Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Revisão de Técnicas para Geração de Classificadores de Margens Largas Multiclasses

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

Nº 221

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Novembro/2003

Revisão de Técnicas para Geração de Classificadores de Margens Largas Multiclasses*

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. Evidências atuais demonstram que classificadores que implícita ou explicitamente maximizam a margem de separação entre padrões de diferentes classes estão entre as abordagens mais promissoras no aprendizado a partir de dados. As técnicas que se utilizam deste princípio na geração de preditores são denominadas "Classificadores de Margens Largas". Originalmente, estes algoritmos são capazes de lidar apenas com problemas binários. Porém, diversas aplicações envolvem a discriminação de mais de duas classes. Este relatório apresenta uma revisão das principais técnicas que podem ser utilizadas na generalização destes classificadores a problemas deste tipo, também denominados multiclasses.

Palavras-Chave: Aprendizado de Máquina supervisionado, Classificadores de Margens Largas, Máquinas de Vetores Suporte (*Support Vector Machines*), *Boosting*, aplicações multiclasses.

Novembro 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**

2 Aprendizado de Máquina **1**

3 Classificadores de Margens Largas **3**

3.1 Margem de um Classificador 3

3.2 A Teoria de Aprendizado Estatístico 6

3.2.1 Limites no Risco Funcional Baseados na Dimensão VC 7

3.2.2 Limites no Risco Funcional Baseados em Margens 10

3.3 Técnicas 12

3.3.1 Máquinas de Vetores Suporte 12

3.3.2 Boosting 16

4 Técnicas para Geração de LMCs Multiclasses **21**

4.1 Abordagem Decomposicional 21

4.1.1 Decomposição um-contra-todos 21

4.1.2 Decomposição todos-contra-todos 23

4.1.3 Decomposição por Códigos de Correção de Erros 28

4.1.4 Estratégia Conjunta - Framework de Integração 30

4.1.5 Outras Técnicas 32

4.2 Solucionando o Problema Multiclasses em Um Passo 34

4.2.1 Máquinas de Vetores Suporte 34

4.2.2 Boosting 36

5 Experimentos: Avaliação de Técnicas Decomposicionais para SVMs **38**

5.1 Materiais e Métodos 38

5.2 Resultados e Discussões 40

6 Conclusão **43**

Referências Bibliográficas **44**

Lista de Figuras

1 Ilustração de divisão no espaço de entradas por um hiperplano 4
2 Ilustração das margens de dois hiperplanos em um mesmo conjunto de dados . . 5
3 Exemplo de distribuição de três pontos no estado bidimensional. 8
4 Exemplo de distribuição de quatro pontos no espaço bidimensional. 8
5 Mapeamento do conjunto de dados para o espaço de características 15
6 Ilustração de um comitê de classificadores (Sharkey, 1996) 17
7 Ilustração de regiões em que os padrões não podem ser classificados segundo a decomposição 1ct original (Abe, 2003) 22
8 Fronteiras na abordagem 1ct com funções de decisão contínuas (Abe, 2003) . . . 23
9 Ilustração de região em que os dados não podem ser classificados na estratégia tct com votação por maioria (Abe and Inoue, 2002) 24
10 Fronteira de decisão extendida na combinação tct com a técnica FSVM (Abe and Inoue, 2002) 25
11 DAG para um problema com quatro classes (Platt et al., 2000) 26
12 Ilustração de fronteira de decisão obtida pela combinação tct por DAGs (Abe, 2003) 26
13 Ilustração de região de pontos para os quais a classificação do DAG difere de acordo com sua seqüência de nós (Kijsirikul and Ussivakul, 2002) 27
14 ADAG para problema com oito classes (Kijsirikul and Ussivakul, 2002) 27
15 Exemplos de classificadores hierárquicos 32

Lista de Tabelas

1 Descrições dos conjuntos de dados multiclasses 39
2 Taxas de acurácia de diferentes estratégias multiclasses 40
3 Taxas de padrões desconhecidos de diferentes estratégias multiclasses 41
4 Tempo médio de treinamento de diferentes estratégias multiclasses (segundos) . 42
5 Número médio de SVs de diferentes estratégias multiclasses 43
6 Número de neurônios intermediários das arquiteturas de RNs no tct-RN 43

Lista de Algoritmos

1 AdaBoost (Schapire, 1999) 18
2 AdaBoost.M1 (Freund and Schapire, 1996) 36
3 AdaBoost.MO (Allwein et al., 2000) 37

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).

Originalmente, os LMCs geram classificadores binários, capazes de diferenciar somente duas classes. Este fato não inviabiliza seu uso em aplicações que envolvem mais de duas classes. Existem diversas técnicas para extendê-los a problemas deste tipo, também denominados multiclasses.

Estas técnicas podem geralmente ser subdivididas em duas categorias distintas, de acordo com a estratégia empregada na generalização dos LMCs ao cenário multiclasses. O primeiro tipo compreende a combinação de diversos preditores binários na solução multiclasses. Seguindo outra vertente, diversos trabalhos recorrem a reformulações nos algoritmos originais dos LMCs, de forma a solucionar o problema multiclasses em um único passo. Este relatório apresenta uma revisão das principais técnicas pertencentes a essas duas categorias.

Iniciando estas descrições, alguns dos principais conceitos de Aprendizado de Máquina, campo de pesquisa do qual os LMCs fazem parte, são apresentados na Seção 2. A seguir, na Seção 3 os LMCs são descritos. As principais técnicas para obtenção de LMCs multiclasses são então apresentadas na Seção 4. Uma análise experimental de algumas técnicas decomposicionais para geração de SVMs multiclasses é realizada na Seção 5. Finalizando, na Seção 6 são apontadas algumas considerações finais deste estudo.

2 Aprendizado de Máquina

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

As diversas técnicas de AM podem ser divididas, de maneira geral, segundo o paradigma empregado pelas mesmas na análise dos dados. Pode-se dizer que o **paradigma de**

aprendizado determina a maneira como o algoritmo de AM se relaciona com seu meio ambiente, ou seja, o modo como se dará o seu aprendizado por meio dos dados (Haykin, 1999). Os dois principais paradigmas de aprendizado existentes são o supervisionado e o não-supervisionado.

No paradigma de aprendizado **supervisionado** tem-se a figura de um "professor externo", o qual representa o conhecimento do ambiente por conjuntos de exemplos na forma (entrada, saída). O algoritmo de AM é então treinado a partir de conjuntos de exemplos rotulados visando aprender uma função desejada. Este é o tipo de paradigma focado neste trabalho.

No paradigma de aprendizado **não-supervisionado** não há a presença de um professor, ou seja, não existem instâncias rotuladas da função a ser aprendida. O algoritmo de AM aprende a representar (ou agrupar) as entradas submetidas segundo uma medida de qualidade. Essas técnicas são utilizadas principalmente quando o objetivo for encontrar padrões ou tendências que auxiliem no entendimento dos dados (Souto et al., 2003).

No contexto de aprendizado supervisionado, dado um conjunto de exemplos rotulados, deve-se produzir um **classificador** (**modelo** ou **hipótese**) capaz de predizer precisamente a classe de novos padrões para os quais não se conhece a classificação. Este classificador representa uma aproximação da **função objetivo** que descreve os dados. Esta é a função que mapeia todos os dados do domínio em que está ocorrendo o aprendizado, incluindo os disponíveis para geração do classificador, em seus respectivos rótulos. Os **rótulos** ou **classes** representam o fenômeno de interesse sobre o qual se deseja fazer previsões.

Cada **exemplo** (também referenciado por **dado**, **padrão** ou **instância**) é tipicamente representado por um **vetor de características**. Cada **característica**, também denominada **atributo**, expressa um determinado aspecto do exemplo. Normalmente, há dois tipos básicos de atributos: **nominal** e **contínuo** (Monard and Baranauskas, 2003). Um atributo é definido como nominal (ou **categórico**) quando não existe uma ordem entre os valores que este pode assumir (por exemplo, entre cores). No caso de atributos contínuos, é possível definir uma ordem linear nos valores assumidos (por exemplo, entre pesos $\in \Re$).

Visando avaliar a taxa de predições corretas ou incorretas (também denominadas **taxa de acurácia** e **taxa de erros**, respectivamente) obtidas por um classificador sobre dados não apresentados em seu treinamento, o conjunto de exemplos é, em geral, dividido em dois subconjuntos disjuntos: de **treinamento** e de **teste**. O subconjunto de treinamento é utilizado no aprendizado do conceito e o subconjunto de teste é utilizado para medir o grau de efetividade do conceito aprendido sobre novos dados.

Outro método comumente utilizado para prover melhores estimativas da acurácia (ou da taxa de erro) é o ***r-fold cross-validation*** (Mitchell, 1997). Segundo o *r-fold cross-validation*, o conjunto de exemplos é dividido em r subconjuntos de tamanho aproxi-

madamente igual. Os padrões de $r-1$ partições são utilizados no treinamento de um classificador, o qual é então testado sobre a partição restante. Este processo é repetido r vezes, utilizando em cada ciclo uma partição diferente para teste. O erro final do classificador é dado pela média dos erros observados sobre cada conjunto de teste.

Estimado o erro ou acurácia de um classificador, a **generalização** deste é definida como sua capacidade de prever corretamente a classe de novos padrões, sendo avaliada por seu desempenho em conjuntos de teste. No caso em que o modelo se especializa nos dados utilizados em seu treinamento, apresentando baixa acurácia quando confrontado com novos dados, tem-se a ocorrência de ***overfitting*** (super ajustamento). É também possível induzir hipóteses que apresentam baixa acurácia mesmo no conjunto de treinamento. Tem-se então uma condição de ***underfitting***. Esta situação pode ocorrer, por exemplo, quando os exemplos de treinamento disponíveis são pouco representativos ou o modelo obtido é muito simples (Monard and Baranauskas, 2003).

3 Classificadores de Margens Largas

Os **Classificadores de Margens Largas** (***Large Margin Classifiers*** - LMCs) têm apresentado bons desempenhos em aplicações de áreas variadas, com acurácias similares ou superiores às obtidas por outras técnicas de AM reconhecidamente eficazes (Hearst et al., 1998; Smola et al., 1999). Estes resultados são alcançados por meio do princípio de maximização da margem de separação dos dados pertencentes a diferentes classes e encontram-se justificados nos fundamentos da Teoria de Aprendizado Estatístico (Smola et al., 1999; Vapnik, 1995).

Nesta seção são apresentados os principais conceitos relacionados aos LMCs. Inicialmente, na Seção 3.1 define-se o conceito de margem de um classificador e são apresentadas as primeiras motivações para o princípio de maximização desta medida. O embasamento teórico dos LMCs é provido pela Teoria de Aprendizado Estatístico, apresentada na Seção 3.2. Na Seção 3.3 as duas principais técnicas pertencentes a esta classe de preditores, as *Support Vector Machines* e o *Boosting*, são descritas.

3.1 Margem de um Classificador

Uma definição formal de um problema de classificação por técnicas de AM envolve um conjunto de dados de treinamento S, composto por n pares $(\mathbf{x}_i, y_i)$, em que $\mathbf{x}_i \in \Re^m$ e $y_i \in \{-1, +1\}$ (por simplicidade, considera-se um problema de classificação binária - com duas classes). 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

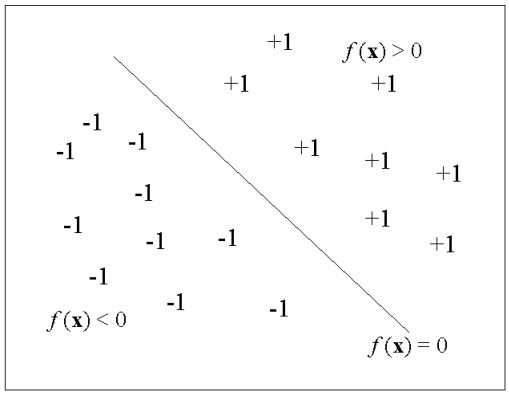

Figura 1: Ilustração de divisão no espaço de entradas por um hiperplano

$(\mathbf{x}, y)$ com acurácia.

Uma forma de realizar esta tarefa é utilizar uma função real $f : \Re^m \rightarrow \Re$ e aplicar uma função sinal $g(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$ aos valores de f. A função $g(\mathbf{x})$ assume valor $+1$ se $f(\mathbf{x}) > 0$ e -1 se $f(\mathbf{x}) < 0$ (se $f(\mathbf{x}) = 0$, $g(\mathbf{x})$ possui valor desconhecido).

Por simplicidade, seja f uma função linear, representada pela Equação 1.

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{1}$$

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 $\mathbf{w} \cdot \mathbf{x} + b = 0$ e b é um termo independente.

A partir desta definição de f, a função g pode ser representada pela Expressão 2.

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) = \begin{cases} +1 & \text{se } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ -1 & \text{se } \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases} \tag{2}$$

A Figura 1 ilustra um conjunto de dados e a divisão deste pela aplicação da função sinal $g(\mathbf{x})$ aos valores de uma função linear $f(\mathbf{x})$. Pode-se verificar que $f(\mathbf{x})$ define um hiperplano separando as classes $+1$ e -1.

Neste caso, o problema de aprendizado passa a ser encontrar os parâmetros $(\mathbf{w}, b)$, ou seja, uma função $f(\mathbf{x})$, para os quais $g(\mathbf{x})$ comete erro mínimo na classificação dos dados de treinamento. Na busca por classificadores lineares com erro mínimo em um conjunto de treinamento, porém, podem existir várias soluções. Na escolha do melhor classificador, 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 3 e mede a distância do padrão $\mathbf{x}_i$ em relação à fronteira $f(\mathbf{x}) = 0$ (ao hiperplano separador, no caso em que f é linear). Uma margem negativa indica uma

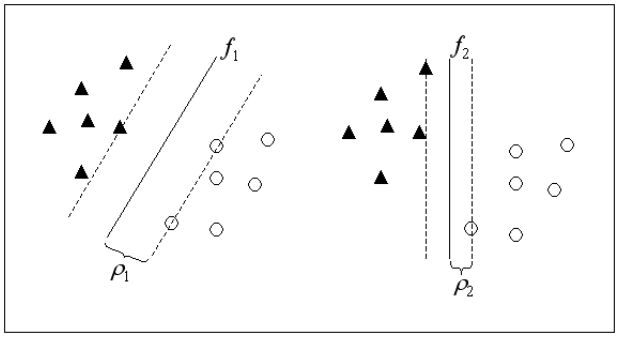

Figura 2: Ilustração das margens de dois hiperplanos em um mesmo conjunto de dados

classificação incorreta.

$$\rho(\mathbf{x}_i, y_i) = y_i f(\mathbf{x}_i) \tag{3}$$

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

$$\rho_f = \min_{1 \leq i \leq n} \rho\left(\mathbf{x}_i, y_i\right) = \min_{1 \leq i \leq n} y_i f\left(\mathbf{x}_i\right) \tag{4}$$

A Figura 2 ilustra dois possíveis classificadores lineares induzidos sobre um mesmo conjunto de dados e suas respectivas margens.

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

$$\rho_{f'} = \arg\max_{f} \rho_f \tag{5}$$

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

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 levam a uma classificação incorreta $g(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$. 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.

Além das motivações apresentadas anteriormente, existem limites no erro de um classificador 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 pela Teoria de Aprendizado Estatístico, apresentada a seguir. Por simplicidade e conveniência, diversas considerações apresentadas são feitas em relação a classificadores lineares. A extensão destes resultados a cada um dos tipos de LMCs são discutidas individualmente na ocasião de descrição dos mesmos.

3.2 A Teoria de Aprendizado Estatístico

Seja 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 $(\mathbf{x}_i, y_i)$, para gerar um classificador (ou função) particular $f' \in F$.

Na escolha desse classificador, é natural a consideração da função que produz o menor erro durante o treinamento, ou seja, daquela 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 6, a porcentagem de classificações incorretas obtidas em S, em que $c(\cdot,\cdot)$ é uma função de custo relacionando a previsão $f(\mathbf{x}_i)$ com a saída desejada y_i. Um tipo de função de custo é a "perda 0/1", definida por $c(f(\mathbf{x}), y) = 1$ se $yf(\mathbf{x}) < 0$ e 0 caso contrário (Müller et al., 2001).

$$R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} c(f(\mathbf{x}_i), y_i) \tag{6}$$

Supõe-se que os exemplos do domínio em que está ocorrendo o aprendizado são gerados independentemente e identicamente distribuídos (i.i.d.) de acordo com uma distribuição de probabilidade P. O desempenho em termos 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** (Equação 7), que quantifica a capacidade de generalização de f (se $R(f)$ for igual a 0, pode-se afirmar que f generaliza de forma "perfeita").

$$R(f) = \int c(f(\mathbf{x}), y) dP(\mathbf{x}, y) \tag{7}$$

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 $\hat{f}$ 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., 1999). Este classificador possui um risco empírico nulo, porém seu risco funcional é igual a 1/2.

A **Teoria de Aprendizado Estatístico** (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.

3.2.1 Limites no Risco Funcional Baseados na 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, relacionando 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 utilizando o 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., 1999).

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 as funções contidas em G têm 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.

Para o caso particular em que $f(\mathbf{x})$ é linear, a dimensão VC do conjunto de funções $G : g(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$ é igual a $m + 1$, em que m é a dimensão de $\mathbf{x}$ (Burges, 1998). Este resultado pode ser analisado por meio da observação das Figuras 3 e 4 (Smola and Schölkopf, 2002).

Na Figura 3, há três pontos no espaço bidimensional. Pode-se verificar que, para

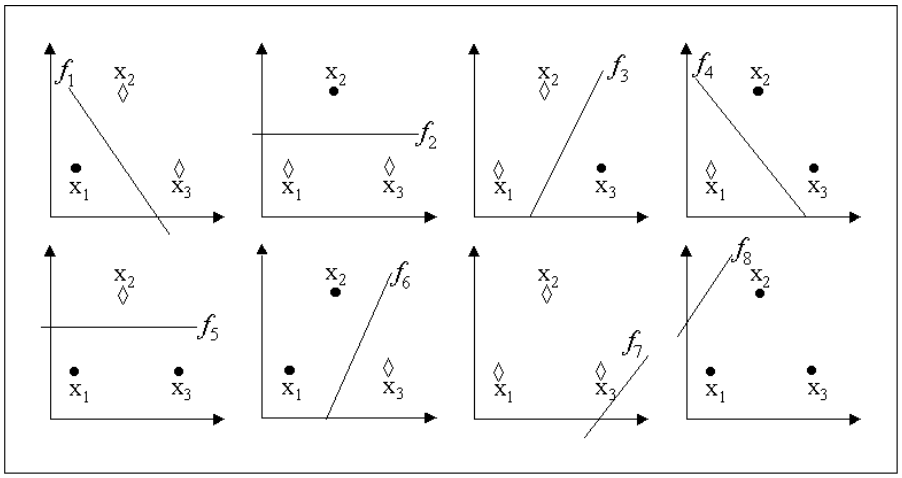

Figura 3: Exemplo de distribuição de três pontos no estado bidimensional (Smola and Schölkopf, 2002)

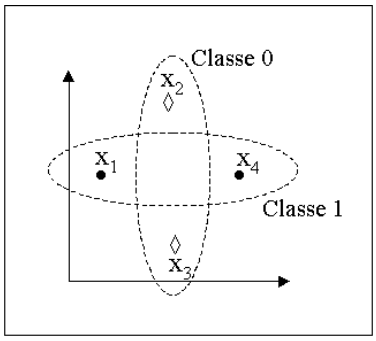

Figura 4: Exemplo de distribuição de quatro pontos no espaço bidimensional (Smola and Schölkopf, 2002)

qualquer conformação arbitrária de rótulos que estes dados possam assumir, é possível determinar retas capazes de fragmentá-los (separá-los de acordo com suas classes). Esta observação se aplica a quaisquer três pontos no espaço $\Re^2$. Porém, para quatro pontos em $\Re^2$, é possível definir rótulos para os dados tal que nenhuma reta seja capaz de fragmentá-los. Esta situação é ilustrada na Figura 4. Para uma fragmentação arbitrária de quatro pontos em $\Re^2$, deve-se então recorrer a uma função de maior complexidade. Portanto, a dimensão VC do conjunto de funções lineares no caso bidimensional é igual a 3, em conformidade com o resultado anteriormente apresentado.

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., 1999).

Teorema 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 $p \geq 1-\theta$ sobre n exemplos e para qualquer hipótese*

g em G, o risco funcional é limitado por

$$R(g) \leq R_{emp}(g) + \sqrt{\frac{c}{n}\left(h + \ln\left(\frac{1}{\theta}\right)\right)} \tag{8}$$

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

$$R(\hat{g}) \leq \inf_{g \in G} R(g) + \sqrt{\frac{c}{n}\left(h + \ln\left(\frac{1}{\theta}\right)\right)} \tag{9}$$

Teorema 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 $p \geq 1-\theta$ sobre n exemplos, o risco funcional da hipótese $g \in G$ satisfaz*

$$R(g) \geq \inf_{g' \in G} R(g') + \sqrt{\frac{c}{n}\left(h + \ln\left(\frac{1}{\theta}\right)\right)} \tag{10}$$

em que c é uma constante universal.

O termo mais à direita das Equações 8, 9 e 10 é também denominado termo de capacidade do conjunto de funções G.

Por meio da observação das equações apresentadas nos Teoremas 1 e 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 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.

Embora os Teoremas 1 e 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, Burges (1998) salienta que, sendo as Equações 8 e 9 limites superiores, não se impede que uma hipótese com mesmo risco empírico e maior dimensão VC tenha melhor desempenho. Em seu estudo, Burges (1998) apresenta inclusive resultados experimentais em que funções com maior dimensão VC (algumas vezes infinita) apresentam melhores resultados.

Podem haver problemas também 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., 1999; 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). Alguns deles são introduzidos a seguir.

3.2.2 Limites no Risco Funcional Baseados em Margens

Uma primeira análise mais refinada nos limites apresentados anteriormente foi proposta por Vapnik (Smola et al., 1999). Esta relaciona a dimensão VC de um classificador linear à margem obtida pelo mesmo na separação dos dados. O Teorema 3 formaliza este resultado.

Teorema 3 *Seja $X_0 \subset \Re^m$ o conjunto de entradas com norma menor ou igual a $R > 0$ ($\|\mathbf{x}_i\| \leq R$, $\forall \mathbf{x}_i \in X_0$) 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.*

$$F = \{\mathbf{x} \rightarrow \mathbf{w} \cdot \mathbf{x} | \, \|\mathbf{w}\| \leq 1, \, \mathbf{x} \in X_0\} \tag{11}$$

Considerando G o conjunto de funções sinal obtidas a partir de $G = \operatorname{sgn}(F)$ e h a dimensão VC de G, tem-se a Equação 12.

$$h \leq \min\left\{\frac{R^2}{\rho^2}, m\right\} + 1 \tag{12}$$

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 $m + 1$ apresentado anteriormente, em que não se levava em conta a margem ρ (Smola et al., 1999). 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 ele requer que todos exemplos, inclusive os de teste, estejam afastados do hiperplano de uma margem ρ.

O Teorema 4 provê limites no risco funcional para funções de decisão lineares em função unicamente da margem (Smola et al., 1999).

Teorema 4 *Seja o* ***erro (risco) marginal*** *de f ($R_\rho(f)$) a proporção de exemplos de treinamento que têm margem menor que ρ.*

$$R_\rho(f) = \frac{1}{n}\sum_{i=1}^{n} I(y_i f(\mathbf{x}_i) < \rho) \tag{13}$$

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

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

$$F = \{f \,|\, f(\mathbf{x}) = (\mathbf{w}\cdot\mathbf{x}) \text{ com } \|\mathbf{w}\| \leq 1 \text{ e } \mathbf{x} \in B_R\} \tag{14}$$

Existe uma constante c tal que, para qualquer distribuição de probabilidade, com probabilidade $p \geq 1-\theta$ sobre n exemplos de treinamento, para todo $\rho > 0$ e toda função $f \in F$, com margem $\geq \rho$ sobre todos exemplos de treinamento (ou seja, $R_\rho(f) = 0$), a Equação 15 é satisfeita.

$$R(f) \leq \frac{c}{n}\left(\frac{R^2}{\rho^2}\log^2\left(\frac{n}{\rho}\right) + \log\left(\frac{1}{\theta}\right)\right) \tag{15}$$

Com probabilidade $p \geq 1-\theta$, para todo $\rho > 0$, toda função $f \in F$ tem erro:

$$R(f) \leq R_\rho(f) + \sqrt{\frac{c}{n}\left(\frac{R^2}{\rho^2}\log^2\left(\frac{n}{\rho}\right) + \log\left(\frac{1}{\theta}\right)\right)} \tag{16}$$

Pode-se verificar por meio do Teorema 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 os dados de treinamento estarem distantes por 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. Em Smola and Schölkopf (2002) limites similares no risco funcional são apresentados para funções do tipo sinal com fronteiras de decisão lineares.

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 assim o erro sobre os dados de teste e de treinamento, respectivamente. O hiperplano que possui

esta margem é denominado ótimo.

Existem diversos outros limites no risco funcional reportados na literatura, assim como outros tipos de medida de complexidade de uma classe de funções (Müller et al., 2001). 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.

3.3 Técnicas

Diversas técnicas foram propostas utilizando os limites providos pela TAE na obtenção de classificadores com baixo risco funcional. Entre elas incluem-se generalizações de algoritmos de AM amplamente difundidos, como o Perceptron de Rosemblatt (1962), e modelos de AM formulados diretamente à partir dos limites da Teoria VC, como as Máquinas de Vetores Suporte (Cortes and Vapnik, 1995).

Esta Seção apresenta duas técnicas pertencentes à classe dos LMCs: as Máquinas de Vetores Suporte e o *Boosting*, sendo que esta última utiliza implicitamente os princípios de maximização de margem apresentados. Outras técnicas são revistas em (Lorena and Carvalho, 2003a).

Nas considerações realizadas, supõe-se que se deseja classificar objetos m-dimensionais $\mathbf{x}$ nas classes -1 e +1. Tem-se, portanto, uma caracterização de problema de classificação binária. Generalizações envolvendo o uso das técnicas identificadas em problemas com mais de duas classes são discutidas na Seção 4.

3.3.1 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). Na aplicação dessa teoria, deve-se escolher o classificador $g(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$ capaz de separar os dados pertencentes ao conjunto de treinamento com o menor risco empírico possível e que também satisfaça a restrição de pertencer a uma família G que minimize o risco funcional.

Considerando $f(\mathbf{x})$ uma função linear, a primeira condição consiste em buscar o par $(\mathbf{w}, b)$ tal que o hiperplano gerado separe os dados de treinamento com erro mínimo. Para satisfazer à restrição de minimização do risco funcional, utiliza-se o resultado da TAE relacionando a minimização deste risco à margem (ρ) obtida pelo classificador na separação dos dados. Portanto, entre os classificadores que minimizam o risco empírico, deve-se escolher aquele que possui a maior margem ρ'.

Maximizar ρ equivale a minimizar a norma de $||\mathbf{w}||$ (Hearst et al., 1998). Logo, pode-se manter ρ fixo e buscar um hiperplano com $||\mathbf{w}||$ pequeno tal que não haja exemplos de treinamento com margem menor que ρ, ou seja, para qualquer $\mathbf{x}_i$, $i = 1, \ldots, n$, deve-se ter $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq \rho$ (Smola and Schölkopf, 2002). Fixando ρ em 1, tem-se o seguinte problema de otimização:

$$\begin{gathered} \textbf{Minimizar: } ||\mathbf{w}||^2 \\ \textbf{Sob as restrições: } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \text{ para } i = 1, \ldots, n \end{gathered} \tag{17}$$

O hiperplano obtido na resolução deste problema é capaz de realizar a classificação de conjuntos linearmente separáveis. De forma a generalizá-lo para conjuntos mais gerais, pode-se utilizar o artifício de permitir que alguns padrões tenham margem menor que ρ. Isto é obtido com o relaxamento das restrições do problema de otimização apresentado, que é reformulado da seguinte maneira:

$$\begin{gathered} \textbf{Minimizar: } ||\mathbf{w}||^2 + B \sum_{i=1}^{n} \varepsilon_i \\ \textbf{Sob as restrições: } \begin{cases} \varepsilon_i \geq 0 \\ y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1 - \varepsilon_i \end{cases} \end{gathered} \tag{18}$$

em que B é uma constante que impõe um peso diferente para o treinamento em relação à generalização e ε representa a variável responsável pela imposição do relaxamento às restrições do problema de otimização.

Este é um problema clássico em otimização denominado **programação quadrática**[1]. O problema de AM é então reduzido a uma forma que pode ser analisada sob o ponto de vista de otimização, 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 $\mathbf{w}$, b e ε, apresentada em 19 (Campbell, 2000).

$$L\left(\mathbf{w}, b, \varepsilon, \eta, \lambda\right) = ||\mathbf{w}||^2 + B \sum_{i=1}^{n} \varepsilon_i - \sum_{i=1}^{n} \eta_i \left[\varepsilon_i - 1 + y_i \left(\mathbf{w} \cdot \mathbf{x} - b\right)\right] - \sum_{i=1}^{n} \lambda_i \varepsilon_i \tag{19}$$

em que os parâmetros η_i e λ_i são denominados **multiplicadores de Lagrange**. O problema passa a ser então a minimização de 19 em relação a $\mathbf{w}$ e b e a maximização dos valores de η_i e λ_i. Os pontos ótimos desta equação são obtidos por meio da resolução das

[1] O problema de otimização para conjuntos linearmente separáveis também é um caso de programação quadrática.

igualdades apresentadas em 20, 21 e 22.

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} y_i \eta_i = 0 \tag{20}$$

$$\frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{w} - \sum_{i=1}^{n} y_i \eta_i \mathbf{x}_i = 0 \tag{21}$$

$$\frac{\partial L}{\partial \varepsilon_i} = B - \eta_i - \lambda_i = 0, \text{ para } i = 1, \ldots, n \tag{22}$$

Substituindo as Equações obtidas em 19, obtém-se o seguinte problema de otimização (denominado dual):

$$\begin{array}{l} \textbf{Maximizar: } \sum_{i=1}^{n} \eta_i - \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} \eta_i \eta_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \textbf{Sujeito a: } \begin{cases} 0 \leq \eta_i \leq B, \, i = 1, \ \ldots, \, n \\ \sum_{i=1}^{n} \eta_i y_i = 0 \end{cases} \end{array} \tag{23}$$

Calculado o valor ótimo $\boldsymbol{\eta}^*$, $\mathbf{w}^*$ é encontrado pela resolução de 21[2]. Uma vez determinada a direção do hiperplano ótimo (ou seja, $\mathbf{w}^*$), b^* pode ser calculado por 24.

$$b^* = -\frac{1}{2} \left[\max_{\{i | y_i = -1\}} (\mathbf{w}^* \cdot \mathbf{x}_i) + \min_{\{i | y_i = +1\}} (\mathbf{w}^* \cdot \mathbf{x}_i) \right] \tag{24}$$

Seja $\boldsymbol{\eta}^*$ o valor ótimo encontrado pela otimização do problema dual. É demonstrado que η_i^* assume valores positivos para exemplos do treinamento que estão a uma distância do hiperplano ótimo menor ou igual à margem (ou seja, padrões sobre as margens ou entre elas). Para os outros padrões, o valor de η_i^* é nulo. Os exemplos para os quais $\eta_i^* > 0$ são denominados **vetores suporte** (*Support Vectors* - SVs).

Fixados os valores de $\mathbf{w}^*$ e b^*, o classificador extraído é definido pela Equação 25.

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn}(\sum_{\mathbf{x}_i \in \text{SV}} \eta_i^* y_i \mathbf{x}_i \cdot \mathbf{x} + b^*) = \begin{cases} +1 \text{ se } \sum_{\mathbf{x}_i \in \text{SV}} \eta_i^* y_i \mathbf{x}_i \cdot \mathbf{x} + b^* > 0 \\ -1 \text{ se } \sum_{\mathbf{x}_i \in \text{SV}} \eta_i^* y_i \mathbf{x}_i \cdot \mathbf{x} + b^* < 0 \end{cases} \tag{25}$$

A partir de 25, 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.

Porém, o classificador obtido ainda tem utilização limitada. Há muitos casos em que

[2]Os valores ótimos das variáveis serão indicados por *.

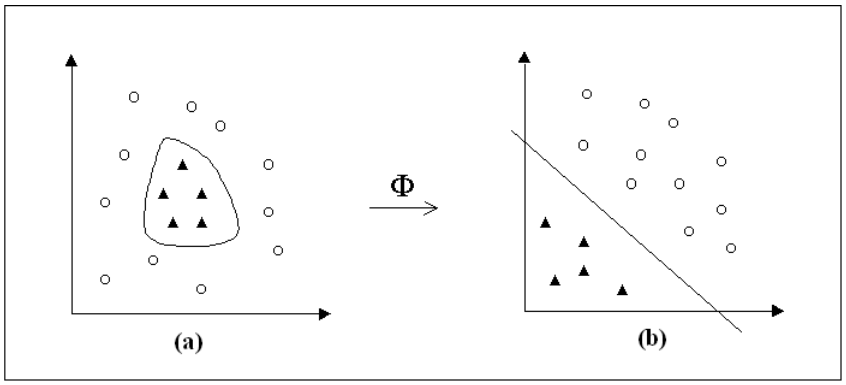

Figura 5: Mapeamento do conjunto de dados no espaço de entrada (a) para o espaço de características (b)

não é possível dividir satisfatoriamente os dados de treinamento por um hiperplano. Um exemplo é apresentado na Figura 5a, em que o uso de uma fronteira curva na separação das classes seria mais adequado.

De forma a generalizar as SVMs para essas situações, define-se funções reais $\Phi_1, \ldots, \Phi_U$ no domínio do espaço de entrada. Por meio da utilização destas funções, mapea-se cada padrão do conjunto de treinamento S para um novo espaço, denominado **espaço de características**, da maneira apresentada em 26.

$$\begin{aligned} \mathbf{x}_i \ \ (i = 1, \ldots, n) \ \ &\mapsto \ \ \mathbf{\Phi}\left(\mathbf{x}_i\right) = \left(\Phi_1\left(\mathbf{x}_i\right), \Phi_2\left(\mathbf{x}_i\right), \ldots, \Phi_U\left(\mathbf{x}_i\right)\right) \\ \Rightarrow \ \ \mathbf{\Phi}\left(S\right) &= \left\{\left(\mathbf{\Phi}\left(\mathbf{x}_1\right), y_1\right), \left(\mathbf{\Phi}\left(\mathbf{x}_2\right), y_2\right), \ldots, \left(\mathbf{\Phi}\left(\mathbf{x}_n\right), y_n\right)\right\} \end{aligned} \tag{26}$$

Estas funções Φ_i podem ser não-lineares. Em particular, U 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 separado por um hiperplano. Este fato é ilustrado na Figura 5.

SVMs lineares podem então ser utilizadas sobre o conjunto de treinamento mapeado no espaço de características (Hearst et al., 1998). Para isto, basta aplicar a função de mapeamento $\mathbf{\Phi}$ a cada padrão nas equações listadas para o caso linear. Por meio desse procedimento, 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 (Haykin, 1999).

De maneira geral, a função Kernel é mais simples que a de mapeamento $\mathbf{\Phi}$, que em grande parte dos casos 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 e os Gaussianos ou RBF (*Radial-Basis Function*), apresentados nas Equações 27 e 28, respectivamente (Haykin, 1999).

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T \cdot \mathbf{x}_j + 1\right)^d \tag{27}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \tag{28}$$

Além de sua boa capacidade de generalização, as SVMs também se destacam por sua robustez em relação a padrões de grandes dimensões. Esta característica as torna apropriadas a aplicações em Bioinformática, em que os dados frequentemente se caracterizam por possuir um grande número de atributos.

Suas principais deficiências encontram-se em sua sensibilidade a escolhas de parâmetros pelo usuário (embora em geral o número de parâmetros das SVMs seja menor que o das Redes Neurais Artificiais, por exemplo) e na dificuldade de interpretação do modelo gerado por esta técnica.

Um estudo mais detalhado a respeito das SVMs pode também ser consultado em (Lorena and Carvalho, 2003b).

3.3.2 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 Árvores de Decisão (ADs) (Drucker and Cortes, 1996) e Redes Neurais Artificiais (RNs) (Schwenk and Bengio, 2000). O *Boosting* gera, portanto, comitês de classificadores.

Um **comitê** (Figura 6) consiste de um conjunto de preditores cujas saídas são combinadas na previsão da classe de novas instâncias (Monard and Baranauskas, 2003). 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 continua-

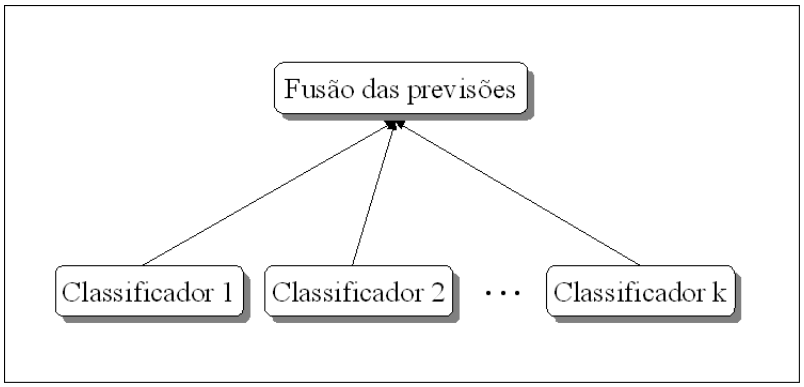

Figura 6: Ilustração de um comitê de classificadores (Sharkey, 1996)

mente 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 aleatória, 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). Em 1995, Freund (1995) propôs o **AdaBoost**, que resolveu grande parte dos problemas dos algoritmos de *Boosting* anteriores.

O Algoritmo 1 apresenta o AdaBoost (AB). Neste algoritmo, D_t representa a distribuição de probabilidade atribuída aos exemplos de treinamento. A variável t corresponde ao ciclo corrente do algoritmo, que é iterado T vezes. Um total de T hipóteses h_t são então geradas. Para implementação do uso da distribuição D_t, 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 γ_t, calculadas a partir do erro ϵ_t cometido pelas mesmas ($\gamma_t \geq 0$ se $\epsilon_t \leq 1/2$ e é maior à medida que ϵ_t diminui).

Freund and Schapire (1999) provaram que o erro de treinamento (risco empírico) do classificador gerado segundo o AB é limitado pela Equação 29, em que $I(q)$ é igual a 1 se q é verdadeiro e 0, caso contrário.

$$R_{emp}(H) = \frac{1}{n}\sum_{i=1}^{n} I\left(H\left(\mathbf{x}_i\right) \neq y_i\right) \leq \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i f\left(\mathbf{x}_i\right)\right) = \prod_{t=1}^{T} Z_t \tag{29}$$

Na Equação 29, $f\left(\mathbf{x}\right) = \sum_{t=1}^{T} \gamma_t h_t\left(\mathbf{x}\right)$, tal que $H(\mathbf{x}) = \mathrm{sgn}(f(\mathbf{x}))$ e Z_t é definido no Algo-

Algoritmo 1 AdaBoost (Schapire, 1999)

1: Dado um conjunto de treinamento $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, com $y_i \in \{-1, +1\}$
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

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

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

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

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

$$H(\mathbf{x}) \leftarrow \operatorname{sgn}\left(\sum_{t=1}^{T} \gamma_t h_t(\mathbf{x})\right)$$

ritmo 1. A desigualdade apresentada nesta equação segue do fato que $\exp(-y_i f(\mathbf{x}_i)) \geq 1$, se $y_i \neq H(\mathbf{x}_i)$.

Tem-se também que (Freund and Schapire, 1997):

$$\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_{t=1}^{T} \sqrt{1 - 4\delta_t^2} \leq \exp\left(-2\sum_{t=1}^{T} \delta_t^2\right) \tag{30}$$

em que $\delta_t = {}^1\!/_2 - \epsilon_t$. Como em problemas binários uma hipótese que classifica as instâncias de forma aleatória tem uma taxa de erro de ${}^1\!/_2$, δ_t mede quanto as hipóteses h_t são melhores que uma previsão randômica. Segundo as Equações 29 e 30, se cada classificador base é um pouco melhor que uma previsão randômica tal que $\delta_t \geq \delta$, para algum $\delta > 0$, então o erro de treinamento cai exponencialmente em função do número de funções base T que são combinadas, uma vez que o limite é no máximo igual a $\exp(-2T\delta^2)$ (Freund, 1995). Logo, o algoritmo base deve obedecer à restrição de gerar classificadores com erro menor que 1/2.

Seguindo as análises sobre o AB, a Equação 31 fornece um limite na capacidade de generalização (risco funcional $R(H)$) desta técnica em função do erro de treinamento, do tamanho do conjunto de treinamento n, da dimensão VC h da hipótese final e do número

de ciclos T (Freund and Schapire, 1997).

$$R(H) \leq \hat{P}r\left[H\left(\mathbf{x}\right) \neq y\right] + \tilde{O}\left(\sqrt{\frac{Th}{n}}\right) \tag{31}$$

em que $\hat{P}r$ denota a probabilidade empírica sobre o conjunto de treinamento e $\tilde{O}$ é uma notação em que termos constantes e logarítmicos são omitidos.

Segundo o limite fornecido em 31, 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 sobre o conjunto de teste (de generalização) continua a cair após o erro no treinamento ser nulo, o que contradiz o que é expresso na Equação 31.

Visando acomodar estas observações empíricas, o trabalho de Schapire et al. (1997) provê uma análise alternativa da capacidade de generalização do AB, em termos de margens. Os limites apresentados neste trabalho também são aplicáveis a outras técnicas baseadas na combinação de votos gerados por vários preditores, como o *Bagging*.

No caso do AB, a margem de um padrão $\rho(\mathbf{x}, y)$ pode ser definida por 32.

$$\rho\left(\mathbf{x}, y\right) = \frac{y f\left(\mathbf{x}\right)}{\sum\limits_{t=1}^{T} |\gamma_t|} = \frac{y \sum\limits_{t=1}^{T} \gamma_t h_t\left(\mathbf{x}\right)}{\sum\limits_{t=1}^{T} |\gamma_t|} \tag{32}$$

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

$$R(H) \leq \hat{P}r\left[\rho\left(\mathbf{x}, y\right) \leq \theta\right] + \tilde{O}\left(\sqrt{\frac{h}{n\theta^2}}\right) \tag{33}$$

para um $\theta > 0$ qualquer. O termo $\hat{P}r[\rho\left(\mathbf{x}, y\right) \leq \theta]$ equivale ao erro marginal sobre o conjunto de treinamento. Satisfazendo os resultados empíricos anteriormente descritos, a Equação 33 não depende do número de ciclos T. Como no Teorema 4.4, o risco funcional (capacidade de generalização) de H é limitado pela soma do erro marginal a um termo de capacidade inversamente proporcional à margem dos exemplos de treinamento. Com isto, o AB segue os limites similares aos providos na Seção 3.2.2.

Os autores Schapire et al. (1997) demonstraram que o AB realiza uma maximização implícita das margens, já que enfatiza os exemplos mais próximos à fronteira de decisão, os

quais em geral são mais difíceis de classificar. Schapire et al. (1997) também observaram que esta maximização é agressiva e que o AB continua a aumentar as margens dos exemplos de treinamento mesmo após o erro neste conjunto ser nulo, o que é diretamente refletido em reduções no erro sobre novos exemplos (no conjunto de teste).

O conceito de maximização de margens aproxima o AB das SVMs (Freund and Schapire, 1999). Freund and Schapire (1996) mostraram que o AB pode ser visto como um método de **programação linear**. Uma diferença direta entre as SVMs e o AB, portanto, se dá em como o problema de otimização é solucionado. No caso das SVMs recorre-se a um problema de programação quadrática, e no AB tem-se uma analogia com a programação linear.

Apesar do AB, como apresentado no Algoritmo 1, ter se mostrado eficiente em diversas aplicações práticas, sua precisão é afetada pela presença de ruídos nos dados. Isto ocorre devido ao artifício de se enfatizar exemplos mais difíceis de serem classificados, uma vez que os ruídos também podem ser caracterizados como tal.

Para minimizar a influência de padrões ruidosos, extensões são propostas visando "suavizar" o enfoque nos dados de treinamento, de forma similar ao que é realizado em relação às SVMs. Entre os trabalhos que utilizam este princípio está o **AdaBoost Regularizado** de Rätsch et al. (1998), que introduz modificações na atualização da distribuição de probabilidade para cada padrão. Esta passa a ser feita segundo a Equação 34.

$$D_{t+1}(\mathbf{x}_i) = D_t(\mathbf{x}_i) \exp\left\{\gamma_t I\left(y_i \neq h_t(\mathbf{x}_i)\right) - a\varsigma_t(\mathbf{x}_i)\left|\gamma_t\right|\right\} \tag{34}$$

A variável ς_t define a influência de um padrão sobre as hipóteses combinadas, e é definida pela Equação 35.

$$\varsigma_t(\mathbf{x}_i) = \sum_{j=1}^{t} \gamma_j D_j(\mathbf{x}_i) \tag{35}$$

Segundo esta expressão, um padrão classificado incorretamente com freqüência (difícil de classificar) tem um valor de ς alto. Sua influência é então minimizada na atualização da distribuição de probabilidade sobre os dados D_t.

Como as SVMs, o AB possui as vantagens de possuir boa capacidade de generalização, inclusive sobre dados de grande dimensionalidade. Além disso, o AB é simples e de fácil implementação.

Como restrições, tem-se que o AB pode falhar em duas situações (Schapire et al., 1997): quando não há dados de treinamento suficientes em relação à complexidade das hipóteses base e quando os erros das hipóteses base sobre os dados de treinamento (os termos ϵ_t no Algoritmo 1) se tornam grandes muito rapidamente.

4 Técnicas para Geração de LMCs Multiclasses

Como visto na seção anterior, as SVMs e o AB foram originalmente formulados para a solução de problemas binários, ou seja, de aplicações que envolvem a discriminação de apenas duas classes. Muitas aplicações, porém, abrangem o reconhecimento de mais classes. Este fato não inviabiliza o uso dos LMCs descritos. Diversas técnicas foram sugeridas para estendê-los a problemas desse tipo, também denominados multiclasses (Vapnik, 1998).

Em uma aplicação multiclasses, o conjunto de dados é composto de pares $(\mathbf{x}_i, y_i)$ tal que $y_i \in \{1, 2, \ldots, C\}$. O classificador deve então ser capaz de discriminar C classes, sendo $C > 2$.

A generalização dos LMCs para este tipo de aplicação pode ser realizada por meio da combinação de preditores gerados em subproblemas binários ou por adaptações em seus algoritmos originais. Na Seção 4.1 é apresentada uma revisão das principais técnicas da literatura que seguem a abordagem decomposicional na solução do problemas multiclasses. Considerações sobre a solução direta do problema multiclasses por meio de reformulações nos algoritmos originais das SVMs e do AB são apresentadas na Seção 4.2.

4.1 Abordagem Decomposicional

Na geração de classificadores multiclasses a partir de preditores binários, a forma mais direta consiste em decompor o problema multiclasses em sub-problemas de classificação binária. As saídas dos preditores binários gerados na solução de cada um desses subproblemas são então combinadas na produção do classificador final.

Esta seção descreve algumas das principais estratégias para solução de problemas multiclasses segundo a abordagem decomposicional.

4.1.1 Decomposição um-contra-todos

A primeira generalização natural de preditores binários a problemas multiclasses pode ser realizada por meio da decomposição **um-contra-todos** (1ct) (Vapnik, 1995). Segundo esta estratégia, dado um problema com C classes, C classificadores binários são gerados. Cada um desses preditores é treinado de forma a distinguir uma classe i das demais.

No treinamento de cada classificador, renomea-se todo o conjunto de treinamento de forma que elementos da classe i apresentem o rótulo +1 e os padrões das demais classes, o rótulo -1.

Dado que nos LMCs discutidos a classificação é dada por uma função sinal $g(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$, por meio deste procedimento tem-se C funções $g_i(\mathbf{x})$, conforme a Equação 36.

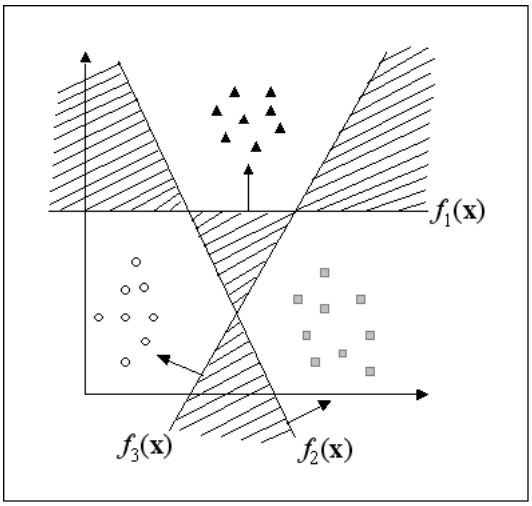

Figura 7: Ilustração de regiões em que os padrões não podem ser classificados segundo a decomposição 1ct original (Abe, 2003)

$$g_i(\mathbf{x}) = \text{sgn}(f_i(\mathbf{x})),\ i = 1, 2, \ldots, C \tag{36}$$

A partir dessas definições, o rótulo $f(\mathbf{x})$ de um novo padrão é dado pelo índice de classe i cuja função $f_i(\mathbf{x})$ correspondente satisfaz a Equação 37.

$$f(\mathbf{x}) = i \,|f_i(\mathbf{x}) > 0 \tag{37}$$

No caso de dois ou mais classificadores possuírem saídas maiores que 0, ou de nenhum classificador produzir esta saída, o padrão $\mathbf{x}$ não pode ser classificado, assumindo rótulo de desconhecido. A Figura 7 ilustra um exemplo com regiões (hachuradas) em que este fenômeno ocorre, para o caso particular em que as funções de decisão utilizadas na separação das classes são lineares.

Na resolução desta questão, Vapnik (1998) sugeriu o uso de **funções de decisão contínuas** na classificação do padrão. Por meio desta formulação, um exemplo $\mathbf{x}$ é classificado segundo a função $f_i(\mathbf{x})$ que produz maior saída (Equação 38).

$$f(\mathbf{x}) = i \,\Big|\, i = \underset{1 \leq i \leq C}{\arg\max}\,(f_i(\mathbf{x})) \tag{38}$$

Para o exemplo da Figura 7, as novas fronteiras de decisão são definidas, em pontilhado, na Figura 8.

Em (Inoue and Abe, 2001) outra forma de combinação de SVMs produzidas pela estratégia 1ct é apresentada. Para tal, os autores definem funções de qualidade capazes de abstrair o grau com que um padrão pertence a cada classe. As SVMs obtidas foram de-

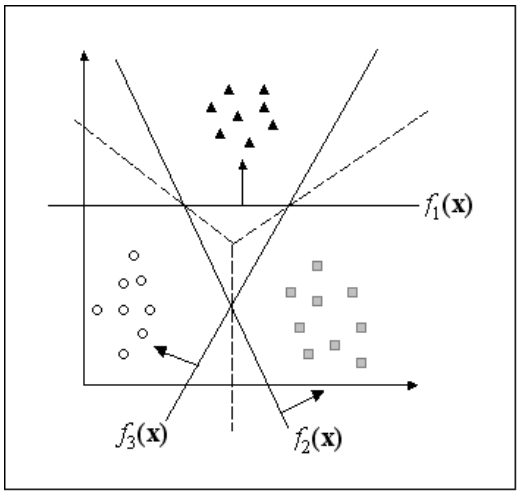

Figura 8: Fronteiras extendidas na abordagem 1ct com funções de decisão contínuas (Abe, 2003)

nominadas ***Fuzzy Suport Vector Machines*** (FSVMs). Porém, Abe (2003) demonstra que esta estratégia equivale à utilizada na 1ct com funções de decisão contínuas, levando à escolha da classe que satisfaz a Equação 38 e à partição do espaço de dados ilustrada na Figura 8.

4.1.2 Decomposição todos-contra-todos

Na decomposição **todos-contra-todos** (tct), dadas C classes, $C(C-1)/2$ classificadores binários são gerados. Cada um destes é responsável por diferenciar um par de classes (i, j), em que $i \neq j$. Logo, tem-se funções do tipo:

$$g_{ij}(\mathbf{x}) = \text{sgn}(f_{ij}(\mathbf{x})),\ i, j = 1, 2, \ldots, C,\ i \neq j \tag{39}$$

No treinamento de cada $g_{ij}(\mathbf{x})$ utiliza-se os padrões das classes i e j unicamente, sendo que os exemplos da classe i assumem rótulo +1 e os da classe j, rótulo -1.

Na união dos preditores gerados, Kreβel (1999) utilizou um esquema de **votação por maioria**. Dado um novo padrão $\mathbf{x}$, cada classificador fornece um voto em sua classe "preferida". O resultado é então dado pela classe que recebe mais votos. Este procedimento é equivalente a computar a Equação 40.

$$f(\mathbf{x}) = i \left| i = \underset{1 \leq i \leq C}{\arg\max} \left(\sum_{j=1;\, i \neq j}^{C} g_{ij}(\mathbf{x}) \right) \right. \tag{40}$$

Esta solução ainda apresenta, porém, o problema de gerar dados com rótulo desconhecido, representando o caso em que mais de uma classe recebe o mesmo número máximo

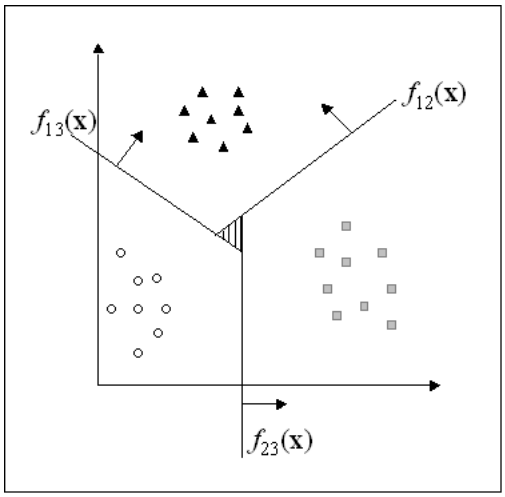

Figura 9: Ilustração de região em que os dados não podem ser classificados na estratégia tct com votação por maioria (Abe and Inoue, 2002)

de votos. A Figura 9 apresenta uma ilustração de região (hachurada) em que os dados não possuem classificação segundo a votação por maioria sobre classificadores lineares.

Visando solucionar este problema para o caso das SVMs, Abe and Inoue (2002) apresentaram uma extensão das FSVMs para a decomposição tct. Do mesmo modo, são introduzidas funções que expressam o grau com que um padrão pertence a uma determinada classe. Para tal, define-se inicialmente funções $m_{ij}(\mathbf{x})$ na direção ortogonal a $f_{ij}(\mathbf{x})$, para cada $i, j = 1, 2, \ldots, C$ e $i \neq j$, da forma apresentada na Equação 41.

$$m_{ij}(\mathbf{x}) = \begin{cases} 1 & \text{para } f_{ij}(\mathbf{x}) \geq 1 \\ f_{ij}(\mathbf{x}) & \text{caso contrário} \end{cases} \tag{41}$$

Define-se então funções de qualidade para cada classe i, por meio da Equação 42.

$$m_i(\mathbf{x}) = \min_{j=1,\ldots,C} (m_{ij}(\mathbf{x})) \tag{42}$$

Um novo padrão é então classificado segundo a Equação 43.

$$f(\mathbf{x}) = i \;\Big|\; i = \arg\max_{1 \leq i \leq C} (m_i(\mathbf{x})) \tag{43}$$

Por meio desta formulação, as fronteiras de decisão da Figura 9 são extendidas segundo as linhas pontilhadas apresentadas na Figura 10.

Em análises experimentais, a FSVM em geral obteve maiores taxas de acurácia que a estratégia tct com votação por maioria, embora seus resultados tenham sido estatisticamente similares.

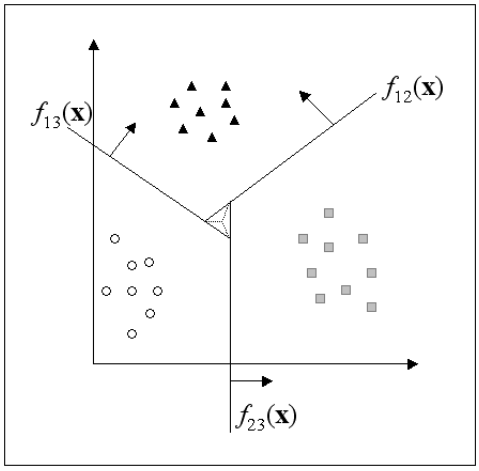

Figura 10: Fronteira de decisão extendida na combinação tct com a técnica FSVM (Abe and Inoue, 2002)

Outra forma de solucionar o problema de regiões com padrões desconhecidos se dá por meio do uso de **Grafos Direcionados Acíclicos** (*Directed Acyclic Graphs* - DAGs) (Platt et al., 2000). Os DAGs são grafos sem ciclos, cujas arestas possuem uma orientação. Na estrutura proposta por Platt et al. (2000), cada nó do grafo corresponde a um preditor binário. O DAG apresenta a vantagem de possuir uma teoria em termos de limites na capacidade de generalização do sistema de classificação obtido. Estes limites se dão em função do número de classes do problema e das margens alcançadas pelos preditores binários em cada nó. Este fato torna o uso de LMCs como preditores base adequado.

Computar a classe de um novo padrão utilizando o DAG equivale a operar uma lista de classes. Neste sentido, a cada nó do grafo é associada uma lista de rótulos. O nó raiz equivale à lista de todas classes. Começando por este nó, o exemplo é testado contra a primeira e a última classes da lista inicial. A classe com menor saída no nó é eliminada da lista e o nó equivalente à nova lista obtida é consultado. Este procedimento prossegue até que reste uma única classe na lista. Para C classes, $C-1$ classificadores são avaliados. Assim, esta estrutura acelera a fase de predição da abordagem tct. A Figura 11 ilustra um exemplo de DAG para quatro classes, utilizando SVMs como preditores base.

Um exemplo de como um DAG com classificadores lineares particiona o espaço de entradas apresentado nas Figuras 9 e 10 é ilustrado na Figura 12. Pode-se verificar que as fronteiras de decisão do DAG favorecem uma ou mais classes, o que representa uma deficiência em relação às FSVMs, que produzem regiões suaves (Figura 10), sem favorecimento de classes.

Outra desvantagem do DAG, apontada por Kijsirikul and Ussivakul (2002), é sua dependência em relação à seqüência de classificadores binários nos nós do grafo. Seja, por exemplo, um problema com quatro classes 1, 2, 3 e 4. A região sombreada na Figura

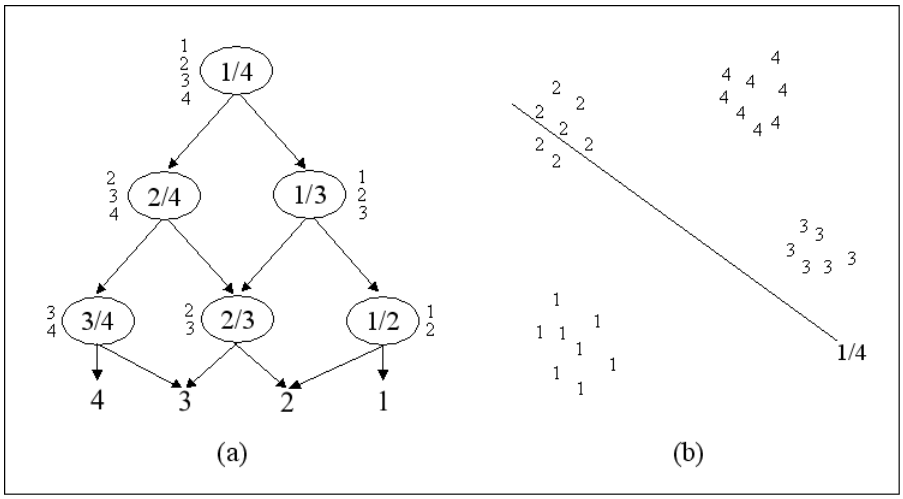

Figura 11: (a) DAG de um problema com quatro classes; (b) ilustração de SVM gerada para o subproblena 1 versus 4 (Platt et al., 2000)

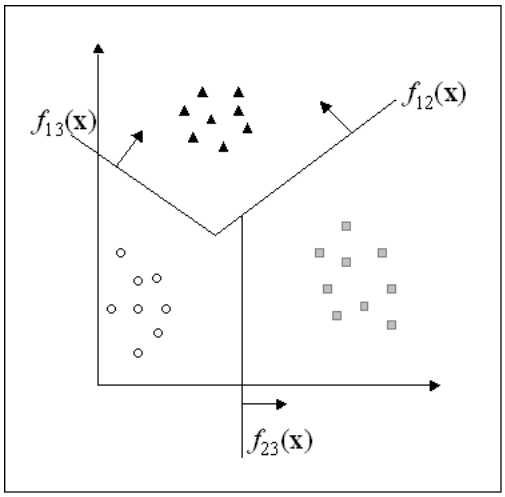

Figura 12: Ilustração de fronteira de decisão obtida pela combinação tct por DAGs (Abe, 2003)

13 ilustra pontos para os quais, dependendo da seqüência de nós visitados, classificações diferentes são produzidas. Seja o ponto $\mathbf{x}$ na região sombreada. Se o nó raiz equivale à lista $\{1, 2, 3, 4\}$, o hiperplano 1 versus 4 elimina a classe 4, pois o ponto $\mathbf{x}$ não se encontra no lado que representa esta classe. A nova lista $\{1, 2, 3\}$ (problema 1 versus 3) leva à eliminação do rótulo 3. A consulta do classificador 1 versus 2 (lista $\{1, 2\}$) remove a classe 1. Logo, a classe de $\mathbf{x}$ é 2. Por outro lado, se o nó raiz for equivalente à lista $\{2, 1, 3, 4\}$, o hiperplano 2 versus 4 elimina a classe 2. Tem-se então a lista $\{1, 3, 4\}$. O classificador 1 versus 4 remove o rótulo 4. A seguir, o hiperplano 1 versus 3 (lista $\{1, 3\}$) elimina a classe 3. Como resultado, tem-se que $\mathbf{x}$ é classificado como pertencente à classe 1, diferindo da primeira classificação produzida.

Outra desvantagem apontada pelos mesmos autores é a de que uma grande quantidade

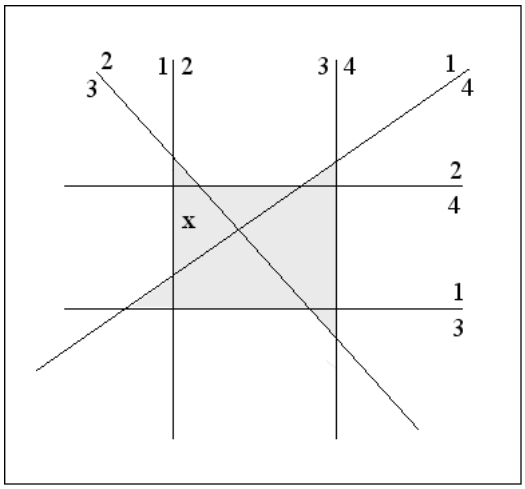

Figura 13: Ilustração de região de pontos para os quais a classificação do DAG difere de acordo com sua seqüência de nós (Kijsirikul and Ussivakul, 2002)

de nós desnecessários precisam ser avaliados na predição da classe correta. O número de vezes que o rótulo correto participa do processo de decisão é $C - 1$, que corresponde à profundidade do grafo. Com isto, os erros cometidos em cada nó consultado tornam-se cumulativos, o que se mostra uma falha principalmente para valores grandes de C.

Estes fatores motivaram Kijsirikul and Ussivakul (2002) a desenvolverem uma nova estratégia para combinação das saídas produzidas por classificadores tct. A nova estrutura, denominada ***Adaptive DAG*** (ADAG), corresponde a um DAG com estrutura reversa (Figura 14). O ADAG tem $C-1$ nós, cada um correspondendo a um classificador binário.

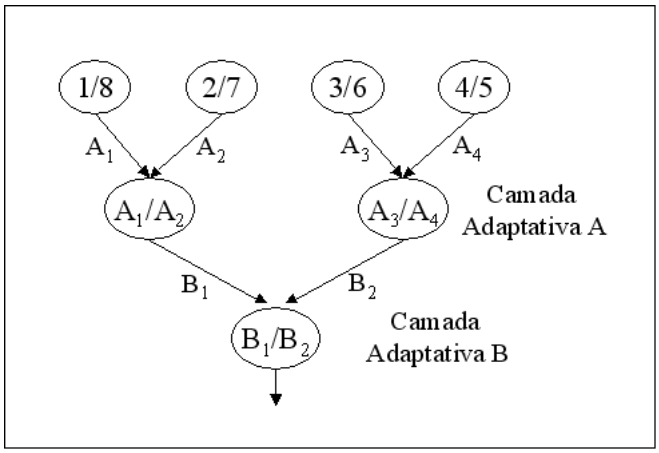

Figura 14: Estrutura de ADAG para problema com oito classes (Kijsirikul and Ussivakul, 2002)

Para classificação de um novo padrão utilizando o ADAG, inicialmente este é avaliado por todos classificadores do primeiro nível, os quais diferenciam pares distintos de classes. Cada um destes produz como saída sua classe "preferida". A cada passagem de nível, o número de classes candidatas é reduzido pela metade. Baseado nas escolhas do nível

anterior, os classificadores das próximas camadas são determinados. Este procedimento prossegue até que reste um único nó, o qual produz então a saída final.

Como no DAG, somente $C-1$ nós são avaliados pelo ADAG na fase de teste. Porém, a classe correta é testada contra outras $\log_2 C$ ou menos vezes, um número consideravelmente menor que as $C-1$ vezes do DAG.

Experimentalmente, o ADAG apresentou maior robustez que o DAG em problemas com números grandes de classes. Embora o ADAG também tenha mostrado menor dependência em relação à ordem dos classificadores binários consultados em cada previsão, ainda foram verificadas diferenças de acurácia entre seqüências distintas de nós visitados. Além disso, é apontado em (Abe, 2003) que ambas soluções baseadas em DAGs geram fronteiras de decisão que favorecem uma ou mais classes, conforme apresentado na Figura 12.

Em uma comparação entre os ADAGs, as FSVMs e a estratégia tct com votação, todas técnicas obtiveram acurácias similares, embora, na média, os ADAGs e as FSVMs tenham se sobressaído (Abe, 2003).

4.1.3 Decomposição por Códigos de Correção de Erros

Em uma estratégia decomposicional alternativa, Dietterich and Bariki (1995) propuseram o uso de códigos distribuídos para representar as C classes de um problema.

Inicialmente, códigos de tamanho l são atribuídos a cada classe. Os autores sugerem que o valor de l seja maior que o número mínimo necessário para diferenciar cada classe unicamente. Os bits adicionais têm a utilidade de prover ao sistema a capacidade de se recuperar de eventuais erros cometidos por alguns preditores no processo de classificação de um novo padrão. Por este motivo, esta técnica possui a denominação de **decomposição por códigos de correção de erros** (*Error-Correcting Output Codes* - ECOC).

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

Um novo padrão $\mathbf{x}$ pode ser classificado avaliando-se as predições dos l classificadores, que geram uma cadeia s de tamanho l. Esta cadeia é então comparada às linhas de $\mathbf{M}$. O padrão é atribuído à classe cuja linha de $\mathbf{M}$ é mais próxima de acordo com a distância de *Hamming*, que conta o número de ocorrências diferentes entre a cadeia e os códigos. A Equação 44 apresenta o cálculo da distância de *Hamming* para um padrão $\mathbf{x}$ em relação a uma classe q.

$$d_H(\mathbf{x}, q) = \sum_{i=1}^{l} 1 - \operatorname{sgn}\left(M(q,i)\, f_i(\mathbf{x})\right) \tag{44}$$

A partir desses cálculos, a classe do novo elemento é fornecida pela Equação 45.

$$f(\mathbf{x}) = i \left| i = \underset{1 \leq i \leq C}{\arg\min} \left(d_H(\mathbf{x}, i) \right) \right. \tag{45}$$

Para prover a capacidade de correção de erros, os códigos para as classes são escolhidos de forma que $\mathbf{M}$ safisfaça duas propriedades: separação de linhas e separação de colunas. Para $3 \leq C \leq 7$, por exemplo, o tamanho de código sugerido pelos autores foi de $2^{C-1} - 1$.

Na separação de linhas, o código de cada classe deve ser bem separado em termos de distância de *Hamming* dos das demais classes. Desta forma, garante-se uma representação única para cada rótulo e minimiza-se a "correlação" entre os códigos de diferentes classes, aumentando o poder de correção.

Na separação por colunas, deve-se determinar códigos tal que cada função (classificador) f_i tenha pouca ou nehuma correlação com outras funções f_j $(i \neq j)$ induzidas. Este requisito se justifica pelo fato de que os erros em posições distintas do código devem ser pouco correlacionados. Se vários erros simultâneos ocorrem, um código com bits correlacionados não será capaz de corrigí-los (Dietterich and Bariki, 1995). Para atingir o objetivo de separação de colunas, exige-se que a distância de *Hamming* entre uma coluna i e as demais seja grande. A distância entre cada coluna i e o complemento das outras também deve ser maximizada, uma vez que a inversão dos rótulos +1 e -1 em um conjunto de dados produz essencialmente o mesmo classificador.

A partir dessas considerações, Dietterich and Bariki (1995) propuseram cinco técnicas para construção de bons códigos de correção de erros. A escolha de cada técnica é determinada pelo número de classes do problema.

Experimentalmente, Dietterich and Bariki (1995) compararam o ECOC a ADs multiclasses[3], à decomposição 1ct com funções de decisão contínuas e a outro método para geração de códigos que se baseia nas características do problema, sem preocupações de correção. Como preditores foram utilizadas ADs e RNs. São apontados ganhos de acurácia do ECOC em alguns domínios. Porém, as características mais atrativas identificadas foram uma robustez do ECOC em relação ao tamanho do conjunto de treinamento e sua independência em relação a atribuições específicas de códigos às classes.

A seguir, um *framework* que unifica as abordagens ECOC e decomposicionais anteriormente apresentadas é descrito (Allwein et al., 2000). Além disso, modificações são introduzidas visando adaptá-las às características dos LMCs.

[3]As ADs são capazes de realizar predições multiclasses diretamente.

4.1.4 Estratégia Conjunta - Framework de Integração

Em (Allwein et al., 2000) é apresentado um *framework* para a unificação de estratégias de solução de problemas multiclasses a partir de preditores binários. Segundo o esquema proposto, as técnicas decomposicionais 1ct e tct são também reduzidas a métodos baseados em códigos.

Para tal, cada elemento da matriz $\mathbf{M}$ de códigos passa a assumir valores em $\{-1, 0, +1\}$. No caso da abordagem 1ct, tem-se uma matriz de dimensão CxC, em que os elementos da diagonal possuem valor +1 e os demais, valor -1. Na decomposição tct, $\mathbf{M}$ possui dimensão Cx$C(C-1)/2$ e cada coluna corresponde a um classificador binário para um par de classes (i, j), com $i \neq j$. Em cada coluna (i, j), o valor do elemento correspondente à linha i é +1 e o valor do elemento correspondente a j é igual a -1. Todos os outros elementos possuem valor 0, indicando que os padrões de outras classes não participam do processo de indução do classificador. Para o ECOC de Dietterich and Bariki (1995), tem-se o esquema de codificação apresentado na Seção 4.1.3.

Como no ECOC, classificadores binários são treinados de forma a aprender os rótulos representados nas colunas de $\mathbf{M}$. A predição da classe de um novo padrão $\mathbf{x}$ também é realizada de forma semelhante à apresentada no ECOC. O padrão é submetido a cada um dos classificadores e uma cadeia s é gerada. Esta é então comparada às linhas de $\mathbf{M}$. Para computação da distância entre as linhas de $\mathbf{M}$ e s, os autores utilizaram duas medidas.

A primeira foi a distância de *Hamming*, que neste *framework* é dada pela Equação 46, em que l representa o número de colunas de $\mathbf{M}$.

$$d_H(\mathbf{x}, q) = \sum_{i=1}^{l} \left(\frac{1 - \operatorname{sgn}(M(q, i) f_i(\mathbf{x}))}{2} \right) \tag{46}$$

Segundo esta expressão, se o valor de $M(q, i)$ for 0, então a classe q contribui com 1/2 na soma total. A predição da classe do novo padrão $\mathbf{x}$ continua sendo dada pela Equação 45, ou seja, pela distância de *Hamming* mínima observada entre s e as linhas de $\mathbf{M}$.

A outra medida de distância, proposta pelos autores Allwein et al. (2000), é denominada baseada em custo (*loss-based*). Esta medida é dada pela Equação 47 e considera a margem com que cada padrão é classificado por um LMC. A motivação em sua formulação reside no fato da distância de *Hamming* ignorar a função de custo utilizada no treinamento dos LMCs, assim como confidências ligadas às previsões feitas por estes tipos de classificadores, as quais podem ser quantificadas pelas margens associadas a cada padrão.

$$d_L(\mathbf{x}, q) = \sum_{i=1}^{l} L(\mathbf{M}(q, i) f_i(\mathbf{x})) \tag{47}$$

Calculados os valores de d_L para cada classe, o rótulo de $\mathbf{x}$ pode ser obtido pela Expressão 48.

$$f(\mathbf{x}) = i \left| i = \underset{1 \leq i \leq C}{\arg\min} \left(d_L(\mathbf{x}, i) \right) \right. \tag{48}$$

A função $L(M(q,i)f_i(\mathbf{x}))$ na Equação 47 é definida para cada LMC baseado na forma com que este trata a questão da maximização de margens.

Para as SVMs, tem-se:

$$L(z) = (1 - z)_+ \tag{49}$$

em que $(a)_+ = \max(a, 0)$. Sendo o problema de otimização das SVMs dado por 18, pode-se reduzir este à forma 50.

$$\text{Minimizar } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \left(1 - y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right)\right)_+ \tag{50}$$

o que remete à função de custo baseada em margens 49[4].

Para o AB, a função L é dada por:

$$L(z) = \exp^{-z} \tag{51}$$

pois é demonstrado que o AB maximiza as margens pela minimização implícita da função 52, em que $f(\mathbf{x}) = \sum_{t=1}^{T} \gamma_t h_t(\mathbf{x})$ (Schapire, 2002).

$$\frac{1}{n} \sum_{i=1}^{n} \exp^{-y_i f(\mathbf{x}_i)} \tag{52}$$

Em experimentos conduzidos comparando a utilização das medidas de distância d_H e d_L sobre as codificações 1ct, tct e ECOC, Allwein et al. (2000) verificaram em vários casos ganhos significativos de acurácia com o uso de d_L. Na comparação entre as técnicas multiclasses utilizando SVMs, verifica-se um pior desepenho da estratégia 1ct. Para o caso do AB, não foi possível identificar uma melhor ou pior abordagem.

Allwein et al. (2000) também sugerem que, apesar do código gerado pelo ECOC possuir boa propriedade de correção, vários dos subproblemas binários criados podem ser difíceis de aprender. Nas decomposições 1ct e tct tem-se problemas mais simples, o que justifica o desempenho comparável dessas três técnicas decomposicionais em vários dos casos testados.

[4]Lembrando que a margem de uma padrão $\mathbf{x}_i$ é dada por $\rho(\mathbf{x}_i, y_i) = y_i f(\mathbf{x}_i)$.

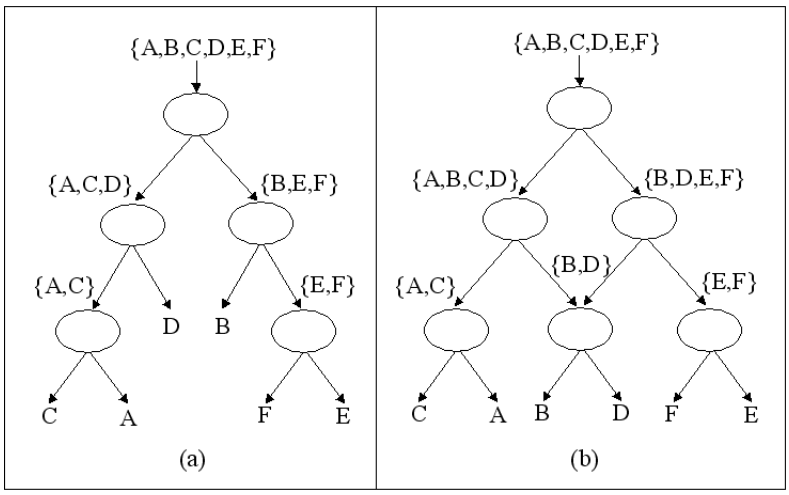

Figura 15: Exemplos de classificadores hierárquicos; (a) estrutura de árvore binária; (b) estrutura hierárquica geral (Schwenker, 2000)

4.1.5 Outras Técnicas

Esta Seção revisa outros desenvolvimentos na geração de classificadores multiclasses a partir de decomposições binárias. Os trabalhos descritos focam o uso de SVMs como preditores binários, embora possam ser aplicados a qualquer algoritmo de aprendizado.

Em (Schwenker, 2000) propõe-se a organização das SVMs binárias em uma estrutura hierárquica. A Figura 15 apresenta duas possíveis estruturas de classificadores hierárquicos para discriminação das letras {A, B, C, D, E, F}. A Figura 15a representa uma estrutura hierárquica em forma de árvore, em que cada nó possui exatamente uma aresta "apontando" para si. A Figura 15b representa uma estrutura mais geral, em uma forma especial de DAG.

Em 15a, o primeiro classificador recebe em seu treinamento todo o conjunto de dados e deve aprender a distinguir as letras {A,C,D} de {B,E,F}. O classificador à esquerda no próximo nível recebe os dados que fazem parte do conjunto {A,C,D} e deve então distinguir entre aqueles que pertencem ao conjunto {A,C} dos que possuem rótulo D. Logo, cada preditor binário recebe um determinado subconjunto de dados rotulados com +1 ou -1 a partir do que será discriminado no próximo nível.

Os conjuntos de classes discriminados em cada nó são definidos por meio do conceito de "classes de confusão". Uma **classe de confusão** representa um subconjunto das classes $\{1, 2, \ldots, C\}$ de um problema para os quais se há uma grande semelhança entre valores de atributos tal que mesmo uma pequena quantidade de ruídos pode levar a classificações errôneas (Schwenker, 2000).

Na obtenção dessas hierarquias de classes de confusão, os autores consideraram duas estratégias: geração de matrizes de confusão e uso de técnicas de agrupamento.

As **matrizes de confusão** oferecem medidas da efetividade de um classificador, por meio da ilustração do número de predições corretas e incorretas para cada classe (Baranauskas and Monard, 2000). Para um determinado conjunto de dados, suas linhas representam os valores de classes verdadeiros e as colunas, as classes previstas por um determinado classificador. Logo, cada elemento $A(i, j)$ de uma matriz de confusão $\mathbf{A}$ representa o número de exemplos da classe i classificados como pertencentes à classe j, para $i, j = 1, \ldots, C$. Tem-se então medidas quantitativas de quais classes o algoritmo de aprendizado tem maior dificuldade em realizar uma diferenciação.

A principal desvantagem no uso de uma matriz de confusão na obtenção das hierarquias está no fato de que algum outro algoritmo de aprendizado multiclasses deve ser empregado na solução de todo problema para sua geração. A avaliação das combinações de partições dos dados a serem utilizadas também pode se tornar computacionalmente cara.

No uso de técnicas de agrupamento na obtenção das hierarquias, os dados são primeiramente agrupados sem o conhecimento das classes a que pertencem. Em cada um dos aglomerados formados pode-se então utilizar o conhecimento das classes anteriormente ignorado para verificar a freqüência relativa com que membros de uma classe i aparecem em um aglomerado com maioria de elementos da classe j.

Em análises experimentais sobre uma base de dados contendo imagens de dígitos manuscritos, Schwenker (2000) comparou os resultados de RNs do tipo *Multilayer Perceptron* (MLP) e *Radial Basis Function* (RBF) (Haykin, 1999), do algoritmo k-NN (Mitchell, 1997) e de SVMs multiclasses geradas segundo as abordagens 1ct com funções de decisão contínuas, tct com votação por maioria e hierárquica com classes de confusão obtidas por agrupamento. De modo geral, a RN do tipo RBF e as SVMs se sobressaíram. Entre as estratégias para geração de SVMs multiclasses os desempenhos foram similares, não havendo diferença estatística significativa entre suas taxas de erro.

Em (Godbole et al., 2002) também é apresentada uma abordagem para geração de SVMs multiclasses baseada em classes que apresentam semelhança/confusão. Neste caso, uma matriz de confusão gerada por um classificador *Naïve-Bayes* (Good, 1965) foi utilizada na determinação das classes a participarem das SVMs binárias. Como na decomposição 1ct, foi gerada uma SVM para cada classe. Para a SVM correspondente à classe i, os exemplos com classe i eram rotulados com +1. Porém, os exemplos rotulados com -1 correspondiam somente àqueles mais difíceis de serem diferenciados da classe i segundo a matriz de confusão. Com este procedimento, reduz-se o número de elementos que participam do treinamento de cada SVM binária, acelerando o processo de obtenção do classificador multiclasses. Integra-se também as vantagens da rapidez dos classificadores *Naïve-Bayes* às altas taxas de acurácia das SVMs. Na fase de teste, um novo padrão $\mathbf{x}$ é primeiramente submetido ao classificador Bayesiano. Se este gera como previsão uma

classe i, recorre-se à SVM correspondente a esta classe, que fornece então uma previsão final mais refinada.

Experimentalmente, a nova estratégia formulada, denominada *GraphSVM*, foi comparada ao classificador *Naïve-Bayes* isoladamente e a uma SVM 1ct com funções de decisão contínuas. A *GraphSVM* superou o classificador Bayesiano significativamente en termos de acurácia. Em relação à SVM 1ct, a *GraphSVM* obteve acurácias semelhantes. Seu tempo de treinamento e custo de armazenagem é, porém, significativamente menor.

Em outro trabalho com SVMs multiclasses, Tax and Duin (2002) endereçaram o problema das abordagens 1ct original (sem fronteiras contínuas) e tct com votação por maioria gerarem classificações com rótulo indeterminado. Esses autores formularam uma estratégia para atribuição de valores de confidência às previsões de cada classificador binário gerado. A classificação final é então dada pela saída com maior confidência.

Na avaliação experimental do novo procedimento para combinação das SVMs binárias, Tax and Duin (2002) verificaram maiores ganhos no caso da decomposição 1ct, com a eliminação da ocorrência de padrões não classificados e a obtenção de boa acurácia.

4.2 Solucionando o Problema Multiclasses em Um Passo

Diversos trabalhos foram devotados a solucionar o problema multiclasses com LMCs em um único passo, sem a necessidade de criar múltiplos preditores binários.

Esta seção descreve alguns dos principais trabalhos que realizaram modificações nos algoritmos originais das SVMs e do AB de forma a solucionar o problema multiclasses diretamente.

4.2.1 Máquinas de Vetores Suporte

Na solução de problemas multiclasses em um passo por SVMs tem-se reformulações no algoritmo de treinamento destas, por meio da solução de um único problema de otimização (Bredensteiner and Bennet, 1999; Guermer, 2000; Weston and Watkins, 1998). Em geral, este procedimento leva à necessidade de solução de problemas de otimização consideravelmente maiories, os quais são computacionalmente mais caros (Hsu and Lin, 2002).

O trabalho de Weston and Watkins (1998) apresentou uma das primeiras reformulações das SVMs para solução multiclasses em um único passo de otimização. A idéia é similar à 1ct. São gerados C classificadores, cada um separando uma classe i das demais, porém todos são obtidos por meio de um único problema de otimização. Tem-se a seguinte

formulação:

$$\begin{array}{r} \text{Minimizar: } \frac{1}{2}\sum_{j=1}^{C} \mathbf{w}_j \cdot \mathbf{w}_j + B\sum_{i=1}^{n}\sum_{j\neq y_i} \varepsilon_i^j \\ \text{Sob as restrições: } \left\{ \begin{array}{c} \mathbf{w}_{y_i} \cdot \mathbf{\Phi}(\mathbf{x}_i) + b_{y_i} \geq \mathbf{w}_j \cdot \mathbf{\Phi}(\mathbf{x}_i) + b_j + 2 - \varepsilon_i^j \\ \varepsilon_i^j \geq 0, \quad i = 1, \ldots, n, \quad j \in \{1, \ldots, C\} \setminus y_i \end{array} \right. \end{array} \tag{53}$$

A função de decisão obtida pela solução deste problema é representada na Equação 54.

$$f(\mathbf{x}) = j \;\Big|\; j = \underset{j=1,\ldots,C}{\arg\max}\ (\mathbf{w}_j \cdot \mathbf{\Phi}(\mathbf{x}) + b_j) \tag{54}$$

Esta equação é equivalente à Expressão 38 do 1ct. Em (Weston and Watkins, 1998) esta formulação foi avaliada experimentalmente sobre conjuntos de dados pequenos, pois o número de variáveis no problema de otimização obtido é muito grande.

Em (Bredensteiner and Bennet, 1999) e (Guermer, 2000) também são apresentadas reformulações no processo de otimização das SVMs. Porém, é observado em (Guermer, 2000) que as formas primais dos problemas de otimização sugeridos são equivalentes à de Weston and Watkins (1998).

Em (Crammer and Singer, 2002) é descrita uma metodologia para geração de códigos de correção de erros que pode ser diretamente aplicada na obtenção de SVMs multiclasses. Na análise realizada por estes autores, a geração de um código discreto que represente as C classes de um problema e minimize o risco empírico obtido na solução deste é NP-completo. É proposta então a utilização de códigos assumindo valores contínuos.

Hsu and Lin (2002) realizaram uma análise experimental dos algoritmos de Crammer and Singer (2002) e de Weston and Watkins (1998) e os compararam às abordagens decomposicionais 1ct com funções de decisão contínuas, tct com votação por maioria e tct com combinação por DAGs. Antes, porém, foram realizadas generalizações no algoritmo de Weston and Watkins (1998) de forma a torná-lo capaz de lidar com aplicações com grandes quantidades de dados.

Em termos de acurácia, nenhuma técnica se mostrou estatisticamente melhor. Em relação ao tempo de treinamento, as estratégias tct foram mais rápidas. As técnicas multiclasses de um passo, por sua vez, foram lentas nesta fase. Com respeito ao tempo de teste (ou seja, o tempo de geração de previsões), as técnicas de um passo foram melhores. Entre as abordagens decomposicionais, o DAG foi o melhor nesta fase.

Além da complexidade maior dos problemas de otimização gerados e dos grandes tempos de treinamento, outra desvantagem identificada das técnicas de um passo para SVMs foi sua dificuldade de implementação em relação às abordagens decomposicionais. Como conclusão, Hsu and Lin (2002) sugerem que as estratégias mais apropriadas para

uso prático na solução de problemas multiclasses com SVMs são a tct com votação por maioria e o DAG.

4.2.2 Boosting

A primeira extensão do AB básico (Algoritmo 1) para a solução de problemas multiclasses foi o AdaBoost.M1 (AB.M1) (Freund and Schapire, 1996), apresentado no Algoritmo 2.

Algoritmo 2 AdaBoost.M1 (Freund and Schapire, 1996)

1: Dado um conjunto de treinamento $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ com $y_i \in \{1, 2, \ldots, C\}$
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

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

6: **se** $\epsilon_t > \frac{1}{2}$ **então**
7: $T \leftarrow t - 1$
8: Sair do *loop*
9: **fim se**
10: Escolher $\beta_t \leftarrow \frac{\epsilon_t}{1-\epsilon_t}$, $\gamma_t \leftarrow \ln\left(\frac{1}{\beta_t}\right)$ e $Z_t \leftarrow \sum_{i=1}^{n} D_t(\mathbf{x}_i) \exp\left(-\gamma_t y_i h_t(\mathbf{x}_i)\right)$
11: Atualizar D_t:

$$D_{t+1}(\mathbf{x}_i) \leftarrow \frac{D_t(\mathbf{x}_i)}{Z_t} \times \begin{cases} \beta_t & \text{se } h_t(\mathbf{x}_i) \neq y_i \\ 1 & \text{se } h_t(\mathbf{x}_i) = y_i \end{cases}$$

12: $t = t + 1$
13: **até** $t = T$
14: Hipótese Final:

$$H(\mathbf{x}) = \arg\max_{y \in \{1, \ldots, C\}} \sum_{t: h_t(\mathbf{x}) = y} \gamma_t$$

O AB.M1 parte do princípio que o algoritmo base deve ser capaz de realizar predições multiclasses. Outra modificação em relação ao AB básico se dá na obtenção da hipótese final, que produz como saída a classe y que maximiza a soma dos pesos das hipóteses h_t que predizem este rótulo. Além disso, se o erro ϵ_t de uma hipótese h_t for maior que $1/2$, o AB.M1 pára o processo de treinamento e utiliza somente as hipóteses já computadas. É argumentado que, para o caso binário, uma hipótese h com erro maior que $1/2$ pode ser substituída por $1 - h$, que possui então erro menor ou igual a este valor. Porém, para $C > 2$, uma hipótese com erro maior que $1/2$ se torna inútil para o algoritmo de *Boosting* (Freund and Schapire, 1996). Logo, o AB.M1 é adequado quando o classificador base é capaz de realizar predições multiclasses e consegue apresentar taxas de acurácia razoavelmente altas, mesmo nas distribuições rígidas criadas pelo algoritmo de *Boosting*.

Na solução desta questão, diversas outras técnicas foram desenvolvidas. Em geral, estas técnicas reduzem o problema multiclasses a um problema binário maior (Schapire, 2002).

Um exemplo é apresentado em (Allwein et al., 2000). Estes autores desenvolveram uma estratégia para indução de uma única hipótese capaz de implementar o problema multiclasses por meio de códigos. Para isto, sendo $\mathbf{M}$ uma matriz de códigos para cada classe como definida na Seção 4.1.4, o algoritmo de aprendizado binário passa a receber entradas na forma $((\mathbf{x}_i, j), M(y_i, j))$, para todos exemplos $\mathbf{x}_i$ e índices de coluna j em que $M(y_i, j) \neq 0$. Esta variante de um passo fundamentada em códigos foi denominada *single-call* e a original, baseada na geração de múltiplos indutores, *multi-call*.

A partir da variante *single-call*, Allwein et al. (2000) apresentaram um algoritmo que incorpora a estratégia de códigos segundo a distância *loss-based* (Equação 47) ao AB. O algoritmo equivale ao AdaBoost.MO (AB.MO) apresentado em (Schapire and Singer, 1999) e é ilustrado no Algoritmo 3.

Algoritmo 3 AdaBoost.MO (Allwein et al., 2000)

1: Dado um conjunto de treinamento $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ com $y_i \in \{1, 2, \ldots, C\}$
2: Inicializar $D_t(\mathbf{x}_i, j) \leftarrow \frac{1}{(nl)}$ 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

$$\begin{aligned}\epsilon_t &\leftarrow \Pr_{(\mathbf{x}_i,j)\sim D_t}\left[M\left(y_i, j\right) \neq h_t\left(\mathbf{x}_i, j\right)\right] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{l} D_t\left(\mathbf{x}_i, j\right) I(M\left(y_i, j\right) \neq h_t\left(\mathbf{x}_i, j\right))\end{aligned}$$

em que $I(q) = 1$ se q é verdadeiro e $I(q) = 0$ caso contrário
6: Escolher $\gamma_t \leftarrow \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ e $Z_t \leftarrow 2\sqrt{\epsilon_t\left(1-\epsilon_t\right)}$ (Z_t é um termo normalizador)
7: Atualizar D_t:

$$D_{t+1}\left(\mathbf{x}_i, j\right) \leftarrow \frac{D_t(\mathbf{x}_i,j)\exp(-\gamma_t M(y_i,j)h_t(\mathbf{x}_i,j))}{Z_t}$$

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

$$H\left(\mathbf{x}\right) = \underset{y\in\{1,\ldots,C\}}{\arg\min}\sum_{j=1}^{l}\exp\left(-M\left(y, j\right)\sum_{t=1}^{T}\gamma_t h_t\left(\mathbf{x}, j\right)\right)$$

Deve-se observar que neste algoritmo a distribuição D_t trabalha agora sobre pares $(\mathbf{x}_i, j)$, em que j representa índices de colunas de $\mathbf{M}$. Generalizando os resultados de Schapire et al. (1997), Allwein et al. (2000) também apresentam limites na generalização deste algoritmo em termos das margens do classificador final. Demonstra-se que o AB.MO, utilizado da forma apresentada no Algoritmo 3, tende a aumentar as margens dos dados de treinamento rapidamente.

Existem ainda outras variantes do AB, como o AdaBoost.M2 (Freund and Schapire,

1996) e o AdaBoost.MH (Schapire and Singer, 1998). Estes algoritmos requerem, porém, esforço adicional no desenvolvimento do algoritmo base (Schapire, 2002). Esta é uma deficiência sobre a estratégia baseada em códigos (AB.MO), que pode utilizar como base qualquer algoritmo de aprendizado.

5 Experimentos: Avaliação de Técnicas Decomposicionais para SVMs

Esta Seção descreve experimentos conduzidos com o objetivo de avaliar o desempenho de algumas das técnicas decomposicionais apresentadas na Seção 4.1 na geração de SVMs multiclasses. As estratégias testadas foram a 1ct com funções de decisão contínuas (1ct), a tct com integração por meio de voto por maioria (tct-V) e por DAGs, e a ECOC utilizando as medidas de distância *Hamming* (Equação 44) e a *loss-based* (Equação 47 - LECOC). No presente estudo, propôs-se também a utilização de RNs do tipo *Multilayer Perceptron* (MLP) (uma conceituação a respeito dessas RNs pode ser consultada em (Haykin, 1999) e (Braga et al., 2000)) na integração dos classificadores gerados pela decomposição tct (tct-RN). Neste contexto, a RN foi empregada como uma técnica para ponderar as previsões feitas por cada classificador.

Dadas as técnicas citadas, comparações foram realizadas considerando as taxas de acurácia e de padrões desconhecidos dos preditores gerados, o tempo despendido na busca por soluções (tempo de treinamento) e a complexidade dos modelos gerados, medida por meio do número médio de vetores suporte consultados em cada nova previsão.

5.1 Materiais e Métodos

Conjuntos de Dados. Os conjuntos de dados empregados neste estudo foram extraídos do *benchmark* UCI (Blake and Merz, 1998). A Tabela 1 descreve esses conjuntos, ilustrando o número de instâncias, de atributos contínuos e nominais, de classes, uma medida do erro majoritário (que representa a proporção de exemplos da classe com maior número de padrões no conjunto de dados) e se há atributos ausentes.

Instâncias com atributos ausentes foram removidas dos conjuntos *bridges* e *pos-operative*. Este procedimento mantece 70 e 87 instâncias nos respectivos conjuntos. Para o caso da base *splice*, instâncias com atributos diferentes dos nucleotídeos A, T, C e G foram eliminados. Este conjunto ficou então com 3175 exemplos.

A maioria dos conjuntos de dados foram pré-processados para que os dados tivessem média 0 e variância 1. As excessões foram os conjuntos *balance* e *splice*. No conjunto

Conjunto de dados	♯ Instâncias	♯ Atributos (cont., nom.)	♯ Classes	Erro Majoritário	Atributos Ausentes
Balance	625	4 (0, 4)	3	46.1%	não
Bridges	108	11 (0, 11)	6	32.9%	sim
Glass	214	9 (9, 0)	6	35.5%	não
Iris	150	4 (4, 0)	3	33.3%	não
Pos-operative	90	8 (0, 8)	3	71.1%	sim
Splice	3190	60 (0, 60)	3	50.0%	não
Vehicle	846	18 (18, 0)	4	25.8%	não
Wine	178	12 (12, 0)	3	48.0%	não
Zoo	90	17 (2, 15)	7	41.1%	não

Tabela 1: Descrições dos conjuntos de dados multiclasses

balance, vários atributos se tornaram nulos, então este pré-processamento não foi aplicado. No caso do conjunto *splice*, como as SVMs lidam somente com atributos numéricos, os dados utilizados tiveram de ser codificados para um formato numérico. Utilizou-se para tal uma codificação canônica, sugerida no trabalho de Pedersen and Nielsen (1997). Segundo esta codificação, cada nucleotídeo (atributo) das seqüências é codificado com quatro bits, da seguinte forma: A = (0 0 0 1), C = (0 0 1 0) e G = (0 1 0 0) e T = (1 0 0 0). O número de atributos utilizados no caso da base *splice* foi então igual a 240.

Metodologia. Para obter melhores estimativas do desempenho dos métodos avaliados, os conjuntos de dados descritos foram divididos segundo o método *10-fold cross-validation*.

No caso das RNs, os conjuntos de treinamento obtidos foram ainda subdivididos em subconjuntos de treinamento e validação, em uma proporção de 75% e 25%, respectivamente. Enquanto o conjunto de treinamento foi empregado na determinação dos pesos das conexões da rede, o conjunto de validação foi utilizado na avaliação da capacidade de generalização da RN sobre novos exemplos, durante seu treinamento. O treinamento da RN foi interrompido quando o erro de validação iniciava a aumentar, em uma estratégia comumente referenciada como *early-stopping* (Haykin, 1999). Com este procedimento, o efeito de *overfitting* pode ser reduzido. O conjunto de validação foi também empregado na determinação da arquitetura da rede. Várias redes com diferentes arquiteturas foram geradas para cada problema e aquela com menor erro de validação foi escolhida como o classificador neural final. Neste trabalho, as arquiteturas testadas foram de RNs com uma camada intermediária composta de 1, 5, 10, 15, 20, 25 e 30 neurônios. O algoritmo *back-propagation* (Rumelhart et al., 1986) foi empregado no treinamento das redes, com taxa de aprendizado de 0.2. Para geração das RNs, utilizou-se o simulador SNNS (*Sttutgart Neural Network Simulator*) (Zell et al., 1995).

O software aplicado na indução das SVMs foi o simulador SVMToch II (Collobert

Conjunto de dados	1ct	tct-V	tct-RN	DAG	ECOC	LECOC
Balance	96.5±3.1	97.9±2.0	**99.2±0.8**	*98.4±1.8*	90.7±4.7	96.5±3.1
Bridges	58.6±19.6	60.0±17.6	58.6±12.5	**62.9±16.8**	58.6±20.7	*61.4±19.2*
Glass	64.9±14.2	67.3±10.3	*68.2±9.7*	**68.7±11.9**	65.3±14.5	65.8±14.6
Iris	*95.3±4.5*	**96.0±4.7**	**96.0±4.7**	**96.0±4.7**	94.7±6.1	*95.3±4.5*
Pos op.	*63.5±18.7*	**64.6±19.3**	62.2±17.1	61.3±26.7	61.3±18.9	63.5±18.7
Splice	*96.8±0.9*	**96.8±0.7**	83.4±1.7	**96.8±0.7**	93.6±1.7	*96.8±0.9*
Vehicle	85.4±4.5	85.5±4.0	84.4±5.3	**85.8±4.0**	81.9±4.7	*85.5±3.6*
Wine	**98.3±2.7**	**98.3±2.7**	*97.2±3.9*	**98.3±2.7**	97.2±4.1	**98.3±2.7**
Zoo	**95.6±5.7**	*94.4±5.9*	*94.4±5.9*	*94.4±5.9*	**95.6±5.7**	**95.6±5.7**

Tabela 2: Taxas de acurácia de diferentes estratégias multiclasses

and Bengio, 2001). Em todos experimentos realizados, utilizou-se um Kernel Gaussiano com desvio padrão de 5. Outros parâmetros foram mantidos com os valores default do simulador SVMTorch II. Embora melhores valores dos parâmetros das SVMs possam diferir entre as diferentes estratégias multiclasses (e também entre os diferentes conjuntos de dados), eles foram mantidos constantes visando facilitar uma comparação entre as técnicas consideradas.

Os códigos utilizados na decomposição ECOC foram obtidos seguindo uma heurística proposta em (Dietterich and Bariki, 1995). Dado um problema com $3 \leq C \leq 7$ classes, C códigos de tamanho $2^{C-1} - 1$ são gerados. O código da primeira classe é composto somente de uns. Para qualquer outra classe i, em que $i > 1$, ele é composto de 2^{C-i} carreiras alternadas de zeros e uns.

O treinamento, combinações e testes de cada estratégia multiclasses foram implementados na linguagem de programação Perl, com realizações de chamadas ao SVMTorch II e ao SNNS.

5.2 Resultados e Discussões

A Tabela 2 apresenta as taxas de acurácia das estratégias investigadas em cada conjunto de dados. O primeiro e o segundo melhores resultados encontram-se indicados em negrito e itálico, respectivamente. A notação empregada indica a média das acurácias observadas nas dez partições de teste ± o desvio padrão dessas medidas em relação a esta média. A Tabela 3 apresenta a mesma informação para o caso da proporção de exemplos classificados como desconhecidos. Neste caso, os piores resultados são destacados.

De acordo com a Tabela 2, as taxas de acurácia alcançadas pelas diferentes técnicas multiclasses foram próximas. Aplicando o teste estatístico t para dados pareados (Achcar and Rodrigues, 2000) ao primeiro e segundo melhores resultados de cada conjunto de dados, não se pode detectar qualquer significância estatística, a 95% de nível de confidência.

Conjunto de dados	1ct	tct-V	tct-RN	DAG	ECOC	LECOC
Balance	0.0±0.0	*0.5±0.8*	0.0±0.0	0.0±0.0	**8.9±4.5**	0.0±0.0
Bridges	0.0±0.0	*5.2±7.4*	0.0±0.0	0.0±0.0	**7.1±10.1**	0.0±0.0
Glass	0.0±0.0	*4.2±4.0*	0.0±0.0	0.0±0.0	**5.7±5.4**	0.0±0.0
Iris	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	**2.0±3.2**	0.0±0.0
Pos operative	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	**4.4±7.8**	0.0±0.0
Splice	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	**4.7±2.0**	0.0±0.0
Vehicle	0.0±0.0	*0.8±0.8*	0.0±0.0	0.0±0.0	**6.5±2.8**	0.0±0.0
Wine	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	**1.1±2.4**	0.0±0.0
Zoo	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0

Tabela 3: Taxas de padrões desconhecidos de diferentes estratégias multiclasses

Porém, os resultados sugerem que a estratégia mais bem sucedida é a DAG. Por outro lado, a técnica ECOC apresenta, em geral, piores desempenhos. Deve-se observar que a simples modificação desta abordagem com o uso de uma medida de distância baseada em margens (LECOC) melhora os resultados substancialmente.

Em uma comparação entre as técnicas de melhor e pior desempenho em cada conjunto de dados, uma significância estatística de 95% pode ser verificada nos seguintes conjuntos: *balance*, *splice* e *vehicle*.

Comparando as três estratégias para combinação tct, não se pode verificar diferenças estatisticamente significantes em um nível de confidência de 95% em termos de acurácia. A excessão é o conjunto de dados *splice*, maior entre os utilizados, em que a integração por RNs se mostrou pior. Deve-se observar que a abordagem neural apresenta uma tendência em diminuir o desvio padrão nas acurácias reportadas sobre alguns conjuntos de dados, indicando um ganho de estabilidade.

Analisando a Tabela 3, pode-se verificar que o ECOC possui maiores taxas de padrões considerados desconhecidos. Este fato ocorre quando dois ou mais códigos de diferentes classes possuem mesma distância de *Hamming* em relação a uma determinada previsão. Pode-se observar também que o uso de uma medida de distância baseada em margens (LECOC) reduz esta taxa a 0%.

O tct-V figura como outra técnica que leva alguns padrões a não serem classificados. Observa-se, também neste caso, que o uso de diferentes estratégias de combinação (RNs e DAG) reduz as taxas de padrões desconhecidos a 0%.

Outras análises mais refinadas das Tabelas 2 e 3 revelam uma tendência de melhor desempenho do tct-V sobre conjuntos de dados com poucas classes. No caso da integração tct-RN, melhores desempenhos foram verificados em problemas com poucos atributos e poucas classes (*balance* e *iris*). Para o DAG, que apresentou em geral maiores acurácias, bons resultados foram observados sobre duas bases com um número maior de classes (seis classes - *bridges* e *glass*) e três aplicações com muitos atributos (*splice*, *vehicle* e

Conjunto de dados	1ct	tct-V	tct-RN	DAG	ECOC	LECOC
Balance	5.7±1.1	**4.2±1.1**	33.7±1.0	**4.2±1.1**	*4.3±0.5*	*4.3±0.5*
Bridges	**11.3±1.1**	43.4±1.9	50.3±10.8	43.4±1.9	64.3±3.4	64.3±3.4
Glass	**10.2±0.8**	*38.7±2.0*	42.1±3.2	*38.7±2.0*	72.5±2.0	72.5±2.0
Iris	**1.7±1.2**	1.9±1.0	11.7±7.0	1.9±1.0	*1.8±1.2*	*1.8±1.2*
Pos op.	**3.6±0.8**	4.4±0.7	7.3±1.2	4.4±0.7	*3.7±0.7*	*3.7±0.7*
Splice	476.7±17.1	**205.9±1.2**	*388.2±4.2*	**205.9±1.2**	497.4±41.5	497.4±41.5
Vehicle	*21.9±0.9*	**17.4±1.1**	96.7±1.3	**17.4±1.1**	45.2±1.3	45.2±1.3
Wine	*4.9±0.6*	**4.7±0.7**	13.5±19.0	**4.7±0.7**	5.0±0.0	5.0±0.0
Zoo	**12.3±1.2**	*36.8±3.5*	50.3±2.1	*36.8±1.2*	108.6±1.9	108.6±1.9

Tabela 4: Tempo médio de treinamento de diferentes estratégias multiclasses (segundos)

wine). As técnicas 1ct e LECOC, por sua vez, obtiveram as mesmas acurácias sobre dois conjuntos de dados, o *wine* e o *zoo*, que possuem um número de atributos semelhante. Estas observações fornecem indicativos iniciais de que técnica é mais apropriada para tipos específicos de problemas. Porém, mais estudos, sobre outras bases de dados, são necessários para confirmar estas tendências.

Similarmente às Tabelas 2 e 3, a Tabela 4 apresenta o tempo médio despendido no treinamento dos diferentes modelos, em segundos. O primeiro e segundo menores tempos são destacados em negrito e itálico, respectivamente. Todos experimentos foram conduzidos em um processador Pentium II dual com 330 MHz e 128 MB de memória RAM. Comparando os tempos reportados, em geral a técnica mais rápida foi a 1ct. As técnicas ECOC e LECOC, por sua vez, foram em geral as mais lentas nesta fase. O menor tempo de treinamento alcançado pelo tct-V e DAG em alguns conjuntos de dados se deve ao fato de que estes métodos treinam cada SVM em subconjuntos de dados menores, o que acelera sua execução. No caso da abordagem neural, o tempo de treinamento da RN também deve ser levado em consideração, o que fornece um tempo total maior que os do tct-V e DAG.

Finalizando, a Tabela 5 apresenta o número médio de SVs dos modelos induzidos consultados na classificação dos padrões de teste. Como na Tabela 4, os dois melhores resultados encontram-se em negrito e itálico. Este valor está relacionado ao tempo de processamento requerido na classificação de novos exemplos. Um número menor de SVs leva a menores tempos de predição (Mayoraz and Alpaydim, 1998).

Pela Tabela 5 pode-se observar que a integração por DAG apresenta os menores números de SVs consultados em todos casos. Isto significa que a estratégia DAG acelera a fase de classificação de novos exemplos. O tct-V figura como a técnica com segundo menor número médio de SVs. Novamente, a utilização de conjuntos de treinamento mais simples na indução dos classificadores binários pode ser a causa deste resultado. No caso da abordagem neural, deve-se também considerar o tempo de predição da RN. Este pode

Conjunto de dados	1ct	tct-V	tct-RNA	DAG	ECOC	LECOC
Balance	208.2±10.7	*115.3±6.5*	*115.3±6.5*	**69.8±3.5**	208.2±10.7	208.2±10.7
Bridges	*129.6±4.1*	175.3±3.4	175.3±3.4	**59.3±2.7**	879.4±29.7	879.4±29.7
Glass	275.5±7.5	*252.2±6.2*	*252.2±6.2*	**100.6±5.3**	2160.4±67.9	2160.4±67.9
Iris	40.8±2.7	*24.5±1.4*	*24.5±1.4*	**16.7±1.3**	40.7±2.8	40.7±2.8
Pos op.	106.8±7.6	*61.0±6.1*	*61.0±6.1*	**54.2±4.9**	107.3±7.5	107.3±7.5
Splice	5043.0±16.4	*3625.2±10.3*	*3625.2±10.3*	**2577.1±9.0**	5043.0±16.4	5043.0±16.4
Vehicle	669.0±11.1	*469.2±5.0*	*469.2±5.0*	**232.1±9.7**	1339.5±22.7	1339.5±22.7
Wine	75.2±1.5	*54.3±3.4*	*54.3±3.4*	**35.3±3.2**	75.2±1.5	75.2±1.5
Zoo	*132.9.6±4.5*	191.2±6.8	191.2±6.8	**62.0±4.8**	1608.4±65.8	1608.4±65.8

Tabela 5: Número médio de SVs consultados na fase de teste por diferentes estratégias multiclasses

Balance	Bridges	Glass	Iris	Pos operative	Splice	Vehicle	Wine	Zoo
1	5	5	10	1	1	5	30	5

Tabela 6: Número de neurônios intermediários das arquiteturas de RNs no tct-RN

ser medido pelo tamanho das RNs geradas. Quanto maior a rede, mais lenta esta se torna. A Tabela 6 apresenta, para cada conjunto de dados, o número de nodos da camada intermediária das melhores RNs obtidas na combinação tct-RN.

Como conclusão, tem-se que o desempenho geral das diversas técnicas testadas foram similares. Quando requer-se velocidade de classificação, mantendo uma boa precisão, os resultados observados indicam que uma alternativa eficiente para a geração de SVMs multiclasses é a técnica DAG.

6 Conclusão

Embora os LMCs SVM e AB sejam originalmente formulados para geração de classificadores binários, alguns artifícios permitem que este sejam aplicados a problemas multiclasses, em que o número de classes é maior que dois.

Para tal, alguns trabalhos recorrem a reformulações nos algoritmos originais de forma a permitir que um problema multiclasses seja solucionado em um único passo (Freund and Schapire, 1996; Weston and Watkins, 1998; Bredensteiner and Bennet, 1999; Guermer, 2000; Crammer and Singer, 2002).

Seguindo outra abordagem, diversas técnicas decompõem o problema multiclasses em múltiplos subproblemas binários, os quais são então integrados na geração do classificador final (Vapnik, 1998; Kreβel, 1999; Inoue and Abe, 2001; Abe and Inoue, 2002; Platt et al., 2000; Kijsirikul and Ussivakul, 2002; Dietterich and Bariki, 1995; Allwein et al., 2000; Schwenker, 2000; Godbole et al., 2002; Tax and Duin, 2002).

Este capítulo apresentou uma revisão bibliográfica de várias técnicas para geração de LMCs multiclasses.

Também apresentou-se experimentos preliminares comparando os resultados de diferentes estratégias decomposicionais para geração de SVMs multiclases. Como no estudo de Hua and Sun (2001), a integração tct por meio de DAGs se mostrou uma alternativa eficiente.

Referências

Abe, S. and Inoue, T. (2002). Fuzzy support vector machines for multiclass problems. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 113–118.

Abe, S. (2003). Analysis of multiclass support vector machines. In *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA'2003)*, Vienna, Austria.

Achcar, J. A. and Rodrigues, J. (2000). Introdução à estatística para ciências e tecnologia. Technical report, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brasil.

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, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brasil. Disponível em: `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 Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods - Support Vector Learning*, pages 43–51. MIT Press.

Blake, C. L. and Merz, C. J. (1998). UCI repository of machine learning databases. Disponível em: `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Boser, B. E., Guyon, I. L., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburg, Pennsylvania, US.

Braga, A., Carvalho, A. C. P. L. F., and Ludemir, T. (2000). *Redes Neurais Artificiais: Teoria e Aplicações*. Editora LTC.

Bredensteiner, E. J. and Bennet, K. P. (1999). Multicategory classification by support vector machines. *Computational Optimizations and Applications*, 12:53–79.

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.

Collobert, R. and Bengio, S. (2001). SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160.

Cortes, C. and Vapnik, V. N. (1995). Support vector networks. *Machine Learning*, 20(3):273–296.

Crammer, K. and Singer, Y. (2002). On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2-3):201–233.

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*, volume 2, pages 753–756, 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. In *Advances in Neural Information Processing Systems*, volume 8, pages 479–485.

Freund, Y. and Schapire, R. E. (1996). Game theory, on-line prediction and boosting. In *Proceedings of the 9th Annual Conference on Computational Learning Theory*, pages 325–332.

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. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780.

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

Godbole, S., Srawagi, S., and Chakrabarti, S. (2002). Scaling multi-class support vector machines using inter-class confusion. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 513–518, Edmonton, Alberta, Canada. ACM.

Good, I. J. (1965). *The estimation of probabilities: an essay on modern bayesian methods*. MIT Press.

Guermer, Y. (2000). Combining discriminant models with new multiclass SVMs. Technical Report NC-TR-00-086, Neuro COLT, Loria Campus Scientifique.

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(2):415–425.

Hua, S. and Sun, Z. (2001). Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8):721–728.

Inoue, T. and Abe, S. (2001). Fuzzy support vector machines for pattern classification. In *Proceedings of International Joint Conference on Neural Networks (IJCNN 2001)*, volume 2, pages 1449–1454.

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.

Kijsirikul, B. and Ussivakul, N. (2002). Multiclass support vector machines using adaptive directed acyclic graph. In *Proceedings of International Joint Conference on Neural Networks (IJCNN 2002)*, pages 980–985.

Kreβel, U. (1999). Pairwise classification and support vector machines. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press.

Lorena, A. C. and Carvalho, A. C. P. L. F. (2003a). Classificadores de margens largas (*Large Margin Classifiers*). Technical Report 195, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brasil. Disponível em: `ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_195.ps.zip`.

Lorena, A. C. and Carvalho, A. C. P. L. F. (2003b). Introdução às máquinas de vetores suporte (*Support Vector Machines*). Technical Report 192, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brasil. Disponível em: `ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_192.ps.zip`.

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

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

Monard, M. C. and Baranauskas, J. A. (2003). Conceitos de aprendizado de máquina. In Rezende, S. O., editor, *Sistemas Inteligentes - Fundamentos e Aplicações*, pages 89–114. Editora Manole.

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.

Pedersen, A. G. and Nielsen, H. (1997). Neural network prediction of translation initiation sites in eukaryotes: Perspectives for EST and genome analysis. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology (ISMB'97)*, pages 226–233.

Platt, J. C., Cristiani, N., and Shawe-Taylor, J. (2000). Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, 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).

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

Rumelhart, D. E., Hinton, G. E., and Willians, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Foundations*, volume 1, pages 318–362. MIT Press, Cambridge, MA.

Schapire, R. E. and Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 80–91.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

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 6th International Joint Conference on Artificial Intelligence*, pages 1401–1406.

Schapire, R. E. (2002). The boosting approach to machine learning: An overview. In Denison, D. D., Hansen, M. H., Holmes, C. C., Mallick, B., and Yu, B., editors, *Proceedings of the Mathematical Sciences Research Institute (MSRI) Workshop on Nonlinear Estimation and Classification*, pages 149–172, Berkeley, CA.

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.

Schwenker, F. (2000). Hierarquical support vector machines for multi-class pattern recognition. In *Proceedings of the 4th International Conference on Knowledge-Based Intelligent Systems and Allied Technologies*, pages 561–565. IEEE Computer Society Press.

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(3/4):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. (1999). Introduction to large margin classifiers. In Smola, A. J., Barlett, P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 1–28. MIT Press.

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

Souto, M. C. P., Lorena, A. C., Delbem, A. C. B., and Carvalho, A. C. P. L. F. (2003). *Técnicas de Aprendizado de Máquina para problemas de Biologia Molecular*, pages 103–152. Minicursos de Inteligência Artificial, Jornada de Atualização Científica em Inteligência Artificial, XXIII Congresso da Sociedade Brasileira de Computação.

Tax, D. M. J. and Duin, R. P. W. (2002). Using two-class classifiers for multiclass classification. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, volume 2, pages 20124–20127, Quebec City, Canada. IEEE Computer Society Press.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, 2 edition.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.

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

Zell, A., Mamier, G., Vogt, M., Hübner, R., Döring, S., Herrmann, K., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeorgiou, A., Posselt, D., Schreiner, T., Kett, B., Clemente, G., and Wieland, J. (1995). SNNS - Stuttgart Neural Network Simulator - user manual-version 4.1. Technical Report 6/95, Institute for Parallel and Distributed High Performance Systems (IPVR), University of Stuttgart.