (bem como dos outros LMCs apresentados) é atribuído ao conceito de maximização de margens. Implicitamente, porém, pode-se considerar que o KFD maximiza a margem média de separação entre as classes, definida como a distância média entre exemplos de classes diferentes (Mika et al., 2001a).

3.6 Considerações Finais

Este Capítulo apresentou uma breve revisão sobre alguns dos principais LMCs descritos na literatura, algoritmos estes que, implícita ou explicitamente, realizam uma maximização da margem de separação entre dados pertencentes a classes distintas.

Além das técnicas citadas, limites baseados em margens também foram demonstrados para classificadores Bayesianos (Cristianini et al., 1997), justificando a resistência desses preditores à ocorrência de *overfitting*. Maiores detalhes sobre esta técnica e as análises realizadas podem ser consultadas em (Cristianini et al., 1997; Cristianini and Shawe-Taylor, 1999).

Barlett (1998) também obteve limites no erro de Redes Neurais Artificiais (RNAs) do tipo perceptron multicamadas (*Multilayer Perceptron* - MLP) baseados na margem alcançada na separação dos dados de treinamento e na magnitude dos pesos das conexões da rede. O principal resultado observado nesta pesquisa foi que, para o caso em que a margem é grande e os pesos das conexões são pequenos, o número de nós internos da rede pode se tornar irrelevante na complexidade da função induzida pela RNA e consequentemente, em sua capacidade de generalização. Estes limites justificam resultados experimentais em que RNAs do tipo MLP tendem a resistir mais que o esperado à ocorrência de *overfitting* a medida que mais conexões são adicionadas à rede. Detalhes sobre essas análises podem ser consultados em (Barlett, 1998).

Capítulo 4

LMCs para Várias Classes

As descrições das técnicas apresentadas no Capítulo 3 se limitaram a problemas com duas classes. Embora os LMCs apresentados sejam originalmente propostos para problemas binários, eles podem ser generalizados a problemas com mais classes (multiclasses).

Em um problema multiclasses, o conjunto de treinamento é composto por pares $(\mathbf{x_i}, y_i)$, tal que $y_i \in 1, \ldots, k$. Este Capítulo aborda soluções para problemas em que $k > 2$.

Deve-se observar que, no caso das ADPs com margens largas, esta generalização é direta, uma vez que os algoritmos para indução de árvores realizam classificações multiclasses diretamente.

Para o caso das SVMs, pode-se também reformular o problema de otimização de forma que ele acomode a realização de classificações multiclasses (Mayoraz and Alpaydin, 1998; Weston and Watkins, 1998). Porém, o este tipo de abordagem ainda não é amplamente utilizado. Para esta e as outras técnicas apresentadas (com exceção das ADPs), métodos que utilizam uma combinação de preditores binários na solução de problemas multiclasses são usualmente empregados.

As Seções 4.1, 4.2 e 4.3 descrevem brevemente três abordagens usuais para realização dessa tarefa, denominadas, respectivamente, "decomposição um-contra-todos", "decomposição todos-contra-todos" e "método de códigos de correção de erros".

4.1 Decomposição um-contra-todos

Para a solução de um problema multiclasses a partir de preditores binários, uma abordagem usual consiste na geração de k classificadores binários, onde k é o número de classes (Weston and Watkins, 1998). Cada preditor é responsável por distinguir uma classe i das demais. Esta metodologia é denominada decomposição "um-contra-todos" (1-c-t) (Mayoraz and Alpaydin, 1998).

Na predição da classe de um padrão $\mathbf{x}$, basta escolher a saída com valor máximo entre os k classificadores, conforme apresentado na Equação 4.1.

$$f(\mathbf{x}) = \arg\max_{1 \leq i \leq k} (f_i(\mathbf{x})) \tag{4.1}$$

4.2 Decomposição todos-contra-todos

Outra abordagem para solução de problemas multiclasses a partir de classificadores binários envolve a construção de $k(k-1)/2$ preditores, separando cada classe de outra. Este método é denominado "decomposição todos-contra-todos" (t-c-t).

Para unir os classificadores gerados, Friedman (1996) propôs o uso de um esquema de votação por maioria, em que cada um dos classificadores fornece uma classe como resultado. A solução final é dada pela classe que recebeu mais indicações.

Platt et al. (2000) argumentam que o esquema de votação por maioria, assim como o método 1ct, não provêem limites no erro de generalização. Além disso, principalmente no caso da decomposição tct, o tamanho dos classificadores gerados é em geral grande e a avaliação de seu resultado pode ser lenta.

Na resolução desses problemas, Platt et al. (2000) sugerem a utilização de um grafo direcionado acíclico[1] (*Directed Acyclic Graph* - DAG), de forma que o problema de agrupamento em várias classes é decomposto em diversas classificações binárias em cada nó do grafo.

A Figura 4.1 apresenta um exemplo de uma DAG para quatro classes. O processamento feito pelo DAG é equivalente a operação de uma lista. Inicialmente, a lista é composta por todas as classes. Em cada nó é gerado um classificador binário que separa os elementos da primeira e última classe da lista. Escolhida uma das classes, a outra é eliminada da lista e o processamento continua da mesma forma, até que um nó folha seja atingido. Portanto, a classificação de um exemplo de um problema de k classes requer a avaliação de $k-1$ nós, ou seja, $k-1$ classificadores. Com isso o tempo de geração de resultados é reduzido.

Platt et al. (2000) demonstraram que essa técnica tem erro de generalização limitado pela margem máxima obtida em cada nó. Como os LMCs utilizam o princípio de maximização de margens, seu uso como indutor base torna-se bastante adequado neste caso.

[1] Um grafo direcionado acíclico é um grafo sem ciclos, cujas arestas possuem uma orientação.

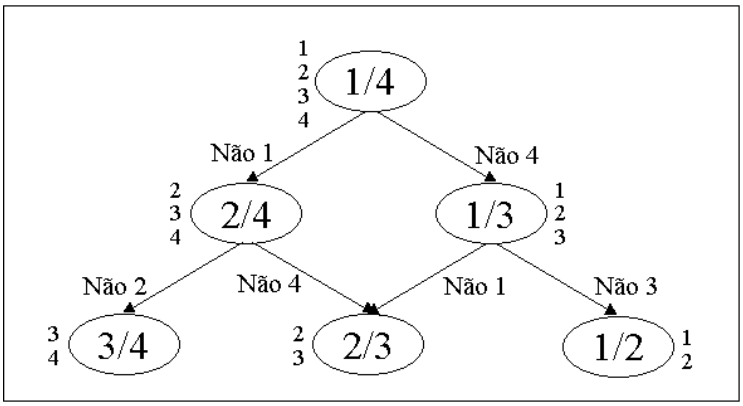

Figura 4.1: Exemplo de grafo direcionado utilizado para classificar quatro classes a partir de preditores binários (Platt et al., 2000).

4.3 Método de Códigos de Correção de Erros

Esta estratégia para obtenção de classificadores multiclasses a partir de preditores binários, proposta por Dietterich and Bariki (1995), utiliza uma saída em forma de códigos para representar as k classes de um problema.

Inicialmente, um código de tamanho l é atribuído a cada classe. O valor de l normalmente é maior que o número necessário para diferenciar cada classe unicamente. Os bits adicionais podem ser utilizados na correção de eventuais erros de classificação. Por este motivo, esta técnica possui a denominação de "método de códigos de correção de erros" (*Error-Correcting Output Codes*).

Uma matriz M com dimensão kxl ($M \in \{-1, +1\}^{k\text{x}l}$) armazena os códigos de cada classe em suas linhas. Cada coluna desta matriz correspone a um classificador, que deve ser gerado de forma a aprender as respostas desejadas apresentadas em M. Logo, l classificadores binários ($f_1(\mathbf{x}, \ldots, f_l(\mathbf{x}))$ são induzidos.

A submissão de um novo padrão neste sistema provê uma string s de tamanho l. Esta string é então comparada a cada linha de M de acordo com alguma medida, como a distância de *Hamming*, que conta o número de bits diferentes entre a string s e os códigos de M. Atribui-se ao novo padrão a classe correspondente à linha de M com menor distância em relação a s.

Allwein et al. (2000) propõem uma generalização deste método de forma a torná-lo mais adequado aos LMCs. Isto é realizado com o uso de uma medida de distância baseada nas margens obtidas por cada preditor binário na classificação do novo padrão. A utilização desta nova medida sobre LMCs se mostra simples e provê melhores resultados que o método original de Dietterich and Bariki (1995).

4.4 Considerações Finais

Embora a maioria dos LMCs apresentados sejam originalmente produzidos para geração de classificadores binários, alguns artifícios permitem que estes sejam aplicados a problemas multiclasses, em que o número de classes é maior que dois.

Duas abordagens usuais para tal são a decomposição "um-contra-todos" e "todos-contra-todos". Sendo k o número de classes, no primeiro caso produz-se k classificadores, cada um separando uma classe i das $k-1$ restantes. A classe de um novo padrão é dada pelo índice do classificador que produz a maior saída. No caso da decomposição "todos-contra-todos", são produzidos classificadores para separação de cada classe i de outra j, em que $i, j = 1, \ldots, k$ e $i \neq j$. Neste caso, a saída de um padrão é usualmente dada por um voto de maioria entre os classificadores. Os classificadores gerados por decomposição todos-contra-todos também podem ser unidos por meio de um grafo direcionado acíclico (Platt et al., 2000).

Outra alternativa, proposta por Dietterich and Bariki (1995), constitui em utilizar códigos para representar cada classe do problema. Nesta abordagem, denominada "método de códigos de correção de erros", classificadores binários são gerados de forma a produzir cada um dos "bits" nos códigos. Para cada novo padrão submetido a este sistema, uma string é gerada. Esta é então comparada aos códigos de cada classe, e o rótulo do novo padrão é o da classe para a qual tem-se menor distância de Hamming. Através de modificações na computação da medida de distância, Allwein et al. (2000) formularam uma versão deste método mais adequada a algoritmos pertencentes à classe dos LMCs.

Cada uma dessas alternativas tem vantagens e desvantagens. Uma discussão e comparação mais detalhada dos dois primeiros métodos decomposicionais descritos neste Capítulo com o uso de SVMs é apresentada em (Hsu and Lin, 2002). Os artigos originais de Dietterich and Bariki (1995); Allwein et al. (2000) apresentam também análises para o caso do "método de códigos de correção de erros".

Capítulo 5

Conclusão

Este relatório apresentou uma introdução aos Classificadores de Margens Largas (*Large Margin Classifiers* - LMCs). Foram apresentados os principais conceitos a respeito deste tipo de classificadores, bem como algumas técnicas pertencentes a esta classe de preditores.

Essas técnicas realizam, implícita ou explicitamente, uma maximização da margem de separação entre dados de diferentes classes. Através deste procedimento, elas alcançam em geral bons desempenhos em termos de generalização. Esta característica é também comprovada teoricamente, através de limites no poder de generalização de um classificador baseado na margem de separação que este induz sobre os dados, providos pela Teoria de Aprendizado Estatístico (TAE) (Vapnik, 1995; Shawe-Taylor et al., 1998).

Entre as técnicas descritas, destacam-se as Máquinas de Vetores Suporte (*Support Vector Machines* - SVMs) e o *AdaBoost* (AB).

As SVMs surgiram pelo emprego direto dos conceitos da TAE. Diversos mecanismos empregados no desenvolvimento desta técnica levaram à formulação de outros algoritmos de margens largas, tais como o Perceptron de Margens Largas, as Árvores de Decisão de Margens Largas e o Discriminante de Fisher com Kernels.

O AB, por sua vez, parte de um princípio bastante distinto na obtenção de classificadores com margens largas, uma vez que este é um algoritmo utilizado na combinação de preditores gerados por outras técnicas de aprendizado. A maximização de margens neste caso é implícita, sendo obtida pelo fato do algoritmo enfatizar padrões "difíceis" no treinamento dos diversos classificadores a serem combinados.

Deve-se ressaltar que apenas uma visão introdutória foi apresentada neste relatório. Para maiores detalhes sobre cada uma das técnicas e conceitos apresentados, as referências originais devem ser consultadas.

Referências Bibliográficas

Allwein, E. L., Shapire, R. E., and Singer, Y. (2000). Reducing multiclass to binary: a unifying approach for magin classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 9–16. Morgan Kaufmann.

Baranauskas, J. A. and Monard, M. C. (2000). Reviewing some machine learning concepts and methods. Technical Report 102, Istituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, `ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_102.ps.zip`.

Barlett, P. and Shawe-Taylor, J. (1999). Generalization performance of support vector machines and other classifiers. In *Advances in Kernel Methods - Support Vector Learning*, pages 43–51. MIT Press.

Barlett, P. (1998). The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536.

Bennet, K. P., Cristianini, N., Taylor, J.-S., and Wu, D. (2000). Enlarging the margins in perceptron decision trees. *Machine Learning*, 41(3):295–313.

Boser, B. E., Guyon, I. L., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Computational Learning Theory*, pages 144–152.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (1998). Arcing classifiers. *Annals of Statistics*, 3(26):801–849.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):1–43.

Campbell, C. (2000). An introduction to kernel methods. In Howlett, R. J. and Jain, L. C., editors, *Radial Basis Function Networks: Design and Applications*, pages 155–192, Berlin. Springer-Verlag.

Cortes, C. and Vapnik, V. N. (1995). Support vector networks. *Machine Learning*, 20:273–296.

Cristianini, N., Shawe-Taylor, J., and Sykacek, P. (1997). Bayesian classifiers are large margin hyperplanes in a hilbert space. In Fisher, D. H., editor, *Proceedings of the International Conference on Machine Learning (ICML)*, pages 109–117, Nashville, Tenesse. Morgan Kaufmann Publishers.

Cristianini, N. and Shawe-Taylor, J. (1999). Large margin classifiers and bayesian voting schemes. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, pages 55–68. MIT Press.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

Cristianini, N. (2000). Large margin strategies in machine learning. In *ISCAS 2000, IEEE International Symposium on Circuits and Systems*, pages II 753–II 756, Geneva, Switzerland. IEEE Computer Society Press.

Dietterich, T. G. and Bariki, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286.

Drucker, H. and Cortes, C. (1996). Boosting decision trees. *Advances in Neural Information Processing Systems*, 8:479–485.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, 7:179–188.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1(55):119–139.

Freund, Y. and Schapire, R. E. (1998). Large margin classification using the perceptron algorithm. In *Computational Learning Theory*, pages 209–217.

Freund, Y. (1995). Boosting, a weak learning algorithm by majority. *Information and Computation*, 2(121):256–285.

Friedman, J. H. (1996). Another approach to polychotomous classification. Technical Report 18, Department of Statistics, Stanford University, `http://www-stat.stanford.edu/reports/friedman/poly.ps.Z`.

Haykin, S. (1999). *Neural Networks - A Compreensive Foundation*. Prentice-Hall, New Jersey, 2 edition.

Hearst, M. A., Schölkopf, B., Dumais, S., Osuna, E., and Platt, J. (1998). Trends and controversies - support vector machines. *IEEE Intelligent Systems*, 13(4):18–28.

Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425.

Kearns, M. and Valiant, L. G. (1988). Learning boolen formulae or finite automata is a hard factoring. Technical Report TR-14-88, Harvard University Aiken Computation Laboratory.

Kearns, M. and Valiant, L. G. (1994). Cryptographic limitations on learning boolean formulae and finite autmomata. *Journal of the Association for Computer Machinery*, 1(41):67–95.

Mayoraz, E. and Alpaydin, E. (1998). Support vector machines for multi-class classification. Research Report IDIAP-RR-98-06, Dalle Molle Institute for Perceptual Artificial Intelligence, Martigny, Switzerland.

Mika, S., Rätsch, G., and Müller, K.-R. (2001a). A mathematical programming approach for kernel fisher discriminants. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13, pages 591–597.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., and Müller, K.-R. (1999). Fisher discriminant analysis with kernels. In Hu, Y.-H., Larsen, J., Wilson, E., and Douglas, S., editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE Computer Society Press.

Mika, S., Smola, A. J., and Schölkopf, B. (2001b). An improved training algorithm for kernel fisher discriminants. In Jaakkola, T. and Richardson, T., editors, *Proceedings of AISTATS*, pages 98–104, San Mateo, CA. Morgan Kaufmann.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Müller, K. R., Mika, S., Rätsch, G., Tsuda, K., and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.

Murthy, S. K., Kasif, S., and Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32.

Platt, J. C., Cristiani, N., and Shawe-Taylor, J. (2000). Large margin DAGS for multiclass classification. In *Advances in Neural Information and Processing Systems*, pages 547–553. The MIT Press.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Quinlan, J. (1996). Bagging, boosting, and C4.5. In *Proceedings of the 13th American Association for Artificial Intelligence National Conference on Artificial Intelligence*, pages 725–730.

Rätsch, G., Onoda, T., and Müller, K.-R. (1998). Soft margins for Adaboost. Technical Report NC-TR-1998-021, Neural and Computational Learning II (NeuroCOLT2).

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, New York.

Russel and Norvig (1995). *Artificial Intelligence - a Modern Approach*. Prentice Hall.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Schapire, R. E. (1999). A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406.

Schapire, R., Freund, Y., Barlett, P., and Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. In Jr., D. H. F., editor, *Proceedings of the International Conference on Machine Learning (ICML97)*, pages 322–330. Morgan Kaufmann.

Schwenk, H. and Bengio, Y. (2000). Boosting neural networks. *Neural Computation*, 8(12):1869–1887.

Sharkey, A. (1996). On combining artificial neural nets. *Connection Science*, 8:299–313.

Shawe-Taylor, J., Barlett, P. L., Williamson, R. C., and Anthony, M. (1998). Structural risk minimization over data-dependent hierarquies. *IEEE Transactions on Information Theory*, 44(5):1926–1940.

Shawe-Taylor, J. and Cristianini, N. (1998). Margin distribution bounds on generalization. Technical Report NC-TR-98-020, NeuroCOLT.

Smola, A. J., Barlett, P., Schölkopf, B., and Schuurmans, D. (1999a). *Advances in Large Margin Classifiers*. MIT Press.

Smola, A. J., Barlett, P., Schölkopf, B., and Schuurmans, D. (1999b). *Introduction to Large Margin Classifiers*, chapter 1, pages 1–28. Volume 1 of Smola et al. (1999a).

Smola, A. J. and Schölkopf, B. (2002). *Learning with Kernels*. The MIT Press, Cambridge, MA.

Utgoff, P. E. (1989). Perceptron trees: a case study in hybrid concept representations. *Connection Science*, 1:377–391.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Vert, J.-P. (2001). Introduction to support vector machines and applications to computational biology (draft). `http://web.kuicr.kyoto-u.ac.jp/~vert/research/semsvm/`.

Weston, J. and Watkins, V. (1998). Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, University of London.