UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

**TESE DE MESTRADO**

# Support Vector Machines, Inferência Transdutiva e o Problema de Classificação

**Autor : Robinson Semolini**
**Orientador : Prof. Dr. Fernando José Von Zuben**

Tese apresentada à Pós-graduação da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como requisito parcial à obtenção do grau de **MESTRE EM ENGENHARIA ELÉTRICA**
**Área de Concentração:** Engenharia de Computação.

**Banca Examinadora :**
Profa. Dra. Nancy Lopes Garcia - DEP. ESTATÍSTICA, IMECC, UNICAMP
Prof. Dr. Márcio Luiz de Andrade Netto - DCA, FEEC, UNICAMP
Prof. Dr. Leandro Nunes de Castro Silva - DCA, FEEC, UNICAMP

Campinas - SP - Brasil
Dezembro de 2002

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Se54s

Semolini, Robinson
Support vector machines, inferência transdutiva e o problema de classificação / Robinson Semolini.--Campinas, SP: [s.n.], 2002.

Orientador: Fernando José Von Zuben.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Inteligência artificial. 2. Classificação. 3. Estatística - Análise. 4. Kernel, Funções de. 5. Otimização matemática. I. Von Zuben, Fernando José. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

# Resumo

Esta dissertação aborda o problema de classificação de dados em duas classes já existentes, utilizando como mecanismo de classificação uma técnica de aprendizado de máquina denominada Support Vector Machines (SVM). Foi dedicado um esforço considerável na apresentação dos aspectos teóricos envolvidos, vinculados à teoria do aprendizado estatístico, incluindo motivação, interpretação geométrica e perspectiva analítica. A tarefa de classificação, geralmente realizada com base nos princípios da inferência indutiva, foi executada pelos princípios da inferência transdutiva, conduzindo à técnica SVM transdutiva, ou simplesmente TSVM. Após um estudo comparativo dos vários pacotes de software disponíveis na literatura, optou-se por aqueles com melhor desempenho em termos de custo computacional, e foi possível classificar em um único passo cada amostra do conjunto de predição. TSVM foi comparada com o método tradicional, SVM indutiva, em uma série exaustiva de experimentos inéditos, apresentando resultados promissores. Em uma das aplicações, TSVM foi utilizada pela primeira vez na forma indutiva para predizer a classificação de futuras amostras.

# Abstract

This dissertation deals with the problem of data classification into two already existing classes, using as classification mechanism a learning machine technique called Support Vector Machines (SVM). Considerable efforts were devoted to the presentation of the pertinent theoretical aspects, associated with statistical learning theory, including motivation, geometric interpretation and analytic perspective. The classification task, generally carried out by the principles of inductive inference, was performed by the principles of transductive inference, guiding to the transductive SVM, or simply TSVM. After a comparative analysis of the software packages available in the literature, the ones with best performance in terms of computational cost were chosen, and it was possible to do a one-step classification of each sample belonging to the prediction set. TSVM was compared with the traditional method, inductive SVM, in an exhaustive series of innovative experiments, with promising results. In one of the applications, TSVM was used by the first time in the inductive form to predict the classification of future samples.

*É com muito orgulho que dedico este trabalho à minha esposa Andréa Ferreira, pela sua grande paciência comigo e por ser a pessoa que mais me incentivou na realização desta dissertação.*

# Agradecimentos

Ao meu orientador, professor Fernando José Von Zuben, pelos seus valiosos conselhos e sugestões, sua boa-vontade e dedicação à tarefa de ajudar-me, por ter acreditado muito neste trabalho e principalmente por mostrar-me através de seu comportamento o que consiste o sentido e a ética da pesquisa acadêmica.

Ao Unibanco, principalmente a Julio César de Almeida Guedes e José Ernesto Turchiari, por terem dado a oportunidade e o apoio necessário à execução deste trabalho.

A todos os meus colegas de trabalho do Unibanco pelo incentivo e motivação, principalmente a Maira S. P. Peris, por ter muitas vezes me ajudado de maneira a permitir a minha ausência do trabalho em situações difíceis.

À Faculdade de Engenharia Elétrica e de Computação e ao Departamento de Engenharia de Computação e Automação Industrial (DCA) pelo acesso às instalações e pelos recursos computacionais disponibilizados.

Aos membros da banca examinadora pelas contribuições na geração final desta dissertação.

Aos meus pais António e Ophelia pelo apoio e por terem a visão de sempre incentivarem o estudo de seus filhos.

A todos os meus familiares que torceram por mim: Jefferson, Ivan, João, Anna, Márcia e Miguel.

Aos meus amigos Daniel Arraes e Ronaldo Picinini, por terem me ajudado muito no começo do curso.

A todos que de alguma forma me ajudaram ou me motivaram no desenvolvimento desta tese.

A Deus por ter me dado forças para superar todos os obstáculos.

*Que de alguma forma a realização desta dissertação, em que o Unibanco, compreendendo os benefícios da aproximação com o meio acadêmico, me deu todo o apoio necessário, sirva de incentivo a outros intercâmbios entre a Iniciativa Privada e as Universidades.*

# Índice

**Capítulo 1 - Introdução**.....**1**

1.1 Apresentação e Motivação.....1

1.2 Pontos Importantes sobre Support Vector Machines.....4

1.3 Objetivos Principais e Contribuições.....6

1.4 Descrição do Conteúdo dos Demais Capítulos.....7

**Capítulo 2 - O Hiperplano Ótimo**.....**9**

2.1 Hiperplano ótimo para classes linearmente separáveis.....9

2.2 Hiperplano ótimo para classes não linearmente separáveis.....11

**Capítulo 3 - Fundamentos Básicos e Teoria do Aprendizado Estatístico**.....**15**

3.1 Introdução.....15

3.2 Conceitos Relevantes de Otimização.....16

3.3 Produto Interno Kernel.....19

3.3.1 Espaço característico.....20

3.3.2 Teorema de Mercer.....22

3.3.3 Tipos de produto interno kernel mais utilizados em SVM.....23

3.4 Teoria do Aprendizado Estatístico.....24

3.4.1 Condições para a Consistência e Convergência da Minimização do Risco Empírico .....25

3.4.2 Dimensão VC.....28

3.4.3 Limitantes para a Generalização.....31

3.4.4 Princípio da Minimização do Risco Estrutural.....33

**Capítulo 4 - Support Vector Machines para o Problema de Classificação**.....**37**

4.1 Introdução.....37

4.2 Caso 1 - Classes linearmente separáveis.....38

4.2.1 Classes linearmente separáveis no espaço original.....38

4.2.2 Classes linearmente separáveis no espaço característico.....41
4.3 Caso 2 - Classes não linearmente separáveis.....42
4.4 Propriedades Estatísticas do Hiperplano Ótimo.....47
4.5 Fatores de Custo.....49
4.6 Convertendo a resposta da SVM em probabilidade.....51

**Capítulo 5 - Algoritmo de Implementação para Support Vector Machines.....55**
5.1 Introdução.....55
5.2 Abordagem do Problema.....56
5.3 Algoritmo geral para a tarefa de Decomposição.....57
5.4 Selecionando Bons Conjuntos de Trabalho.....59
5.5 Reduzindo o Número de Variáveis.....61
5.6 Questões para uma Implementação Eficiente.....63
5.6.1 Critério de Parada.....63
5.6.2 Cálculo do Gradiente e do Critério de Parada.....64
5.6.3 Armazenagem do cálculo dos Produtos Internos Kernel.....65
5.7 Outras Maneiras de Implementação da SVM.....65
5.7.1 Razão da escolha do algoritmo SVM$^{light}$.....67

**Capítulo 6 - Inferência Transdutiva aplicada a Support Vector Machines.....69**
6.1 Introdução.....69
6.2 Inferência Transdutiva.....69
6.3 Aspectos da Teoria do Aprendizado Estatístico.....72
6.4 Support Vector Machines associada à Inferência Transdutiva.....74
6.5 Algoritmo de implementação para SVM Transdutiva (TSVM).....78
6.5.1 Abordagem do Problema.....78
6.5.2 O Algoritmo TSVM.....79
6.5.3 Análise do funcionamento do algoritmo TSVM.....82
6.5.4 Outras maneiras de implementação da Inferência Transdutiva aplicada a SVM .....83

**Capítulo 7 - Aplicações**.....**85**

7.1 Aplicação 1: "Support Vector Machines Transdutiva para o Diagnóstico de Câncer e Classificação de Dados de Expressão Gênica".....85

7.1.1 Motivação.....85

7.1.2 Introdução.....86

7.1.3 Análise dos Dados e Resultados.....87

7.1.3.1 Tumor infantil *Small Round Blue Cell*…….....…….87

7.1.3.2 Dados de Expressão Gênica da levedura de brotamento *Saccharomyces cerevisiae*.....92

7.1.4 Conclusões.....100

7.2 Aplicação 2: "Construindo Modelos de Concessão de Crédito Bancário para a Predição de Inadimplência, com variações da quantidade de Amostras de Treinamento".....102

7.2.1 Motivação.....102

7.2.2 Introdução.....103

7.2.3 Dados.....104

7.2.4 Medida de Desempenho.....106

7.2.5 Especificações das Técnicas.....106

7.2.6 Resultados.....108

7.2.7 Conclusões.....110

**Capítulo 8 - Conclusões**.....**113**

8.1 Questões em Aberto e Perspectivas Futuras.....115

8.2 Possíveis extensões deste trabalho.....117

**Referências Bibliográficas**.....**119**

**Índice Remissivo de Autores**.....**127**

# Capítulo 1

# Introdução

## 1.1 Apresentação e Motivação

O problema de classificação, foco principal desta dissertação, pode ser definido formalmente como o processo pelo qual padrões ou sinais recebidos são distribuídos por um número prescrito de classes (categorias). Presente em todas as áreas de atuação científica, a classificação representa um amplo conjunto de problemas de grande significado prático.

Classificação é uma tarefa que o ser humano freqüentemente executa sem maiores dificuldades. Recebemos dados (padrão ou sinal) do mundo exterior através de nossos sentidos e podemos reconhecer, em algum determinado contexto, a que classe pertencem estes dados. Podemos fazer isto quase que imediatamente e com praticamente nenhum esforço, caso o conhecimento necessário para executar a classificação já tenha sido adquirido através de um processo de aprendizagem.

Porém, nos casos em que a tarefa de classificação deve ser feita considerando dados pertencentes a espaços de grande dimensão e nos casos em que os atributos disponíveis para caracterizar cada amostra não esclarecem de forma óbvia o que diferencia um padrão pertencente a uma classe de outro pertencente a outra classe, o ser humano vai encontrar muitas dificuldades para executar a classificação. Sendo assim, a automatização do processo de classificação passa a ser de grande interesse e sua viabilidade aumenta conforme cresce o poder de processamento e memória dos computadores.

Técnicas estatísticas são utilizadas na maioria dos casos para resolver o problema de classificação pela utilização de computadores, como exemplos: Regressão Logística, Análise Discriminante e Árvore de Decisão. Mais recentemente, a técnica de inteligência artificial que emprega modelos artificiais de redes neurais tem sido muito empregada na execução desta tarefa.

Para o caso da existência de duas classes no formato simples de identificador binário "sim/não" ou "pertence/não pertence", o problema é dito ser de *classificação binária*. Quando existe um número finito e maior do que dois de categorias ou classes, o problema é dito ser de *classificação em múltiplas classes*.

Esta dissertação abordará o caso mais simples: o problema de classificação binária de dados. Além disso, aqui a tarefa de classificação, a qual é geralmente realizada pelos princípios da tradicional inferência indutiva, será executada pelos princípios da inferência transdutiva, originalmente proposta junto à teoria do aprendizado estatístico por Vapnik (1998). Além disso, será utilizado como mecanismo de classificação a técnica de aprendizado de máquina denominada Support Vector Machines (Vapnik, 1995), que vem despertando muito interesse nos últimos anos.

Utilizando a inferência transdutiva, a estimação da classe para os dados de interesse é produzida em um único passo. Isto representa um modo alternativo ao caso da tradicional inferência indutiva, que necessita de dois passos:

- O primeiro passo, o *indutivo*, que consiste em descobrir a dependência funcional entre as variáveis de entrada e as variáveis de saída;
- O segundo passo, o *dedutivo*, que utiliza esta dependência funcional para realizar a classificação dos dados de interesse.

A partir dos princípios da inferência transdutiva, utiliza-se no treinamento do classificador (técnica de classificação) dois conjuntos de dados: o tradicional de treinamento e o de predição. No caso do conjunto de treinamento, os dados já estão previamente classificados em suas classes, e para o conjunto de predição, os dados ainda não estão classificados em suas respectivas classes. O objetivo é classificar estes dados pertencentes ao conjunto de predição. Treinando o classificador com estes dois conjuntos de dados, será possível classificar os dados do conjunto de predição diretamente em um único passo.

De acordo com Vapnik (1998), a inferência transdutiva pode ser vista como uma das direções de pesquisa mais promissoras no desenvolvimento da teoria do aprendizado estatístico, que poderá ter uma enorme influência não somente em discussões técnicas dos métodos de generalização, mas também no entendimento das maneiras de inferência adotadas pelo ser humano.

Contudo, Vapnik (1998) ressalta que, desde que a inferência transdutiva foi discutida em 1974 (Vapnik & Chervonenkis, 1974), poucos artigos foram publicados nesta área. Por outro lado, em diversos experimentos, principalmente com pequenos conjuntos de dados de treinamento, foi demonstrada a vantagem de se utilizar esta inferência contra a inferência indutiva, com uma significativa redução no número de erros de classificação no conjunto de predição (Joachims, 1999b ; Demiriz & Bennett, 2000).

As técnicas (ou mecanismos) já apresentadas na literatura, e que adotaram os princípios da inferência transdutiva para gerar uma melhor generalização e aumento do desempenho, são:

- Support Vector Machines, técnica sugerida inicialmente por Vapnik (1998) para ser utilizada associada com os princípios da inferência transdutiva.
- Árvore de Decisão (Wu *et al*. , 1999);
- Regressão Linear (Cataltepe & Magdon-Ismael, 1998);
- Mixture of Experts (Miller & Uyar, 1997).

Como ferramenta de classificação a ser adaptada aos princípios da inferência transdutiva, será utilizado neste estudo a técnica de aprendizado de máquina Support Vector Machines (SVM), introduzida por Vapnik em 1992 (Boser *et al*., 1992), e que, em poucos anos desde que foi introduzida, já apresenta um desempenho superior à maioria dos outros métodos em uma ampla variedade de aplicações.

Wahba *et al.* (2001) escreveram como um resultado do Workshop em Estimação e Classificação Não-Lineares em Berkeley, que o recente livro de Cristianini & Shawe-Taylor (2000), "Uma introdução a Support Vector Machines", tem o impressionante ranking (para um livro técnico) na livraria Amazon.com (http://www.amazon.com) como um dos 4.500 livros mais populares. Repetindo uma pesquisa sugerida por Grace Wahba na Web, pelo site http://www.google.com, visando localizar páginas da internet a partir da palavra-chave "Support Vector Machines", foi retornada uma lista de 230.000 itens. Estes resultados mostram o porquê da grande procura pela área por pesquisadores envolvidos em técnicas supervisionadas para aprendizado de máquina.

As áreas mais tradicionais com aplicações de SVM na forma tradicional (a indutiva) estão citadas a seguir, porém muitas outras aplicações podem ser encontradas na literatura:

- Reconhecimento de Dígitos Escritos à Mão (Boser *et al*., 1992 ; Vapnik, 1995);
- Reconhecimento de Imagem (Pontil & Verri, 1998 ; Chapelle *et al*. , 1999);

- Classificação de Textos (Joachims, 1998 ; Dumais *et al*. , 1998);
- Bioinformática - Análise de dados de expressão gênica (Brown *et al*. , 2000);
- Bioinformática - Detecção de Proteínas Homólogas (Jaakkola *et al*. , 1999);
- Detecção de imagens da Face Humana, perante qualquer outra imagem (Osuna *et al*., 1997b);
- Database Marketing : Prospecção de clientes para a aquisição de novos produtos (Bennet *et al*. , 1998).

Quanto às aplicações da SVM modificada para atender os princípios da inferência transdutiva, a área com maior aplicação é a de Classificação de Textos (Joachims, 1999b; Nigam *et al*., 1998; Blum & Mitchell, 1998). Importantes resultados nesta mesma linha de aplicação também foram obtidos por Demiriz & Bennett (2000) e Fung & Mangasarian (1999).

## 1.2 Pontos Importantes sobre Support Vector Machines

SVM implementa um mapeamento não-linear (executado por um produto interno kernel escolhido a priori) dos dados de entrada para um espaço característico de alta-dimensão, em que um hiperplano ótimo é construído para separar os dados linearmente em duas classes. Quando os dados de treinamento são separáveis, o hiperplano ótimo no espaço característico é aquele que apresenta a máxima margem de separação. Para dados de treinamento em que as amostras das diversas classes apresentam superposição (dados não separáveis), uma generalização deste conceito é utilizada.

SVM baseia-se nos princípios da minimização do risco estrutural, proveniente da teoria do aprendizado estatístico, a qual está baseada no fato de que o erro do algoritmo de aprendizagem junto aos dados de validação (erro de generalização), é limitado pelo erro de treinamento mais um termo que depende da dimensão VC (dimensão Vapnik e Chervonenkis), que é uma medida da capacidade de expressão de uma família de funções. O objetivo é construir um conjunto de hiperplanos tendo como estratégia a variação da dimensão VC, de modo que o risco empírico (erro de treinamento) e a dimensão VC sejam minimizados ao mesmo tempo. Será visto que, na construção do hiperplano ótimo, a maximização da margem de separação implica na minimização da dimensão VC. Desta maneira, o hiperplano ótimo efetua uma completa realização do princípio de minimização do risco estrutural.

O treinamento da SVM consiste em um problema de otimização quadrático que é atrativo pela garantia da convergência para um mínimo global da superfície de erro (exceto quando algum problema de precisão numérica está presente), onde o erro refere-se à diferença entre a resposta desejada e a saída da SVM.

A transformação do problema de otimização primal em sua representação dual será muito importante para SVM, pois permitirá que o problema de dimensionalidade seja deixado de lado pela utilização da representação dual do problema de otimização, que calcula os parâmetros do hiperplano ótimo tendo os dados de treinamento na forma de produto interno e assim formando uma matriz quadrada de mesma dimensão da quantidade de dados de treinamento. Assim, o número de parâmetros ajustados não depende do número de atributos sendo utilizados, ou seja, da dimensão do espaço a que pertencem os dados de treinamento.

O uso do produto interno kernel oferece uma solução alternativa para projetar os dados em um espaço característico de alta dimensão, aumentando o desempenho da SVM. A utilização da representação dual torna isto possível, pois os dados de treinamento nunca aparecem isolados, mas na forma de produto interno entre pares de amostras. Substituindo o produto interno por uma escolha apropriada de um tipo de produto interno kernel, isto conduzirá implicitamente a um mapeamento não-linear para um espaço característico de alta dimensão, sem aumentar o número de parâmetros ajustados.

Uma das maiores vantagens da SVM é a sua flexibilidade. Utilizando os conceitos básicos de maximização de margem, dualidade e produto interno kernel, pode-se adaptar o problema de classificação binária (apenas com duas classes), que foi a abordagem que originou a formulação da SVM, para resolver muitos outros tipos de problemas. A seguir, estão os problemas mais conhecidos resolvidos por SVM :

- Regressão, que foi o segundo problema a ser abordado por SVM, apenas com a modificação na formulação original da função-objetivo para uma em que o erro, medido pela distância do valor estimado em relação ao valor real, é igual a zero para valores pequenos desta distância, e de valor crescente para quando a distância ao valor real é maior do que um determinado limiar (Vapnik, 1995 ; Cristianini & Shawe-Taylor, 2000);
  Envolvendo Regressão e SVM temos as seguintes aplicações:
  - Técnicas Bayesianas para Regressão utilizando SVM (Chu *et al*., 2001);
  - Quadrados Mínimos para SVM (Suykens *et al*., 2000 ; Van Gestel *et al*. , 2002);

  - Quadrados Mínimos Ponderados para SVM (Suykens *et al*., 2002);
  - Kernel Logistic Regression (Keerthi *et al*., 2002);
  - Ensembles utilizando SVM (Lima *et al*., 2002).
- Classificação em Múltiplas Classes (Vapnik, 1998 ; Weston & Watkins , 1999);
- Técnicas de Clusterização (Ben-Hur *et al*. , 2001);
- Predição de Séries Temporais (Muller *et al*. , 1999);
- Detecção de Novidade (*Novelty Detection*) (Bennett & Campbell, 2000);
- Estimação de Densidades (Vapnik, 1998 ; Vapnik & Mukherjee, 1999).

## 1.3 Objetivos Principais e Contribuições

Um dos objetivos desta dissertação é apresentar uma ampla explanação da técnica de aprendizado de máquina denominada Support Vector Machines, que baseada em um elenco consistente de referências bibliográficas, irá oferecer uma leitura introdutória original, por evidenciar e formalizar devidamente as idéias principais, em todos os seus detalhes mais relevantes.

Além disso, é dado destaque à explicação dos princípios da inferência transdutiva, tentando não se prender a aspectos teóricos, mas fornecendo os esclarecimentos de forma simples e didática, recorrendo inclusive a interpretações geométricas.

Foi realizado um estudo criterioso dos algoritmos computacionais disponíveis na literatura e aqueles que apresentavam as melhores relações de custo computacional foram adotados para a fase de aplicação a problemas práticos de classificação. As duas aplicações apresentadas nesta dissertação realizam a classificação binária com base em conceitos de inferência transdutiva associados à técnica Support Vector Machines, e destacam-se as seguintes contribuições :

- **Aplicação 1** : "Support Vector Machines Transdutiva para Diagnóstico de Câncer e Classificação de Dados de Expressão Gênica".

  Esta é a primeira aplicação dos princípios da inferência transdutiva a um problema de Bioinformática (Semolini & Von Zuben, 2002), apresentando resultados promissores, com uma série de vários experimentos para a comparação com o método indutivo. Ambos os métodos utilizando SVM como técnica de classificação.

  São apresentados também, resultados comprovando a existência de correlação entre a complexidade do problema de classificação, a porcentagem de amostras que se

caracterizam como vetores-suporte e o aumento da diferença entre o desempenho do método transdutivo comparado ao indutivo.

- **Aplicação 2** : "Construindo Modelos de Concessão de Crédito Bancário com pequenas amostras para predição de inadimplência".

  Esta aplicação responde à questão proposta por Joachims (1999b): "Será possível utilizar a função de decisão, obtida com a ajuda dos princípios da inferência transdutiva, para predizer de forma indutiva futuras amostras de predição?".

  Resultados satisfatórios foram obtidos com esta aplicação, mostrando uma melhora significativa da capacidade de generalização e aumento do desempenho da função de decisão aplicada à predição de novas amostras.

## 1.4 Descrição do Conteúdo dos Demais Capítulos

O Capítulo 2 desta dissertação apresenta o conceito do Hiperplano Ótimo, primeiro passo para a introdução da formulação da técnica Support Vector Machines. O Capítulo 3 fornece os fundamentos básicos dos conceitos relevantes de Otimização, Produto Interno Kernel e Teoria do Aprendizado Estatístico. A partir dos conceitos destes dois capítulos será possível resolver o problema de como encontrar o hiperplano ótimo, descrito no Capítulo 4, que levará à introdução da técnica de aprendizado de máquina denominada Support Vector Machines, formulando todas as suas demonstrações matemáticas e detalhando suas propriedades.

O Capítulo 5 descreve em detalhe um dos melhores algoritmos de implementação da técnica Support Vector Machines denominado $SVM^{light}$ (Joachims, 1999a), e uma visão geral de outros algoritmos existentes na literatura.

O Capítulo 6 pode ser considerado uma segunda parte desta dissertação, onde é introduzido os princípios da Inferência Transdutiva, mostrando seus aspectos teóricos baseados na teoria do aprendizado estatístico. É apresentada a aplicação dos conceitos da inferência transdutiva tendo SVM como técnica de classificação, assim como o algoritmo proposto por Joachims (1999b) para a sua implementação, denominado TSVM.

No Capítulo 7 é apresentado o resultado de duas aplicações envolvendo inferência transdutiva associada com SVM, sempre comparando os resultados com o método indutivo, e no Capítulo 8 encontram-se os comentários conclusivos sobre a pesquisa realizada e sobre os resultados obtidos, além de sugestões de futuras pesquisas e possíveis extensões deste trabalho.

# Capítulo 2

# O Hiperplano Ótimo

## 2.1 Hiperplano ótimo para classes linearmente separáveis

Quando o aprendizado supervisionado é aplicado ao problema de classificação, as amostras de treinamento são formadas pelo conjunto de dados de entrada associados às suas correspondentes respostas pré-classificadas (rótulos ou dados de saída). Após o treinamento, o objetivo é classificar novas amostras, ainda não rotuladas.

Considere o seguinte conjunto de dados de treinamento:

$$( x_i , y_i )_{\,1 \le i \le N} \ , \quad x \in R^m, \quad y \in \{+1, -1\},$$

onde $x_i$ é o dado de entrada para a amostra $i$ e $y_i$ é a correspondente resposta desejada.

Classificações binárias são freqüentemente realizadas pelo uso de funções $g : X \subseteq R^m \rightarrow R$ com a seguinte estratégia : as amostras são designadas para a classe positiva, se $g(x) \geq 0$, e caso contrário, para a classe negativa.

Será considerado nesta seção que as classes representadas pelos rótulos $y_i = +1$ e $-1$ são linearmente separáveis. A superfície de decisão será representada por um hiperplano na forma :

$$g(x) = ( w^T x ) + b = 0 \ , \qquad \textbf{(2.1)}$$

onde $w \in R^m$ é o vetor de pesos, e $b \in R$ é o intercepto.

Assim podemos aplicar a seguinte estratégia de decisão:

$$\begin{aligned} ( w^T x ) + b \geq 0 \qquad & \text{para } \ y = +1; \\ ( w^T x ) + b < 0 \qquad & \text{para } \ y = -1. \end{aligned} \qquad \textbf{(2.2)}$$

Para descrever o lugar geométrico dos hiperplanos separadores, será utilizada a seguinte forma canônica (onde o vetor $w$ e o escalar $b$ são re-escalados de tal maneira a atender as desigualdades):

$$( w^{\mathrm{T}}x ) + b \geq +1 \qquad \text{para } y = +1; \tag{2.3}$$
$$( w^{\mathrm{T}}x ) + b \leq -1 \qquad \text{para } y_{\mathrm{i}} = -1.$$

A seguir, é apresentada a notação compacta para as desigualdades (2.3):

$$y\,[( w^{\mathrm{T}}x) + b] \geq 1. \tag{2.4}$$

Para um dado vetor de pesos $w$ e intercepto $b$, a separação entre o hiperplano $g(x) = (w^{\mathrm{T}}x) + b = 0$ e o dado de entrada mais perto é chamada de *margem de separação* denotada por $\rho$. Sempre que for possível obter um $\rho > 0$, existirão infinitos hiperplanos, dentre os quais se busca um hiperplano particular em que a *margem de separação* $\rho$ *é maximizada.* De acordo com esta condição, a superfície de decisão é dita ser *o hiperplano ótimo* e a técnica de aprendizado de máquina utilizado para a determinação deste hiperplano é denominada Support Vector Machines (SVM), sendo que os dados de treinamento que se encontram à distância $\rho$ do hiperplano são chamados vetores-suporte (*support vectors*).

O conceito de hiperplano ótimo foi desenvolvido por Vapnik e Chervonenkis em 1965 (Vapnik & Chervonenkis, 1974). A Figura 2.1 apresenta uma visão geométrica da construção do hiperplano ótimo para um espaço bi-dimensional, além da interpretação dos vetores-suporte.

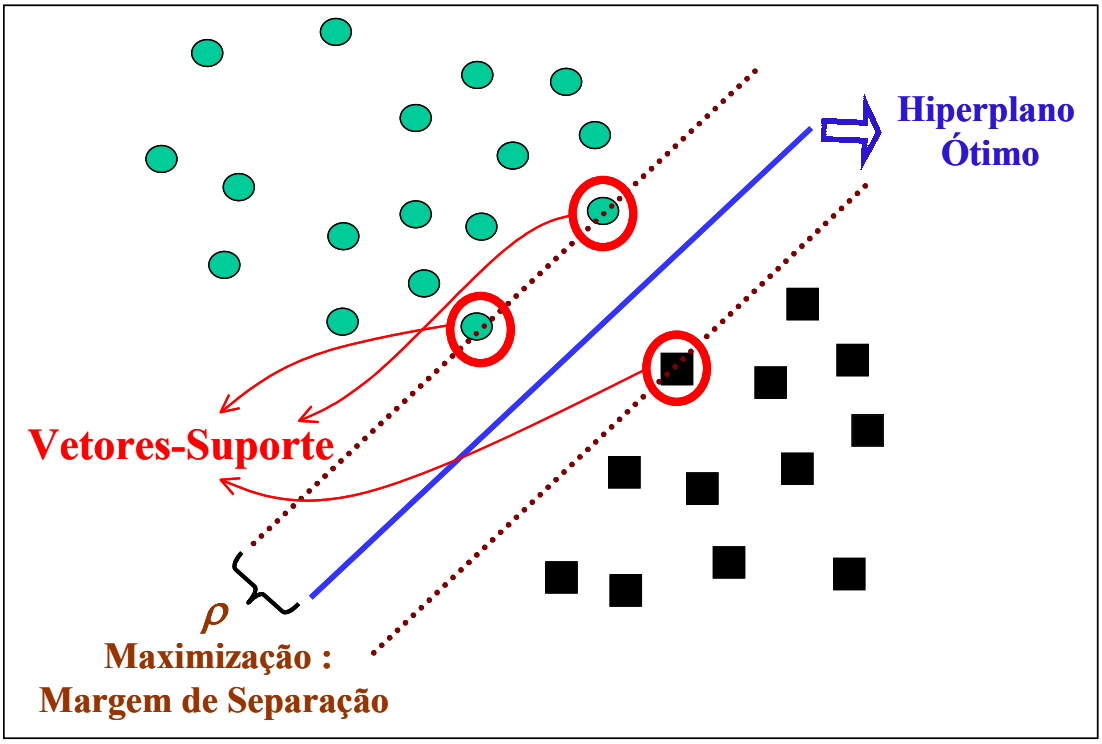


**Figura 2.1** : O hiperplano ótimo separando os dados com a máxima margem $\rho$, os vetores-suporte (*support vectors*) e uma distribuição dos dados no $R^2$.

Os dados para os quais o resultado da equação (2.4) é igual a 1 são os vetores-suporte, pois são aqueles que se encontram à distância $\rho$ do hiperplano ótimo.

Os vetores-suporte exercem um papel importante nas operações deste tipo de aprendizagem de máquina. Em termos conceituais, eles são os pontos que se encontram mais perto da superfície de decisão e, portanto, são os de classificação mais difícil. Como tal, eles têm uma relação direta com a localização da superfície de decisão. Veremos no Capítulo 4 uma explicação mais detalhada sobre os vetores-suporte.

Considere a forma canônica (2.3) com a seguinte modificação (hiperplanos com a margem igual a 1 são conhecidos como hiperplanos canônicos):

$$\begin{aligned} &( w^{\mathrm{T}}x^{\mathrm{sv}+} ) + b = +1 \ , \quad \text{sendo } x^{\mathrm{sv}+} \text{ um vetor-suporte pertencente à classe } y = +1; \\ &( \mathrm{w}^{\mathrm{T}}\mathrm{x}^{\mathrm{sv}-} ) + b = -1 \ , \quad \text{sendo } x^{\mathrm{sv}-} \text{ um vetor-suporte pertencente à classe } y = -1. \end{aligned} \qquad \textbf{(2.5)}$$

Para calcular a distância algébrica dos vetores-suporte para o hiperplano ótimo, ou seja, o valor da margem $\rho$, é preciso primeiro normalizar o vetor de pesos $w$, e usando a equação do hiperplano canônico (2.5), temos :

$$\rho = \frac{1}{2}\left[\left(\left(\frac{w}{\|w\|}\right)^{T} x^{sv+}\right) - \left(\left(\frac{w}{\|w\|}\right)^{T} x^{sv-}\right)\right] = \frac{1}{\|w\|}. \qquad \textbf{(2.6)}$$

A equação (2.6) mostra que maximizar a margem de separação entre as classes é equivalente a minimizar a norma euclidiana do vetor de pesos $w$.

Em resumo, o hiperplano ótimo definido pela equação (2.4), apresenta um vetor de pesos $w$ que leva à máxima separação entre as amostras positivas e negativas. Esta condição ótima é alcançada minimizando a norma euclidiana do vetor de pesos $w$.

## 2.2 Hiperplano ótimo para classes não linearmente separáveis

Considere o caso mais difícil de classificação, quando as classes não são linearmente separáveis. Dadas as amostras de treinamento, não é possível construir um hiperplano separador sem encontrar erros de classificação. Todavia, é possível encontrar um hiperplano que minimiza a probabilidade do erro de classificação junto às amostras de treinamento.

Sendo assim a margem de separação entre as classes é dita ser flexível (*soft*), pois irão existir pontos $(x_{\mathrm{i}}, y_{\mathrm{i}})_{1 \le \mathrm{i} \le N}$ que violam a inequação (2.4).

Esta violação pode acontecer de três maneiras:

- O ponto $(x_i,y_i)$ encontra-se dentro da região de separação, porém no lado correto da superfície de decisão, ver Figura 2.2a. Neste caso as classes são linearmente separáveis, porém houve uma escolha incorreta do hiperplano;
- O ponto $(x_i,y_i)$ encontra-se no lado incorreto da superfície de decisão, porém dentro da região de separação, ver Figura 2.2b. Também neste caso as classes são linearmente separáveis, porém houve uma escolha por um hiperplano de maior margem;
- O ponto $(x_i,y_i)$ encontra-se no lado incorreto da superfície de decisão e fora da região de separação, ver Figura 2.2c.

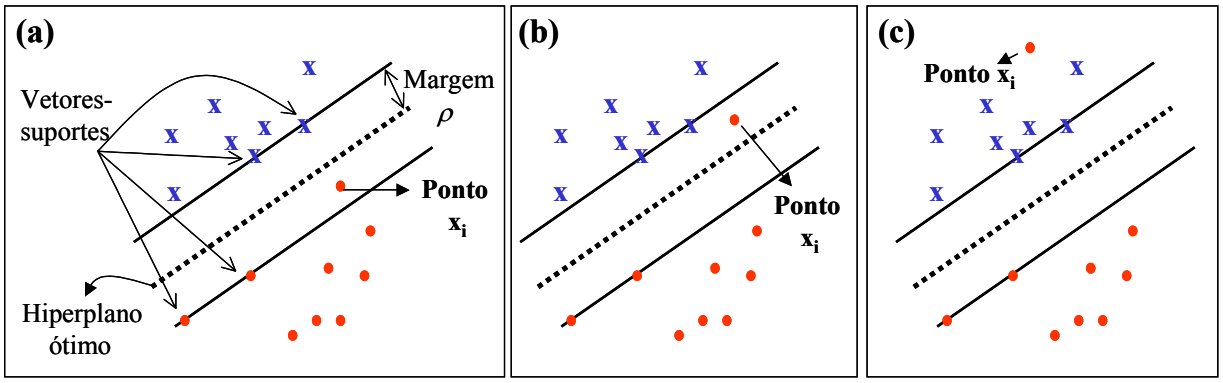


**Figura 2.2** : (a) Ponto $x_i$ encontra-se dentro da região de separação e no lado correto;
(b) Ponto $x_i$ encontra-se no lado incorreto da superfície de decisão, porém dentro da região de separação;
(c) Ponto $x_i$ encontra-se no lado incorreto da superfície de decisão e fora da região de separação.

Note que existe classificação correta no caso (a) e incorreta nos casos (b) e (c).

Para tratar o problema de classes não linearmente separáveis, introduziremos uma nova variável não-negativa, $\{\xi_i\}_{1 \le i \le N}$, na definição de hiperplano separador (superfície de decisão) apresentada a seguir:

$$y_i [( w^T x_i ) + b] \ge 1 - \xi_i , \quad i = 1, \ldots , N . \qquad \textbf{(2.7)}$$

Os escalares $\xi_i$ são chamados de *variáveis de folga*, e medem os desvios dos pontos $(x_i , y_i)_{1 \le i \le N}$ para a condição ideal de separação das classes. Para $0 \le \xi_i \le 1$, o ponto encontra-se dentro da região de separação mas do lado correto da superfície de decisão. Para $\xi_i > 1$, o ponto encontra-se do lado incorreto do hiperplano separador. Os vetores suportes são os pontos em que o resultado da equação (2.7) é igual a $1-\xi_i$ , mesmo que $\xi_i > 0$. Ilustramos isto através da Figura 2.3.

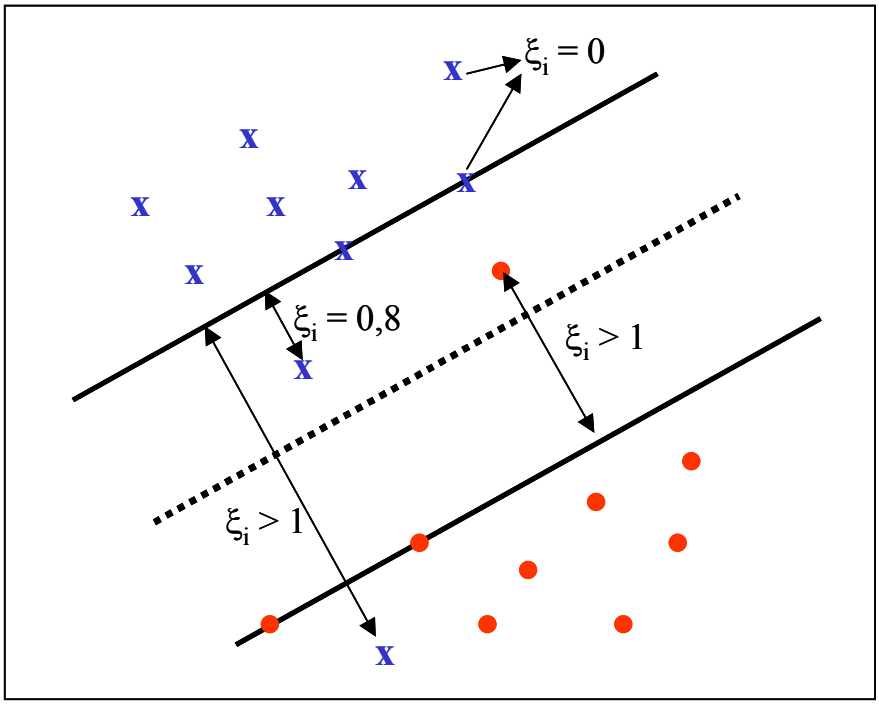


**Figura 2.3** : Exemplos de valores e situações da variável de folga $\xi$

Note que, retirando do conjunto de treinamento uma amostra em que $\xi_i > 0$, a superfície de decisão tem grande chance de mudar, mas retirando uma amostra em que $\xi_i = 0$ e o resultado da equação (2.7) é maior do que 1, a superfície de decisão permanecerá sendo a mesma.

O objetivo é encontrar o hiperplano separador em que o erro de classificação incorreta, baseado no conjunto de treinamento, é minimizado. Podemos fazer isto minimizando a função

$$\theta(\xi) = \sum_{i=1}^{N} I(\xi_i - 1) \qquad \textbf{(2.8)}$$

em relação ao vetor de pesos $w$, considerando a equação do hiperplano separador (2.7) e mais a seguinte restrição de desigualdade :

$$(w^T w) \leq \rho^{-1} = A_k \,. \qquad \textbf{(2.9)}$$

A restrição (2.9) é a condição de que os parâmetros $w$ e $b$ que definem o hiperplano minimizem o número de erros no conjunto de treinamento sobre a condição que eles pertençam ao subconjunto de elementos da estrutura $S_k = \{(w^T x) + b: (w^T w) \leq A_k\}$ determinados pela constante $A_k$.

A função indicadora $I(\xi_i - 1)$ é definida por :

$$I(\xi_i - 1) = \begin{cases} 0 & se \ \ (\xi_i - 1) \leq 0; \\ 1 & se \ \ (\xi_i - 1) > 0. \end{cases} \qquad \textbf{(2.10)}$$

Infelizmente, a minimização de $\theta(\xi)$ em relação a *w* é um problema de otimização não-convexo da classe ***NP-completo*** (não determinístico em tempo polinomial).

Para fazer este problema de otimização matematicamente tratável, aproximamos a função $\theta(\xi)$ por :

$$\theta(\xi) = \sum_{i=1}^{N} \xi_i \qquad \textbf{(2.11)}$$

restrito ao hiperplano separador (2.7) e à restrição de desigualdade (2.9).

Chamamos o hiperplano construído com base na solução deste problema de otimização de *hiperplano ótimo generalizado* ou, por simplificação, de *hiperplano ótimo.*

# Capítulo 3

# Fundamentos Básicos e Teoria do Aprendizado Estatístico

## 3.1 Introdução

Para o melhor entendimento do Capítulo 4 : "Support Vector Machines para o Problema de Classificação", serão abordados neste capítulo os seguintes tópicos:

- **Conceitos Relevantes de Otimização**: A transformação do problema de otimização primal em sua representação dual através do tratamento Lagrangeano é uma estratégia que se tornou padrão em Support Vector Machines, pois, entre outros benefícios, reduziu o problema da maldição da dimensionalidade;
- **Produto Interno Kernel**: Permite o mapeamento não-linear do conjunto de dados para um espaço característico de alta dimensão, no qual o hiperplano ótimo será construído de forma explícita;
- **Teoria do Aprendizado Estatístico**: A maximização da margem de separação na construção do hiperplano ótimo é baseada na teoria do aprendizado estatístico. Mais precisamente, emprega-se os princípios indutivos da minimização do risco estrutural, baseado no fato de que o erro de generalização é limitado pelo erro no conjunto de treinamento mais um termo que depende da dimensão VC.

Apesar destes três tópicos serem independentes entre si, todos eles serão importantes quando utilizados em conjunto na resolução do problema de como encontrar o hiperplano ótimo, descrito no Capítulo 2.

## 3.2 Conceitos Relevantes de Otimização

Neste tópico, serão descritos resultados da teoria de otimização que irão contribuir para a formulação e a tarefa de treinamento para Support Vector Machines.

Uma abordagem mais detalhada sobre este assunto pode ser encontrada nas seguintes referências : Luenberger (1984) e Bazaraa *et al*. (1993).

Começaremos com a formulação geral de um problema de minimização de uma função sujeita a determinadas restrições.

Definição 3.1 : (Problema de otimização primal) **(3.1)**

$$\begin{aligned} &\textit{Minimizar} && f(w), && w \in \Omega \subseteq R^n. \\ &\textit{Sujeito a :} && g_i(w) \le 0, && i = 1, \ldots, k; \\ & && h_j(w) = 0, && j = 1, \ldots, m. \end{aligned}$$

*onde* $f : \Omega \subseteq R^n \to R$ *é a função-objetivo;*

$g_i : \Omega \subseteq R^n \to R$, $i = 1,\ldots, k$; $h_j : \Omega \subseteq R^n \to R$, $j = 1,\ldots, m$

*são utilizadas para definir as restrições funcionais.*

A *região factível* será denotada por :

$$F = \{ w \in \Omega: g_i(w) \le 0, i = 1,\ldots, k ; h_j(w) = 0, j = 1,\ldots, m \}.$$

A solução do problema de otimização será o ponto $w^* \in F$ tal que não exista outro ponto $w \in F$ com $f(w) < f(w^*)$. Este ponto será chamado de *mínimo global*. Quando $\exists\ \varepsilon > 0$ tal que $f(w) \ge f(w^*)$, $\forall\ w \in \Omega$ com $\|w - w^*\| < \varepsilon$, o ponto $w^*$ será chamado de *mínimo local*. Quando a função $f$ é convexa, um mínimo local $w^*$ é também mínimo global.

Um problema de otimização em que a função-objetivo é quadrática, enquanto que as restrições são todas lineares, é chamado de problema de otimização *quadrático*, e se a função-objetivo e todas as restrições são convexas o problema é chamado de *convexo*.

No problema de treinamento para Support Vector Machines, as restrições serão lineares e a função-objetivo será convexa e quadrática. Com isso o problema de otimização também será *convexo e quadrático.*

Para resolver este tipo de problema de otimização, faz-se necessário apresentar a Teoria Lagrangeana e suas extensões.

A Teoria Lagrangeana foi desenvolvida por Lagrange, em 1797, apenas com restrições de igualdade, generalizando os resultados de Fermat de 1629. Em 1951, Kuhn e Tucker estenderam o método e permitiram restrições de desigualdade. Este novo método conduz às conhecidas condições de Kuhn-Tucker.

Teorema 3.2 : (Fermat)

*A condição necessária para $w^*$ ser um mínimo de f(w), $f \in C^1$, onde $C^1$ é o conjunto das funções contínuas em* $\Omega$, *é que $\partial f(w^*)/\partial w = 0$. Esta condição, junto com a de convexidade de f , é também uma condição suficiente.*

Em problemas restritos, é necessário uma função que incorpore tanto a função-objetivo quanto as restrições, e que sua estacionariedade defina a solução. Esta função é a Lagrangeana ($L : R^n \times R^m \rightarrow R$) que é definida como uma combinação linear da função-objetivo mais cada restrição associada ao seu respectivo *multiplicador de Lagrange* $\beta_j$ ,

$$L(w,\beta) = f(w) + \sum_{j=1}^{m} \beta_j h_j(w) \cdot$$

Teorema 3.3 : (Lagrange)

*A condição necessária para o ponto $w^*$ ser um mínimo de f(w), sujeito a $h_j(w) = 0$, $j = 1, \ldots, m$ com f, $h_j \in C^1$, $j = 1, \ldots, m$ é :*

$$\frac{\partial L}{\partial w}(w^*, \beta^*) = 0; \qquad \frac{\partial L}{\partial \beta}(w^*, \beta^*) = 0 \cdot$$

A condição acima é também suficiente se $L(w,\beta^*)$ é uma função convexa em $w$.

Considere agora o caso mais geral de um problema de otimização, quando existem restrições tanto de igualdade como desigualdade, como o problema (3.1). Assim, a função Lagrangeana generalizada ($L : R^n \times R^k \times R^m \rightarrow R$) é dada por :

$$L(w,\alpha,\beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{j=1}^{m} \beta_j h_j(w) \cdot$$

É conveniente agora definir o *problema dual* Lagrangeano e em seguida três teoremas sobre dualidade.

Definição 3.4 :

*O problema dual Lagrangeano referente ao problema primal (3.1) é o seguinte :*

*Maximizar* $\theta(\alpha, \beta)$.

*Sujeito a :* $\alpha \geq 0$.

*onde* $\theta(\alpha,\beta) = inf_{w \in \Omega} L(w,\alpha,\beta)$.

Teorema 3.5 : (Teorema fraco da dualidade)

*Sendo* $\Omega$ *a região factível do problema primal (3.1) e* $(\alpha^*,\beta^*)$ *a solução factível do problema dual da Definição 3.4, então* $f(w) \geq \theta(\alpha^*,\beta^*)$ *para* $w \in \Omega$.

Este teorema afirma que o valor da solução dual é limitada superiormente pelo valor da solução primal.

Se $f(w^*) = \theta(\alpha^*,\beta^*)$, onde as restrições do problema primal e dual são satisfeitas, então $w^*$ e $(\alpha^*,\beta^*)$ resolvem o problema primal e dual respectivamente. Neste caso, $\alpha_i^* g_i(w^*) = 0$, para $i = 1, \ldots, k$.

Resolvendo e comparando as soluções dos problemas primal e dual, espera-se que a diferença entre as duas soluções no ponto ótimo seja zero. Contudo esta expectativa não é sempre atendida e a diferença entre os valores do problema primal e dual é chamado de *gap de dualidade*.

Um caminho para detectar a ausência do gap de dualidade é a presença de um *ponto de sela* $(w^*,\alpha^*,\beta^*)$ satisfazendo : $L(w^*,\alpha,\beta) \leq L(w^*,\alpha^*,\beta^*) \leq L(w,\alpha^*,\beta^*)$, com $w \in \Omega$, $\alpha \in R^k$ e $\beta \in R^m$.

Teorema 3.6 :

*A tripla* $(w^*,\alpha^*,\beta^*)$ *é um ponto de sela da função Lagrangeana para o problema primal se, e somente se, seus componentes são a solução ótima dos problemas primal e dual e não há gap de dualidade. Assim os dois problemas têm custo dado por :*

$$f(w^*) = \theta(\alpha^*,\beta^*).$$

Teorema 3.7 : (Teorema forte da dualidade)

*Dado o problema de otimização (3.1) com o domínio convexo e $g_i$ , $i = 1,\ldots, k$ e $h_j$ , $j = 1,\ldots, m$ sendo funções afins, ou seja, do tipo $h(w) = Aw-b$, o gap de dualidade é igual a zero.*

Teorema 3.8 : (Condições de Kuhn-Tucker)

*Dado o problema de otimização (3.1) com o domínio convexo , $f \in C^1$ convexa, $g_i$ , $i = 1,\ldots, k$ e $h_j$, $j = 1,\ldots, m$ sendo funções afins, a condição necessária e suficiente para o ponto $w^*$ ser o ótimo, é a existência de $\alpha^*$, $\beta^*$ satisfazendo :*

$$\frac{\partial L}{\partial w}(w^*, \alpha^*, \beta^*) = 0;$$

$$\frac{\partial L}{\partial \beta}(w^*, \alpha^*, \beta^*) = 0;$$

$$\alpha_i^* g_i(w^*) = 0 , \quad i = 1, \ldots, k;$$

$$g_i(w^*) \leq 0 \quad , \quad i = 1, \ldots, k;$$

$$\alpha_i^* \geq 0 \quad , \quad i = 1, \ldots, k.$$

O tratamento Lagrangeano para o problema de otimização convexo fornece a alternativa da descrição dual, que freqüentemente tende a ser mais fácil de ser resolvida do que o descrição primal, a qual apresenta restrições de desigualdade difíceis de serem resolvidas. Esta estratégia tornou-se padrão na teoria de Support Vector Machines, porque a representação dual permitirá trabalhar em um espaço de alta dimensão, devido ao número de parâmetros ajustados não depender do número de atributos sendo utilizados (dimensão dos dados de entrada). Veremos também que as amostras associadas aos multiplicadores de Langrange maiores do que zero (restrições ativas) serão denominadas de vetores-suporte.

## 3.3 Produto Interno Kernel

O nome kernel é derivado da teoria do operador integral. A teoria de kernels é antiga, o teorema de Mercer foi escrito em 1908. Porém, o conceito de produto interno kernel foi primeiro utilizado por Aizerman *et al*. (1964a, 1964b) na formulação do método de funções potenciais, que representaram o precursor dos modelos de regressão com funções de base radial.

Aproximadamente no mesmo período, em 1965, Vapnik e Chervonienkis (Vapnik & Chervonienkis, 1974) desenvolveram o conceito do hiperplano ótimo (Capítulo 2). A combinação do uso destes dois poderosos conceitos, feita por Vapnik, originou a formulação de Support Vector Machines.

### 3.3.1 Espaço característico

Uma estratégia de pré-processamento em algoritmos de aprendizado como os que serão considerados neste estudo, envolve a mudança de representação dos dados na forma:

$$x = (x_1,...,x_m) \mapsto (\phi_1(x),...,\phi_M(x)), \quad \text{onde } M >> m.$$

Este passo é equivalente ao mapeamento não-linear do espaço dos dados de entrada $X$ em um novo espaço, $F_C = \{ \phi(x) \mid x \in X \}$, chamado de *espaço característico*. Iremos denotar o vetor $\{\phi_j(x)\}_{1 \le j \le M}$ como o conjunto de transformações não-lineares definidas a priori. As medidas originais de representação dos dados serão chamadas de *atributos* e as medidas no espaço $F_C$ serão chamadas de *características*.

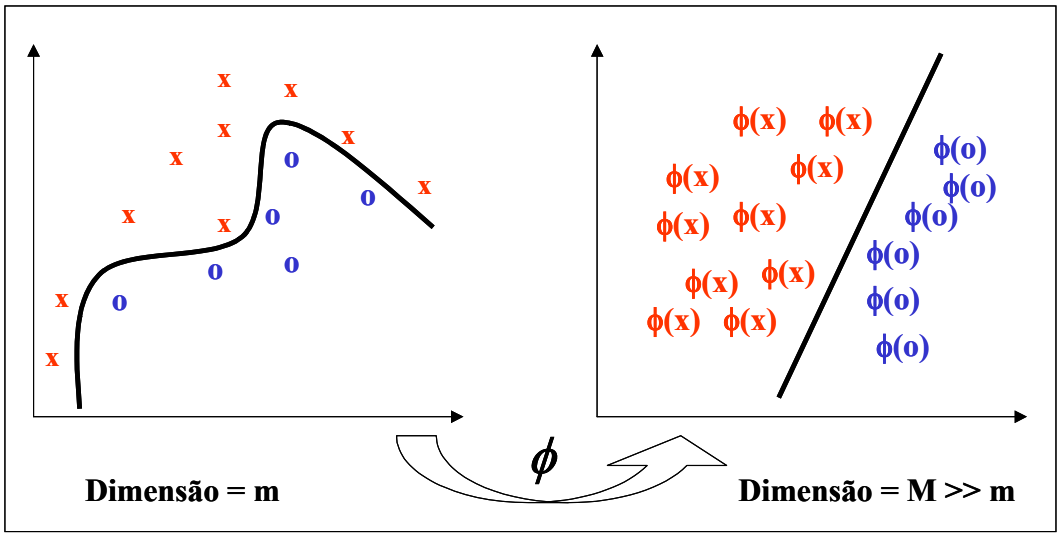


**Figura 3.1**: Exemplo de mapeamento para o espaço característico, onde é possível a separação linear das duas classes, originalmente não separáveis, por um hiperplano. Foi utilizado $M$=2 por questões de visualização, pois $M >> m$.

A Figura 3.1 mostra um exemplo de mapeamento não-linear dos dados de entrada pertencentes a um espaço bidimensional para um espaço característico de mesma dimensão (na prática este mapeamento é feito para um espaço de dimensão muito maior do que a original). A

idéia é que no primeiro espaço os dados não podem ser separáveis por um hiperplano, mas no segundo espaço isto seria possível.

Isto significa que se pode construir um algoritmo em dois passos: primeiro um mapeamento não-linear, escolhido a priori, mapeia os dados de entrada para um espaço característico $F_C$, e depois um hiperplano é utilizado como superfície de decisão para classificar os dados neste novo espaço :

$$\sum_{j=1}^{M} w_j \phi_j(x) + b = 0,$$

onde $\{w_j\}_{1 \le j \le M}$ é o vetor de pesos e $b$ o intercepto. Pode-se simplificar este hiperplano escrevendo :

$$\sum_{j=0}^{M} w_j \phi_j(x) = w^T \phi(x) = 0, \tag{3.2}$$

onde $\phi_0(x) = 1$, para todo $x$, faz o papel de intercepto.

O vetor $\phi(x)$ representa a imagem induzida no espaço característico a partir do vetor de entrada $x$.

Será visto em detalhes, no Capítulo 4, que o vetor de pesos poderá ser substituído por:

$$w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i), \tag{3.3}$$

onde $\phi(x_i)$ corresponde à imagem induzida do $i$-ésimo padrão de entrada.

Substituindo a equação (3.3) na (3.2), é possível definir a superfície de decisão calculada no espaço característico como :

$$\sum_{i=1}^{N} \alpha_i y_i \phi^T(x_i) \phi(x) = 0. \tag{3.4}$$

O termo $\phi^T(x_i)\phi(x)$ representa o produto interno de dois vetores induzidos no espaço característico pelo vetor de entrada $x$ e pelo $i$-ésimo padrão $x_i$ .

Agora é possível introduzir o *produto interno kernel* denotado por $K(x,x_i)$ e definido como segue:

$$K(x, x_i) = \phi^T(x_i)\phi(x) = \sum_{j=0}^{M} \phi_j(x_i)\phi_j(x), \qquad \text{para } i = 1, 2, \ldots, N. \qquad \textbf{(3.5)}$$

O produto interno kernel apresenta características importantes por ser uma função simétrica em seus argumentos : $K(x, x_i) = K(x_i, x)$.

O mais importante é que se pode usar o produto interno kernel para construir o hiperplano ótimo no espaço característico sem ter que considerar este espaço de forma explícita. Isto pode ser visto substituindo a equação (3.5) na (3.4):

$$\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) = 0. \qquad \textbf{(3.6)}$$

### 3.3.2 Teorema de Mercer

A expansão da equação (3.5) para o produto interno kernel $K(x,x_i)$ é um caso especial do teorema de Mercer, que surgiu no contexto de análise funcional. Este teorema pode ser formulado como segue: (Mercer, 1909 ; Courant & Hilbert, 1970):

*Seja K(x,x') uma função kernel contínua e simétrica definida na região fechada [a, b]×[a, b]. A função kernel K(x,x') pode ser expandida em uma série :*

$$K(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(x')$$

*com coeficientes $\lambda_i$>0. Para esta expansão ser válida e convergir absolutamente e uniformemente, é uma condição necessária e suficiente que :*

$$\int_a^b \int_a^b K(x, x')\psi(x)\psi(x')dxdx' \geq 0$$

*para todo $\psi$(.) no qual*

$$\int_a^b \psi^2(x)dx < \infty .$$

As funções $\psi(x)$ são chamadas de auto-funções (*eigenfunctions*) da expansão e os coeficientes $\lambda_i$ são chamados de autovalores. O fato de todos os autovalores serem positivos significa que a função kernel é *definida positiva*.

Sobre o teorema de Mercer pode-se fazer as seguintes observações:

- Para $\lambda_i \neq 1$, a *i*-ésima imagem $\sqrt{\lambda_i}\phi_i(x)$ induzida no espaço característico pelo vetor de entrada *x* é uma auto-função da expansão;
- Na teoria, a dimensionalidade do espaço característico (o número de autovalores e as auto-funções) pode ser infinita.

O teorema de Mercer nos diz simplesmente quando uma função candidata a kernel é de fato um produto interno kernel em algum espaço determinado e portanto admissível para ser utilizada no treinamento de Support Vector Machines (SVM). Porém este teorema não indica como obter as funções $\phi_i(x)$.

### 3.3.3 Tipos de produto interno kernel mais utilizados em SVM

O principal requisito para o produto interno kernel ser utilizado em SVM é que ele satisfaça o teorema de Mercer. Assim, apresentamos os três tipos mais comuns de produtos internos kernel utilizados em SVM:

- *Função de Base Radial (RBF)* :

$$K(x,x') = exp(-||x' - x||^2 / 2\sigma^2) , \quad \textbf{(3.7)}$$

onde o parâmetro $\sigma^2$ (interpretado como a variância da RBF) é especificado a priori pelo usuário;

- *Função Polinomial* :

$$K(x,x') = (x'^T x + 1)^d , \quad \textbf{(3.8)}$$

onde a parâmetro *d* (grau do polinômio) é especificado a priori pelo usuário;

- *Perceptron* :

$$K(x,x') = tanh(\beta_0 x'^T x + \beta_1) , \quad \textbf{(3.9)}$$

onde $\beta_0$ e $\beta_1$ são os parâmetros ajustados pelo usuário, sendo que apenas para determinados valores destes parâmetros o teorema de Mercer é satisfeito.

A seguir, são apresentados dois exemplos em que as funções de decisão no espaço de entrada são não-lineares como mostrado na Figura 3.2. Nestes exemplos, o problema original é mapeado não-linearmente para um espaço característico de alta dimensão através de um produto interno kernel, sendo que o produto interno kernel que possivelmente apresenta o melhor desempenho é o da função polinomial (3.8) com parâmetro $d = 2$. Neste novo espaço, um hiperplano ótimo será construído.

Assim, o espaço de entrada bi-dimensional será mapeado para o espaço característico de dimensão igual a 5 através da função polinomial de grau 2, como demonstrado a seguir:

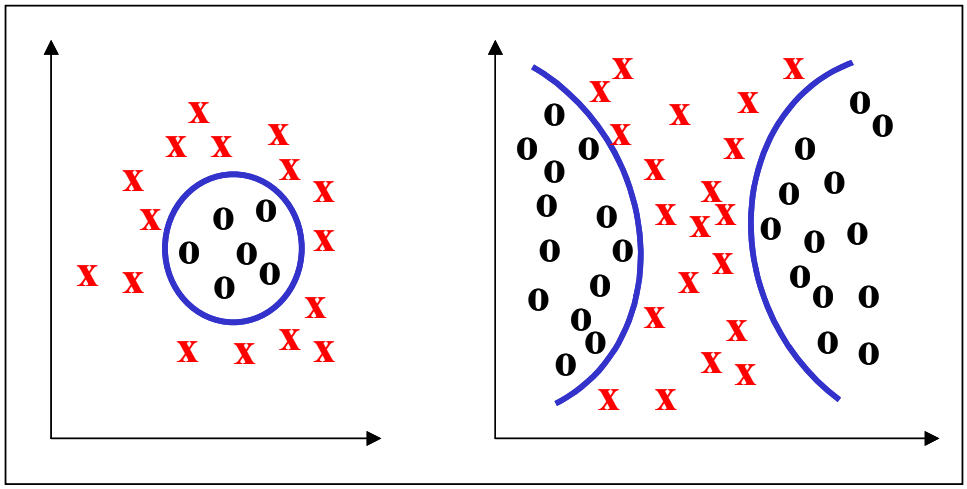

**Figura 3.2** : Dois exemplos em que a função de decisão no espaço de entrada é não-linear e possivelmente um produto interno kernel do tipo polinomial com grau 2 poderá mapear este espaço para um espaço característico de alta dimensão, onde o hiperplano ótimo será construído.

Sendo $x^T = [x_1, x_2]$ , $x_i^T = [x_{i1}, x_{i2}]$ e $K(x, x_i) = \phi^T(x)\phi(x_i) = (x_i^T x + 1)^2$ , é possível expressar $K(x,x_i)$ em termos de :

$$K(x, x_i) = (1 + x^T x_i) \times (1 + x^T x_i) = 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2} .$$

A imagem do vetor de entrada $x$, induzido no espaço característico, é então deduzida ser :

$$\phi^T(x) = \left[ 1, x_1^2, \sqrt{2}x_1 x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2 \right] \quad e \quad \phi^T(x_i) = \left[ 1, x_{i1}^2, \sqrt{2}x_{i1} x_{i2}, x_{i2}^2, \sqrt{2}xi_1, \sqrt{2}x_{i2} \right].$$

Portanto, o espaço de entrada bi-dimensional, quando aplicado um produto interno kernel do tipo polinomial de grau 2, é mapeado para um espaço característico de dimensão 5.

## 3.4 Teoria do Aprendizado Estatístico

Desenvolvida por Vapnik desde 1971, a Teoria do Aprendizado Estatístico é conhecida também como a Teoria de Vapnik-Chervonenkis (VC).

Ela é considerada a melhor teoria atualmente disponível para a estimação estatística de amostras finitas, estudo da dependência funcional e do aprendizado preditivo (Vapnik, 1998).

Esta teoria define rigorosamente todos os conceitos relevantes e produz demonstrações matemáticas para todos os resultados importantes, e será descrita nos tópicos a seguir.

### 3.4.1 Condições para a Consistência e Convergência da Minimização do Risco Empírico

Considere $( x_i , y_i )_{1 \le i \le N}$, $x_i \in R^m$, $y_i \in R$, denotando os pares de dados de entrada e saída de treinamento, independentes e identicamente distribuídos, gerados de acordo com alguma função densidade de probabilidade conjunta desconhecida $F_{x,y}(x,y)$.

O problema do treinamento supervisionado é encontrar uma função particular $F(x,w) = \hat{y}$, sendo $w$ o vetor de parâmetros ou pesos, de tal modo que $\hat{y}_i$ aproxima a resposta desejada $y_i$ de acordo com algum critério estatístico. A viabilidade do treinamento supervisionado depende das amostras de treinamento conterem informação suficiente para a construção de um algoritmo capaz de gerar um bom desempenho de generalização.

A medida de perda ou discrepância entre a resposta desejada $y_i$ e a resposta obtida $\hat{y} = F(x,w)$ é a *função de erro.* Um ponto forte da teoria do aprendizado estatístico é a de não depender da forma da função de erro.

Como exemplo, utilizaremos o caso do problema de regressão, onde é comum o uso da função de erro quadrática dada por :

$$L( y , F(x,w) ) = ( y - F(x,w) )^2.$$

O objetivo do aprendizado supervisionado é encontrar uma função $F(x,w)$ dentre uma classe de funções $\{F(x,w), w \in W\}$ que minimize a *função-risco*:

$$R(w) = \int L( y , F(x,w) ) \, dF_{x,y}(x,y) . \qquad \textbf{(3.10)}$$

Dado o conjunto de treinamento $( x_i , y_i )_{1 \le i \le N}$, e utilizando os princípios indutivos da minimização do risco empírico, a minimização da função-risco desconhecida é substituída pela minimização da *função-risco empírico* conhecida:

$$R_{emp}(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(x_i, w)). \qquad \textbf{(3.11)}$$

A idéia básica da minimização do risco empírico é trabalhar com a função $R_{emp}(w)$ que difere de $R(w)$ em dois aspectos : não depende da função densidade de probabilidade conjunta desconhecida $F_{x,y}(x,y)$ ; pode ser minimizada com respeito ao vetor de pesos $w$.

Seja $w_{emp}$ e $F(x,w_{emp})$ o vetor de pesos e o correspondente mapeamento que minimiza a função-risco empírico (3.11), e similarmente $w_o$ e $F(x,w_o)$ o vetor de pesos e o correspondente mapeamento que minimiza a função-risco (3.10), ambos os vetores pertencentes a $W$. O problema apresentado é sobre quais condições o mapeamento aproximado $F(x,w_{emp})$ está "próximo" do mapeamento desejado $F(x,w_o)$, medidos pela diferença entre $R_{emp}(w)$ e $R(w)$.

Para algum $w = w^*$ fixo, a função-risco $R(w^*)$ determina a esperança matemática da variável aleatória definida por :

$$Z_{w^*} = L( y , F(x,w^*) ).$$

Por outro lado, a função-risco empírico $R_{emp}(w^*)$ é a média aritmética (empírica) da variável aleatória $Z_{w^*}$.

De acordo com a *lei dos grandes números* (James, 1981), que constitui um dos mais importantes resultados da teoria de probabilidade, nos casos gerais em que o tamanho do conjunto de treinamento $N$ cresce para infinito, a média empírica da variável aleatória $Z_{w^*}$ converge para seu valor esperado. Isto proporciona uma justificativa teórica para o uso da função-risco empírica $R_{emp}(w)$ no lugar da função-risco $R(w)$.

Definição (Vapnik, 1995):

*Pode-se dizer que o Princípio da Minimização do Risco Empírico é consistente para um conjunto de funções* $L( y , F(x,w) )$, $w \in W$ *e para a função densidade de probabilidade* $F_x(x)$ *se as seguintes duas seqüências convergem em probabilidade para o mesmo limite*, ver Figura 3.3:

$$R(w_N) \xrightarrow[N\to\infty]{Prob} \inf_{w\in W} R(w);$$

$$R_{emp}(w_N) \xrightarrow[N\to\infty]{Prob} \inf_{w\in W} R(w).$$

Em outras palavras o princípio da Minimização do Risco Empírico é consistente se ele fornece uma seqüência de funções $L( y , F(x,w_N) )$, com $N = 1,2, ...$ para o qual o risco esperado e o risco empírico convergem para o mínimo valor possível do risco.

Porém, somente porque a média empírica de $Z_{w^*}$ converge para seu valor esperado, não há razão para esperar que o vetor de pesos $w_{emp}$, que minimiza $R_{emp}(w)$, minimizará também $R(w)$.

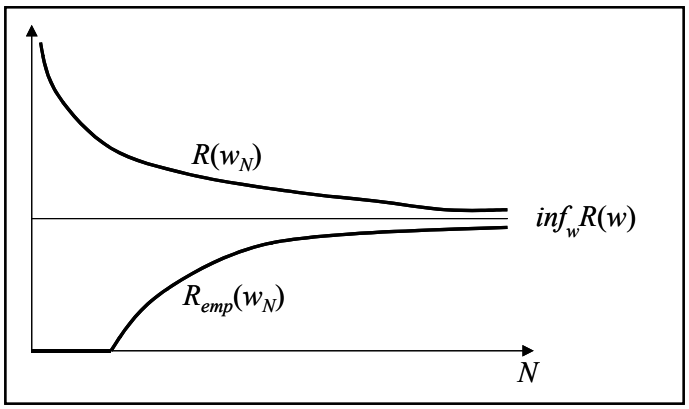


**Figura 3.3** : Consistência da Minimização do Risco Empírico. O processo de aprendizado é consistente se o risco esperado $R(w)$ e o risco empírico $R_{emp}(w)$ convergem em probabilidade para o mínimo valor possível do risco, $\inf_w R(w)$. Para o problema de classificação, o risco empírico corresponde à freqüência do erro de classificação no conjunto de dados de treinamento, e o risco esperado é a probabilidade de erro de classificação calculada sobre a função densidade de probabilidade conjunta desconhecida $F_{x,y}(x,y)$. Para um conjunto específico de dados de treinamento, podemos esperar que $R_{emp}(w^*) < R(w^*)$ porque o processo de aprendizagem muitas vezes estima uma função que minimiza o risco empírico mas não necessariamente o risco verdadeiro, porém quando o número de amostras do conjunto de dados de treinamento cresce para infinito, podemos esperar que $R_{emp}(w^*)$ seja próximo de $R(w^*)$. Para pequeno número de amostras o $R_{emp}(w) = 0$, pois geralmente o processo de aprendizado consegue aproximar a resposta desejada sem erros na amostra de treinamento.

Esta exigência será satisfeita da seguinte maneira: se $R_{emp}(w)$ aproxima $R(w)$ uniformemente em $w$ com alguma precisão $\varepsilon$, então o mínimo de $R_{emp}(w)$ diverge do mínimo de $R(w)$ por uma quantidade que não excede $2\varepsilon$. Formalmente, isto significa que é possível impor uma condição estrita de que, para qualquer $w \in W$ e $\varepsilon > 0$, a relação probabilística

$$P\left( \sup_w \left| R(w) - R_{emp}(w) \right| > \varepsilon \right) \to 0 \qquad \textit{quando} \qquad N \to \infty \tag{3.12}$$

é válida (Vapnik, 1982). Quando (3.12) é satisfeita, é dito que ocorre a *convergência uniforme do vetor de pesos w do risco empírico médio para o seu valor esperado.* Esta convergência é uma condição necessária e suficiente para o *princípio da minimização do risco empírico* (Vapnik, 1982, 1998).

Uma interpretação deste princípio é que, antes de empregar um algoritmo de aprendizado, todas as funções de aproximação são igualmente prováveis (plausíveis). Com a progressão da execução do algoritmo de aprendizado, a plausibilidade das funções de aproximação $F(x,w)$ que

são consistentes com as amostras de treinamento $( x_i , y_i )_{1 \le i \le N}$ são aumentadas. Conforme o tamanho do conjunto de treinamento $N$ cresce, e o espaço de entrada é por meio disso densamente povoado, o ponto mínimo da função-risco empírico $R_{emp}(w)$ converge em probabilidade para o ponto mínimo da verdadeira função-risco $R(w)$.

### 3.4.2 Dimensão VC

A teoria da convergência uniforme da função-risco empírico $R_{emp}(w)$ para a real função-risco $R(w)$ inclui limitantes na razão de convergência, que são baseados em um importante parâmetro denominado de *dimensão Vapnik e Chervonenkis*, ou simplesmente *dimensão VC*, assim denominada em honra a seus criadores, Vapnik e Chervonenkis (1971). A dimensão VC é uma medida da capacidade ou força de expressão de uma família de funções classificadoras obtidas por meio de um algoritmo de aprendizagem.

Considere um problema de classificação de padrões binários, $y \in \{0,1\}$. Seja $\mathscr{F}$ um conjunto de implementações dicotômicas (funções classificadoras binárias) por um algoritmo de aprendizado, isto é:

$$\mathscr{F} = \{ F(x,w):\ w \in W ,\ F: R^m \times \mathrm{W} \rightarrow \{0,1\} \}.$$

Seja $\mathscr{L}$ o conjunto de $N$ pontos em um espaço de entrada $\mathscr{R}$ m-dimensional, isto é :

$$\mathscr{L} = \{ x_i \in \mathscr{R} ;\ i = 1, 2, \ldots , N \}.$$

A implementação dicotômica de um algoritmo de aprendizagem, que divide $\mathscr{L}$ em dois subconjuntos disjuntos $\mathscr{L}_0$ e $\mathscr{L}_1$, pode ser escrita da seguinte maneira:

$$F(x,w) = \{ 0 \text{ para } x \in \mathscr{L}_0 \ ; 1 \text{ para } x \in \mathscr{L}_1 \}.$$

Seja $\Delta_{\mathscr{F}}(\mathscr{L})$ o número de implementações dicotômicas distintas pelo algoritmo de aprendizado, e $\Delta_{\mathscr{F}}(l)$ o máximo de $\Delta_{\mathscr{F}}(\mathscr{L})$ sobre todo $\mathscr{L}$ com $|\mathscr{L}| = l$, onde a cardinalidade $|\mathscr{L}|$ é o número máximo de elementos de $\mathscr{L}$. Pode-se dizer que $\mathscr{L}$ é particionado por $\mathscr{F}$ se $\Delta_{\mathscr{F}}(\mathscr{L}) = 2^{|\mathscr{L}|}$, isto é, se todas as divisões binárias de $\mathscr{L}$ podem ser produzidas pelas funções em $\mathscr{F}$. Refere-se a $\Delta_{\mathscr{F}}(l)$ como a *função de crescimento*.

A Figura 3.4 ilustra, em um espaço bi-dimensional $\mathscr{R}$ o conjunto $\mathscr{L}$ consistindo de três pontos $x_1$, $x_2$ e $x_3$, e todas as partições binárias produzidas por funções em $\mathscr{F}$.

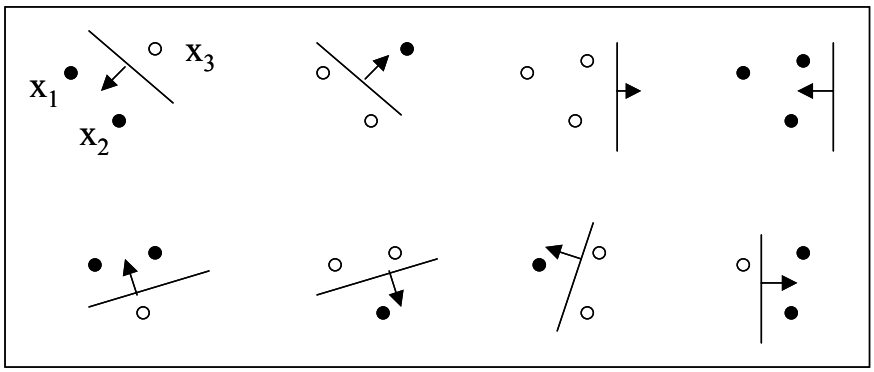


**Figura 3.4** : Três pontos no $R^2$ particionados por retas orientadas

Com o conjunto $\mathscr{X}$ consistindo de três pontos e a cardinalidade dada por $|\mathscr{X}| = 3$, conseqüentemente $\Delta_{\mathscr{F}}(\mathscr{X}) = 2^3 = 8$ , como podemos observar na Figura 3.4.

É possível definir formalmente a dimensão VC como (Vapnik e Chernovenkis, 1971):

*A dimensão VC de um conjunto de funções dicotômicas $\mathscr{F}$ é a cardinalidade do maior conjunto $\mathscr{X}$ que é particionado por $\mathscr{F}$.*

Em outras palavras, a dimensão VC de $\mathscr{F}$, denotada por VCdim($\mathscr{F}$), é o maior $N$ tal que $\Delta_{\mathscr{F}}(N) = 2^N$. Em termos mais familiares, a dimensão VC de um conjunto de funções de classificação $\{ F(x,w): w \in W \}$ é o número máximo de amostras de treinamento que podem ser classificadas sem erro, para todas as possíveis atribuições de rótulos às amostras.

Consideremos o exemplo da regra de decisão do hiperplano ótimo, descrita no Capítulo 2, em um espaço de entrada $\mathscr{R}$ $m$-dimensional:

$$\mathscr{F}: y = \varphi[(w^{\mathrm{T}}x) + b] \text{ ,} \qquad \textbf{(3.13)}$$

onde $y = 1$ para $[(w^{\mathrm{T}}x) + b] \geq 0$, e $y = 0$ caso contrário.

A dimensão VC para o hiperplano ótimo é dada por :

$$\mathrm{VCdim}(\mathscr{F}) = m + 1 \text{ .} \qquad \textbf{(3.14)}$$

Para demonstrar este resultado, considere a situação descrita na Figura 3.4, onde $m = 2$ e temos 3 pontos. Quando se busca reproduzir a situação anterior com 4 pontos, não é possível representar todas as possibilidades de separação apenas pelas retas orientadas. A Figura 3.5 apresenta um desses casos.

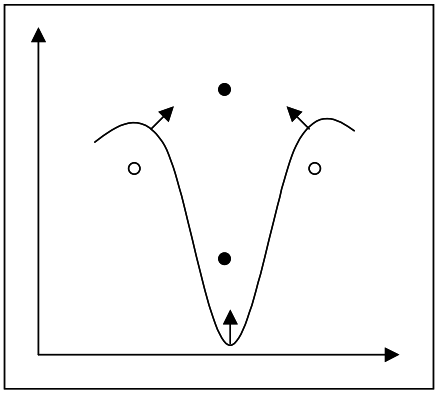

**Figura 3.5** : Quatro pontos no $R^2$ divididos em duas classes e que não podem ser separados por uma reta orientada.

Portanto, a dimensão VC para o hiperplano ótimo com $m = 2$ é igual a 3, de acordo com a fórmula (3.14) e comprovado graficamente pelas Figuras 3.4 e 3.5. Nota-se que a dimensão VC para o hiperplano ótimo é igual ao número de parâmetros do hiperplano (3.13).

Com a dimensão VC fornecendo uma medida de capacidade de um conjunto de funções classificadoras, pode-se esperar que o algoritmo de aprendizagem com muitos parâmetros a serem ajustados terá uma grande dimensão VC, por outro lado um algoritmo de aprendizagem com poucos parâmetros terá uma pequena dimensão VC. A seguir, é apresentado um contra-exemplo para esta afirmação.

Considere a família de funções indicadoras, com apenas um parâmetro, definida como:

$$f(x,a) = \text{sgn}(\text{sen}(ax)),\ a \in R\ ,$$

onde sgn(.) é a função sinal. A exigência é encontrar $N$ pontos que podem ser particionados. Esta exigência é satisfeita pelo conjunto de funções $f(x,a)$ pela escolha de $x_i = 10^{-i}$, para $i = 1, 2, ..., N$. Para separar estes pontos em duas classes, determinadas pela seqüência $y_1, y_2, ..., y_N$ , com $y_i \in \{-1,+1\}$, é suficiente escolher o parâmetro $a$ de acordo com a fórmula:

$$a = \pi\left(1 + \sum_{i=1}^{N} \frac{(1 - y_i)10^i}{2}\right) .$$

Como $N$ pode ser qualquer, conclui-se que a dimensão VC da família de funções indicadoras $f(x,a)$ com um único parâmetro $a$ é infinita.

O conceito de dimensão VC nos proporciona a conclusão de que o número de amostras necessárias para a aprendizagem de uma classe de funções de interesse é proporcional à dimensão

VC desta classe. Conseqüentemente, a estimativa da dimensão VC é um conceito fundamental. Porém, na maioria dos casos é muito difícil calcular a dimensão VC na forma analítica.

### 3.4.3 Limitantes para a Generalização

Nesta seção, será abordado como se chega aos limitantes (limite superior de confiança) para o erro de generalização em função da razão da convergência uniforme do princípio da minimização do risco empírico. Como alguns passos das demonstrações para o cálculo dos limitantes não são triviais, optou-se por apenas apresentar os resultados, sem demonstrar todos os passos. A referência para uma leitura mais detalhada está contida em Vapnik (1998).

Considere o exemplo da regra de decisão através do hiperplano ótimo (equação (3.13)), para a classificação de padrões binários, no qual a resposta desejada $y \in \{0,1\}$. A correspondente função de erro tem somente dois valores:

$$L(y, F(x,w)) = \begin{cases} 0 & se \quad F(x,w) = y; \\ 1 & caso \ \ contrário. \end{cases}$$

A função-risco $R(w)$ (equação (3.10)) para o hiperplano ótimo pode ser interpretada como a probabilidade de classificação incorreta calculada sobre a função densidade de probabilidade conjunta desconhecida $F_{x,y}(x,y)$, denotada por $P(w)$, e a função-risco empírico $R_{emp}(w)$ (equação (3.11)) é a freqüência do erro de treinamento, denotado por $v(w)$.

De acordo com a *lei fraca dos grandes números* (James, 1981), a freqüência empírica da ocorrência de um evento converge quase que certamente para a verdadeira probabilidade do evento quando o número de amostras cresce para infinito. Assim, dentro do contexto desta seção, para um vetor de pesos $w \in W$ e uma precisão $\varepsilon > 0$, a relação probabilística dada a seguir é válida:

$$P\left( \sup_{w} \left| P(w) - v(w) \right| \ > \varepsilon \right) \to 0 \qquad quando \qquad N \to \infty. \tag{3.15}$$

O conceito de dimensão VC proporciona um limitante para a razão da convergência uniforme. Especificamente, para um conjunto de funções de classificação com dimensão VC igual a $h$, a seguinte inequação é válida (Vapnik, 1982, 1998):

$$P\left( \sup_{w} \left| P(w) - v(w) \right| \ > \varepsilon \right) < \left( \frac{2eN}{h} \right)^{h} \exp(-\varepsilon^2 N), \tag{3.16}$$

onde $N$ é o número de amostras e "$e$" é a base do logaritmo natural.

Deve-se deixar o lado direito da inequação (3.16) pequeno para grandes valores de $N$ a fim de conseguir uma convergência uniforme. O fator $\exp(-\varepsilon^2 N)$ é muito útil nesta relação, pois decai exponencialmente com o aumento de $N$. O outro fator $(2eN/h)^h$ representa o limitante da função de crescimento $\Delta_{\mathscr{F}}(l)$ para uma família de funções $\mathscr{F} = \{ F(x,w): w \in W \}$ e para $1 \le h \le l$, como obtido pelo *lema de Sauer* (Sauer, 1972). Desde que esta função não cresça muito rápido, o lado direito da inequação irá para zero quando $N$ for para infinito; esta condição é satisfeita se a dimensão VC $h$ é finita.

Portanto a dimensão VC finita é uma condição necessária e suficiente para a convergência uniforme do princípio da minimização do risco empírico.

Seja $\alpha$ a probabilidade de ocorrência do evento:

$$P\left( \sup_{w} \left| P(w) - v(w) \right| \ \ge \varepsilon \right) < \alpha. \qquad \textbf{(3.17)}$$

Usando o limitante descrito pelo lado direito da inequação (3.16) e com probabilidade de ocorrência do evento igual a $\alpha$, pode-se definir o limitante superior de confiança $\varepsilon_o(N,h,\alpha)$ como (Vapnik, 1992):

$$\varepsilon_0(N,h,\alpha) = \sqrt{\frac{h}{N}\left[ \log\left( \frac{2N}{h} \right) + 1 \right] - \frac{1}{N}\log\alpha} \quad . \qquad \textbf{(3.18)}$$

Então, com probabilidade 1-$\alpha$, podemos afirmar que, para qualquer vetor de pesos $w \in W$, a seguinte inequação é válida:

$$P(w) < v(w) + \varepsilon_o(N,h,\alpha) \ . \qquad \textbf{(3.19)}$$

O limitante da inequação (3.16) com o valor de $\varepsilon = \varepsilon_o(N,h,\alpha)$ é alcançado para o pior caso onde $P(w) = 1/2$. Para pequenos valores de $P(w)$, situação mais freqüente na prática, o limitante é obtido considerando uma modificação da inequação (3.16) (Vapnik 1982, 1998) :

$$P\left( \sup_{w} \left| \frac{P(w) - v(w)}{\sqrt{P(w)}} \right| > \varepsilon \right) < \left( \frac{2eN}{h} \right)^h \exp\left( -\frac{\varepsilon^2 N}{4} \right) \ . \qquad \textbf{(3.20)}$$

Assim, com probabilidade 1-$\alpha$ , e qualquer vetor de pesos $w \in W$, temos (Vapnik 1982, 1998):

$$P(w) \le v(w) + \varepsilon_l(N,h,\alpha,v) \ , \qquad \textbf{(3.21)}$$

onde $\varepsilon_l(N,h,\alpha,v)$ é o novo limitante superior de confiança definido como:

$$\varepsilon_1(N,h,\alpha,v) = 2\varepsilon_0^2(N,h,\alpha)\left(1+\sqrt{1+\frac{v(w)}{\varepsilon_0^2(N,h,\alpha)}}\right). \qquad \textbf{(3.22)}$$

Este segundo intervalo de confiança difere do primeiro principalmente por incluir o erro de treinamento $v(w)$.

Quando não há erro de treinamento, $v(w)$ é igual ou próximo de zero, o limitante superior de confiança (3.22) reduz-se a:

$$\varepsilon_1(N,h,\alpha,0) = 4\varepsilon_0^2(N,h,\alpha)\ . \qquad \textbf{(3.23)}$$

### 3.4.4 Princípio da Minimização do Risco Estrutural

Define-se o *erro de generalização* $v_{gene}(w)$ como a freqüência do erro obtido para um vetor de pesos $w$ específico, quando testado em um conjunto de dados não apresentado anteriormente, denominado de dados de validação, os quais têm supostamente a mesma distribuição dos dados de treinamento.

Então, na visão do teorema da razão da convergência uniforme, podemos afirmar que, com probabilidade 1-$\alpha$ , para um número de amostras de treinamento $N > h$ , e simultaneamente para todas as funções classificadoras $F(x,w)$, o erro de generalização é menor do que o *risco garantido* definido por (Vapnik, 1992, 1998) :

$$v_{gene}(w) < v_{garant}(w) = v(w) + \varepsilon_1(N,h,\alpha,v)\ . \qquad \textbf{(3.24)}$$

Para um número $N$ fixo de dados de treinamento, o erro de treinamento decresce monotonicamente conforme a dimensão VC é aumentada, enquanto que o limitante superior de confiança aumenta proporcionalmente. Conseqüentemente, o risco garantido e o erro de generalização caminham para os seus mínimos. Estas tendências estão ilustradas na Figura 3.6.

Antes de se encontrar o ponto mínimo do risco garantido, o problema de aprendizagem é dito estar em *underfitting* (sub-ajuste) no sentido de que a dimensão VC, ou a capacidade do algoritmo, é muito pequena para o montante de dados de treinamento $N$. Depois do ponto mínimo, o problema de aprendizagem é dito estar em *overfitting* (sobre-ajuste) por causa da capacidade do algoritmo ser muito grande comparado ao montante de dados disponíveis para o treinamento.

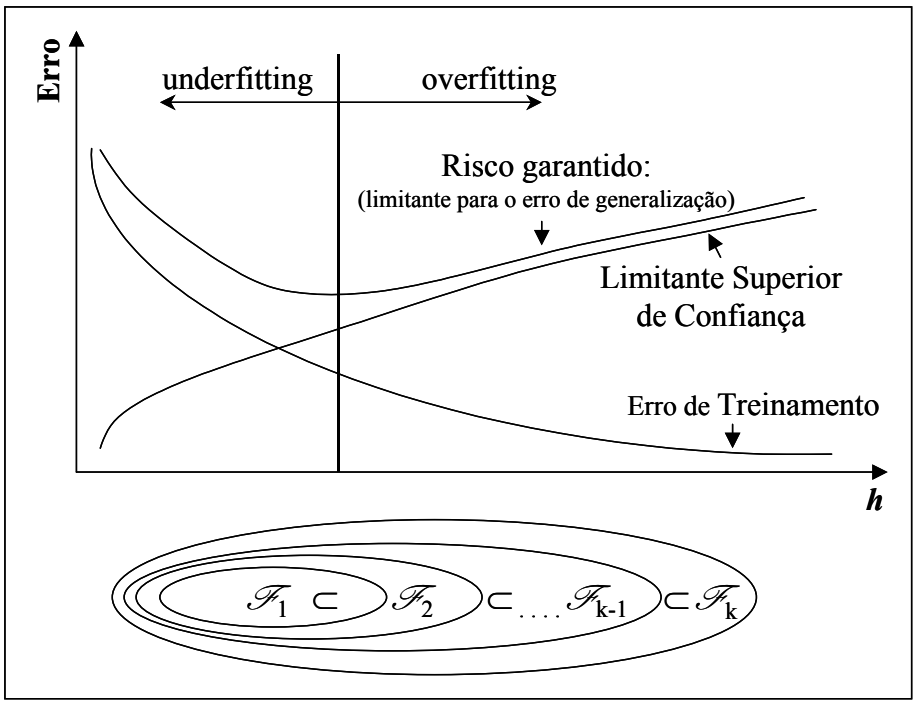


**Figura 3.6** : Relação entre erro de treinamento, limitante superior de confiança e risco garantido, quando temos o número $N$ de dados de treinamento fixo.

O desafio na resolução de um problema de aprendizado supervisionado é, portanto, obter o melhor desempenho de generalização equiparando a capacidade do algoritmo com o montante das amostras de dados de treinamento para o problema em questão. O método de minimização do risco estrutural proporciona um procedimento indutivo para encontrar este objetivo, fazendo com que a dimensão VC do algoritmo de aprendizado seja uma variável de controle (Vapnik, 1992, 1998).

Para explicar este procedimento indutivo, considere um conjunto de funções classificadoras de padrões $F(x,w)$

$$\mathscr{F}_j = \{ F(x,w): w \in W_j \} \ , \ j = 1,2,..., k$$

com este conjunto de funções tendo a seguinte estrutura (ver Figura 3.6):

$$\mathscr{F}_1 \subset \mathscr{F}_2 \subset \ldots \subset \mathscr{F}_{k-1} \subset \mathscr{F}_k$$

com as correspondentes dimensões VC satisfazendo:

$$h_1 \leq h_2 \leq \ \ldots \leq h_{k-1} \leq h_k \leq \infty,$$

implicando em dimensões VC finitas para cada função classificadora.

Portanto o método de minimização do risco estrutural procede da seguinte maneira:

1- O risco empírico (erro de treinamento) para cada função classificadora é minimizado;

2- A função classificadora $\mathscr{F}^*$ com o menor risco garantido é identificada. Esta função particular proporciona o melhor compromisso entre erro de treinamento e o limitante superior de confiança (complexidade).

O objetivo é encontrar um algoritmo específico para o qual o decrescimento da dimensão VC seja mais acentuado do que o crescimento (aumento) no erro de treinamento, e este aumento seja o menor possível.

# Capítulo 4

# Support Vector Machines para o Problema de Classificação

## 4.1 Introdução

Support Vector Machines (SVM) é uma técnica de aprendizado de máquina, fundamentada nos princípios indutivos da Minimização do Risco Estrutural. Estes princípios são provenientes da Teoria do Aprendizado Estatístico, a qual está baseada no fato de que o erro da técnica de aprendizagem junto aos dados de validação (erro de generalização) é limitado pelo erro de treinamento mais um termo que depende da dimensão VC (equações (3.21) e (3.22)).

SVM implementa um mapeamento não-linear (executado por um produto interno kernel escolhido a priori) dos dados de entrada para um espaço característico de alta-dimensão, em que um hiperplano ótimo é construído para separar os dados linearmente em duas classes.

Quando os dados de treinamento são separáveis, o hiperplano ótimo no espaço característico é aquele que apresenta a máxima margem de separação $\rho$ (ver Figura 2.1). Para dados de treinamento em que as amostras das duas classes apresentam superposição (dados não separáveis), uma generalização deste conceito é utilizada.

A técnica SVM foi introduzida por Vapnik em 1992 (Boser *et al*., 1992), formulada com todas as demonstrações matemáticas em seu livro de 1995 (Vapnik, 1995) e, em 1998, descrita em outro livro de sua autoria, com maiores detalhes (Vapnik, 1998).

Outras boas referências sobre o assunto podem ser encontradas em Cristianini & Shawe-Taylor (2000), Haykin (1999) e Burges (1998).

Com os tópicos já apresentados nos Capítulos 2 e 3, será possível descrever em detalhes os conceitos associados a SVM.

## 4.2 Caso 1 - Classes linearmente separáveis

### 4.2.1 Classes linearmente separáveis no espaço original

Para o problema de classificação binária em que as duas classes são linearmente separáveis, como descrito na Seção 2.1, o objetivo é encontrar o hiperplano ótimo, definido pela equação (2.4), maximizando a margem de separação $\rho$ (ver Figura 2.1) através da minimização da norma Euclidiana do vetor de pesos *w* do hiperplano (ver equação 2.6).

Com os conceitos de otimização apresentados na Seção 3.2, podemos formular o problema de otimização, em sua representação primal, para encontrar o hiperplano ótimo para classes linearmente separáveis:

*A partir dos dados de treinamento linearmente separáveis* $(x_i, y_i)_{1 \le i \le N}$, $x \in R^m$, $y \in \{+1, -1\}$, *onde x são os dados de entrada e y corresponde à resposta desejada, encontre o valor do vetor de pesos w e intercepto b que resolvem o seguinte problema :*

$$\begin{aligned} &\textit{Minimizar}: \quad V(w,b) = \frac{1}{2} \| w \|^2 . \\ &\textit{Sujeito a}: \quad \forall_{i=1}^{N}: \; y_i \left[ w^T x_i + b \right] \ge 1. \end{aligned} \tag{4.1}$$

O escalar 1/2 foi incluído por conveniência da representação.

De acordo com os teoremas de otimização apresentados na seção 3.2, pode-se resolver o problema (4.1) utilizando o método dos multiplicadores de Lagrange, para transformar o problema de otimização primal em seu correspondente problema dual.

Considere a função Lagrangeana referente ao problema (4.1) :

$$L(w,b,\alpha) = \frac{1}{2}(w^T w) - \sum_{i=1}^{N} \alpha_i \left[ y_i (w_i^T x_i + b) - 1 \right], \tag{4.2}$$

onde os multiplicadores de Lagrange $\alpha_i$ são todos não negativos. Sendo a função-objetivo do problema (4.1) convexa e todas as suas restrições funções afins, o correspondente problema de otimização dual é encontrado aplicando as condições de Kuhn-Tucker (ver Teorema 3.8) junto à função Lagrangeana (4.2):

$$\begin{aligned} \frac{\partial L}{\partial w}(w,b,\alpha) &= w - \sum_{i=1}^{N} y_i \alpha_i x_i = 0; \\ \frac{\partial L}{\partial b}(w,b,\alpha) &= \sum_{i=1}^{N} y_i \alpha_i = 0. \end{aligned} \tag{4.3}$$

Substituindo as relações obtidas :

$$w = \sum_{i=1}^{N} y_i \alpha_i x_i \tag{4.4}$$

e

$$\sum_{i=1}^{N} y_i \alpha_i = 0 \tag{4.5}$$

na função Lagrangeana (4.2), obtém-se :

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j (x_i^T x_j) - \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j (x_i^T x_j) + \sum_{i=1}^{N} \alpha_i \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j (x_i^T x_j). \end{aligned} \tag{4.6}$$

Agora é possível formular o problema de otimização dual :

*A partir dos dados de treinamento linearmente separáveis* $( x_i , y_i )_{1 \le i \le N}$, $x \in R^m$, $y \in \{+1, -1\}$, *encontre os multiplicadores de Lagrange* $( \alpha^*_i )_{1 \le i \le N}$ *que resolvem* o problema de otimização quadrático :

$$\begin{aligned} &\textit{Maximizar} : \quad W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j (x_i^T x_j). \\ &\textit{Sujeito a} : \quad \sum_{i=1}^{N} y_i \alpha_i = 0; \\ &\qquad \forall_{i=1}^{N} : \alpha_i \ge 0. \end{aligned} \tag{4.7}$$

O problema de otimização dual (4.7) é totalmente formulado em termos dos dados de treinamento. Além disso, a função $W(\alpha)$ a ser maximizada depende somente dos dados de entrada na forma de produto interno ( $x_i^T x_j$ ) $_{1 \le i \le N \; ; \; 1 \le j \le N}$.

Outro ponto importante é que este problema de otimização tem uma única solução que pode ser eficientemente encontrada. Portanto não há a presença de pontos de mínimos locais, como em outras técnicas de classificação.

Determinando o vetor de multiplicadores de Lagrange ótimos $\alpha^*$, pode-se calcular o vetor de pesos ótimo $w^*$ utilizando a equação (4.4),

$$w^* = \sum_{i=1}^{N} y_i \alpha_i^* x_i \qquad \text{onde } y \in \{+1, -1\},\ \alpha_i^* \in R^+ \text{ e } x \in R^m . \tag{4.8}$$

Assim $w^*$ é o vetor que encontra o hiperplano ótimo com a máxima margem de separação $\rho$ (equação (2.6)).

O valor do intercepto ótimo $b^*$ é encontrado utilizando a equação (4.8), com o auxílio das restrições primais (4.1):

$$b^* = -\frac{1}{2}\left[ \max_{\{i|y_i=-1\}} \left( \sum_{j=1}^{N_{sv}} y_j \alpha_j (x_i^T x_j) \right) + \min_{\{i|y_i=+1\}} \left( \sum_{j=1}^{N_{sv}} y_j \alpha_j (x_i^T x_j) \right) \right], \tag{4.9}$$

onde $N_{sv}$ é o numero de vetores-suporte.

Utilizando a condição de complementariedade de Kuhn-Tucker (Teorema 3.8), obtemos a seguinte relação:

$$\forall_{i=1}^{N} : \quad \alpha_i^* \left[ y_i (w^{*T} x_i + b^*) - 1 \right] = 0 \tag{4.10}$$

que proporciona uma importante informação sobre a estrutura da solução. Isto implica que somente para os dados de entrada $x_i$ para o qual a margem é 1 (e, portanto, localizados à distância $\rho$ do hiperplano) tem-se seu correspondente $\alpha^*$ diferente de zero. Todos os outros dados de entrada têm o parâmetro $\alpha^*$ igual a zero.

Através das condições de complementariedade de Kuhn-Tucker, pode-se demonstrar que :

$$(w^{*T} w^*) = \sum_{i=1}^{N_{sv}} \alpha_i^* \quad . \tag{4.11}$$

Portanto, a norma do vetor de pesos *w** que está associado ao hiperplano de máxima margem é também dado por :

$$\left\| w^* \right\| = \rho^{-1} = \left( \sum_{i=1}^{N_{sv}} \alpha_i^* \right)^{+\frac{1}{2}} . \qquad \textbf{(4.12)}$$

Como já visto no Capítulo 2 (Figura 2.1), os dados de entrada com a margem igual a 1 são chamados de vetores-suporte , sendo justamente aqueles com os multiplicadores de Lagrange $\alpha^*$ diferentes de zero. Logo são os únicos pontos que exercem influência na construção do hiperplano de máxima margem.

Além disso, o hiperplano ótimo é expresso somente em termos deste conjunto de vetores-suporte, como descrito a seguir :

$$f(x) = \text{sgn}\left( \sum_{i=1}^{N_{sv}} y_i \alpha_i^* (x_i{}^T x) + b^* \right) . \qquad \textbf{(4.13)}$$

Os dados de entrada que não são vetores-suporte também não têm nenhuma influência na função de decisão produzida pela técnica de SVM.

A função de decisão (4.13) é utilizada da seguinte maneira: se o resultado de $f(x)$ for negativo, o ponto $x$ pertence à classe negativa ; se o resultado de $f(x)$ for positivo, o ponto $x$ pertence à classe positiva.

### 4.2.2 Classes linearmente separáveis no espaço característico

Para o problema de classificação binária em que as duas classes não são linearmente separáveis no espaço original, porém apartir de um mapeamento não-linear executado por um produto interno kernel (Seção 3.3) do espaço original para um espaço característico de dimensão muito maior do que a dimensão o espaço original, o problema de classificação originalmente não-linear, torna-se linearmente separável no espaço característico.

A seguir é formulado o problema de otimização, em sua representação primal, para encontrar o hiperplano ótimo para classes linearmente separáveis no espaço característico:

*A partir dos dados de treinamento* $(x_i, y_i)_{1 \le i \le N}$, $x \in R^m$ *e* $y \in \{+1, -1\}$, *linearmente separáveis no espaço característico definido pelo produto interno kernel* $K(x_i,x_j)$, *encontre os multiplicadores de Lagrange* $(\alpha^*_i)_{1 \le i \le N}$ *que resolvem o problema de otimização quadrático* :

$$\begin{aligned} &\textit{Maximizar}: \quad W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j). \\ &\textit{Sujeito} \quad a: \quad \sum_{i=1}^{N} y_i \alpha_i = 0 \qquad ; \qquad \forall_{i=1}^{N} : \alpha_i \ge 0. \end{aligned} \tag{4.14}$$

Assim a função de decisão dada pela SVM é :

$$f(x) = \operatorname{sgn}\left( \sum_{i=1}^{N_{sv}} y_i \alpha_i^* K(x_i, x) + b^* \right) \tag{4.15}$$

que é equivalente ao hiperplano de máxima margem no espaço característico definido implicitamente pelo produto interno kernel $K(x_i,x_j)$, satisfazendo o teorema de Mercer (Seção 3.2.2) e, como conseqüência deste teorema, sendo uma matriz definida positiva. Assim, o problema (4.14) continua sendo convexo e com uma única solução.

O único grau de liberdade deste hiperplano ótimo é a escolha de qual produto interno kernel utilizar. O conhecimento prévio do problema pode ajudar na escolha do tipo de produto interno kernel, e com isto restará apenas ajustar seus parâmetros (ver equações (3.7), (3.8) e (3.9)).

## 4.3 Caso 2 - Classes não linearmente separáveis

Na Seção 2.2, foram introduzidas as variáveis de folga $\{\xi_i\}_{1 \le i \le N}$ para definir o hiperplano separador para classes não linearmente separáveis, como mostrado na equação (2.7). Para encontrar este hiperplano separador, devemos minimizar o erro de classificação incorreta dado pela função (2.8). Com isto, recai-se num problema de otimização não convexo da classe NP-completo. Para tornar este problema de otimização tratável computacionalmente, substitui-se a função de minimização do erro de classificação, dada pela equação (2.8), por uma aproximação dada pela função (2.11).

Para a simplificação dos cálculos, é possível propor a seguinte formulação do hiperplano ótimo a ser minimizado em relação ao vetor de pesos *w* (Vapnik, 1995) :

$$\theta(w,\xi) = \frac{1}{2}(w^T w) + C\sum_{i=1}^{N}\xi_i \quad . \tag{4.16}$$

A minimização do primeiro termo da equação (4.16) está relacionada à minimização da dimensão VC da SVM, como já visto no caso 1. Quanto ao segundo termo, ele pode ser visto como um limitante superior para o número de erros na amostra de treinamento. Portanto, a formulação dada pela equação (4.16) está em acordo com os princípios de minimização do risco estrutural.

O *parâmetro C* controla a relação entre a complexidade do algoritmo e o número de amostras de treinamento classificados incorretamente. Ele pode ser visto como um *parâmetro de penalização.*

O usuário é quem escolhe o parâmetro *C*, geralmente determinado experimentalmente pela validação do desempenho da SVM via conjunto de dados de validação, ou então empregando técnicas de validação cruzada baseadas no conjunto de dados de treinamento.

Portanto, pode-se formular o problema de otimização em sua representação primal para encontrar o hiperplano ótimo para classes não linearmente separáveis como:

*A partir dos dados de treinamento* $(x_i, y_i)_{1 \le i \le N}$, $x \in R^m$ *e* $y \in \{+1, -1\}$, *onde x são os dados de entrada e y corresponde à resposta desejada, encontre valores para o vetor de pesos w, intercepto b e variáveis de folga* $\xi_i$ *que resolvem o seguinte problema :*

$$\begin{aligned} &\textit{Minimizar}: \quad V(w,b,\xi) = \frac{1}{2}\| w \|^2 + C\sum_{i=1}^{N}\xi_i. \\ &\textit{Sujeito a}: \quad \forall_{i=1}^{N}: \quad y_i\left[w^T x_i + b\right] \ge 1 - \xi_i; \\ &\qquad\qquad\quad \forall_{i=1}^{N}: \quad \xi_i \ge 0. \end{aligned} \tag{4.17}$$

*onde o parâmetro C > 0 é especificado pelo usuário.*

O Caso 1, onde foram estudadas as classes linearmente separáveis, pode ser visto como um caso especial desta formulação, com todos os $\xi_i = 0$, com $1 \le i \le N$.

Utilizando o método dos multiplicadores de Lagrange, podemos transformar o problema de otimização primal em seu correspondente problema dual.

Considere a função Lagrangeana referente ao problema (4.17):

$$L(w,b,\xi,\alpha,r) = \frac{1}{2}(w^T w) + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i({w_i}^T x_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{N} r_i\xi_i \quad , \qquad \textbf{(4.18)}$$

onde os multiplicadores de Lagrange $\alpha_i$ e $r_i$ são todos não negativos. Sendo a função-objetivo do problema (4.17) convexa e todas as suas restrições funções afins, o correspondente problema de otimização dual é encontrado aplicando as condições de Kuhn-Tucker (ver Teorema 3.8) junto à função Lagrangeana (4.18):

$$\begin{aligned} \frac{\partial L}{\partial w}(w,b,\xi,\alpha,r) &= w - \sum_{i=1}^{N} y_i\alpha_i x_i = 0; \\ \frac{\partial L}{\partial b}(w,b,\xi,\alpha,r) &= \sum_{i=1}^{N} y_i\alpha_i = 0; \\ \frac{\partial L}{\partial \xi}(w,b,\xi,\alpha,r) &= C - \alpha_i - r_i = 0. \end{aligned} \qquad \textbf{(4.19)}$$

Substituindo as relações obtidas na função Lagrangeana (4.18), obtém-se:

$$L(w,b,\xi,\alpha,r) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j ({x_i}^T x_j) \qquad \textbf{(4.20)}$$

que é idêntica à relação obtida para o caso das classes linearmente separáveis (equação (4.6)). A única diferença é que a restrição $C - \alpha_i - r_i = 0$, em conjunto com $r_i \geq 0$, faz com que $\alpha_i \leq C$, *e*nquanto que $\xi_i \neq 0$ ocorre somente se $r_i = 0$.

As condições de complementaridade de Kuhn-Tucker para este caso são definidas como:

$$\begin{aligned} &\forall_{i=1}^{N}: \quad \alpha_i^*\left[y_i\left(w^* \cdot x_i + b^*\right) - 1 + \xi_i\right] = 0; \\ &\forall_{i=1}^{N}: \quad \xi_i(\alpha_i - C) = 0. \end{aligned} \qquad \textbf{(4.21)}$$

Agora é possível formular o problema de otimização dual para o caso geral em que o hiperplano ótimo é construído no espaço característico, através de um mapeamento não-linear definido implicitamente por um produto interno kernel escolhido a priori.

*A partir dos dados de treinamento* $(x_i, y_i)_{1 \le i \le N}$, $x \in R^m$ *e* $y \in \{+1, -1\}$, *e utilizando o espaço característico definido implicitamente pelo produto interno kernel* $K(x_i,x_j)$, *encontre os multiplicadores de Lagrange* $(\alpha^*_i)_{1 \le i \le N}$ *que resolvem o problema de otimização quadrático* :

$$\begin{aligned} &\textit{Maximizar}: \quad W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j). \\ &\textit{Sujeito a}: \quad \sum_{i=1}^{N} y_i \alpha_i = 0 \qquad ; \qquad \forall_{i=1}^{N} : 0 \le \alpha_i \le C. \end{aligned} \tag{4.22}$$

*onde o parâmetro* $C > 0$ *é especificado pelo usuário.*

A única e principal diferença do caso de classes separáveis para o caso de classes não separáveis (equação 4.14) é que a restrição $\alpha_i \geq 0$ é substituída por uma restrição mais forte, $0 \leq \alpha_i \leq C$.

O vetor de pesos $w$ é calculado da mesma maneira que no caso das classes linearmente separáveis (equação 4.8). O intercepto $b$ é encontrado com o auxílio das restrições primais (4.17) através da seguinte equação:

$$b^* = -\frac{1}{2}\left[ \max_{\{i|y_i=-1\}} \left( \sum_{j=1}^{N_{sv}} y_j \alpha_j K(x_i, x_j) \right) + \min_{\{i|y_i=+1\}} \left( \sum_{j=1}^{N_{sv}} y_j \alpha_j K(x_i, x_j) \right) \right].$$

Foi visto anteriormente que os dados de entrada para os quais $\alpha_i > 0$ são chamados de vetores-suporte. Para diferenciar entre os pontos com $0 < \alpha_i < C$ e aqueles com $\alpha_i = C$, a primeira categoria é chamada de vetores-suporte não limitados e a segunda categoria de vetores-suporte limitados. Se a solução contém no mínimo um vetor-suporte não limitado, ela é considerada estável, caso contrário ela é considerada instável (Joachims, 2000).

A função de decisão dada pela SVM é:

$$f(x) = \operatorname{sgn}\left( \sum_{i=1}^{N_{sv}} y_i \alpha_i^* K(x_i, x) + b^* \right). \tag{4.23}$$

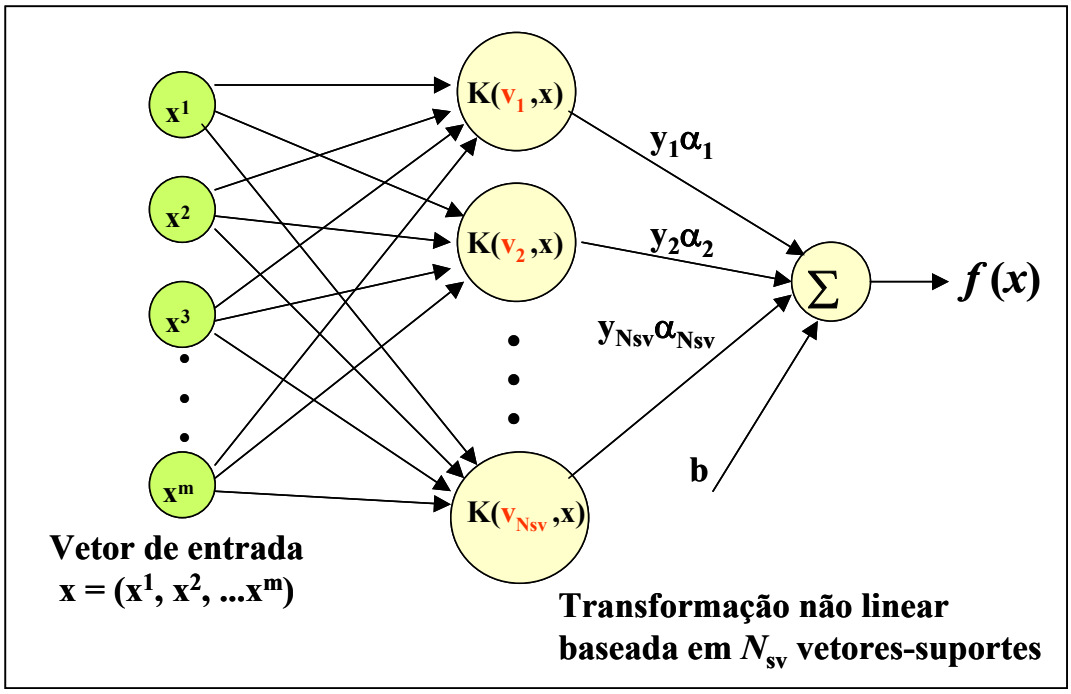


**Figura 4.1** : Arquitetura da SVM na forma de uma rede neural com uma camada escondida. O número de nós da camada de entrada é igual à dimensão do vetor de entrada. A quantidade de vetores-suporte $N_{sv}$ determina a quantidade de nós na camada escondida (onde $v$ são os vetores-suportes). O nó de saída constrói um função linear no espaço característico, o qual é determinada por uma transformação não-linear escolhida a priori por meio de produtos internos kernel.

A Figura 4.1 mostra uma representação gráfica da função de decisão (4.23), onde os valores de $f(x)$ sendo positivos indicam que a amostra pertence à classe positiva, e sendo negativos indicam que a amostra pertence à classe negativa.

Através das condições de complementaridade de Kuhn-Tucker (4.21) pode-se demonstrar que:

$$(w^{*T} w^*) = \sum_{i,j=1}^{N_{sv}} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

e, portanto, a norma do vetor de pesos $w^*$ que realiza o hiperplano de máxima margem é dada por:

$$\left\| w^* \right\| = \rho^{-1} = \left( \sum_{i,j=1}^{N_{sv}} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \right)^{+\frac{1}{2}} .$$

SVM proporciona um método que controla a complexidade da técnica independentemente da dimensão dos dados. Em particular, o problema da complexidade é resolvido em um espaço de alta dimensão, usando um hiperplano com o parâmetro $C$ (que controla a relação entre a complexidade do algoritmo e o número de amostras de treinamento classificados incorretamente) como uma superfície de decisão no espaço característico, resultando em um ótimo desempenho de generalização. O problema da dimensionalidade é reduzido pela utilização da representação dual do problema de otimização, que calcula os parâmetros do hiperplano ótimo tendo os dados de treinamento na forma de produto interno e assim formando uma matriz quadrada ("Matriz Kernel" para o caso de utilização de transformações não-lineares) de mesma dimensão da quantidade de dados de treinamento.

O treinamento da SVM consiste em um problema de otimização quadrático que é atrativo por duas razões:

- A menos de problemas numéricos ao longo dos cálculos computacionais, é garantida a convergência para um mínimo global da superfície de erro, onde o erro refere-se à diferença entre a resposta desejada e a saída da SVM;
- Os cálculos podem ser executados eficazmente, pois as restrições do problema dual são mais simples de serem resolvidas do que as restrições do problema primal.

Comparando com Redes Neurais Artificiais, o mais importante é que, utilizando o produto interno kernel, SVM calcula automaticamente parte dos parâmetros pertinentes à escolha da função kernel. Por exemplo:

- RBF - Função de Base Radial (equação (3.7)) : o número de funções de base radial e seus centros são computados automaticamente pelo número de vetores-suporte e de seus valores, respectivamente. Já as variâncias das RBF's são fixas, e especificadas a priori pelo usuário;
- Perceptron com uma camada escondida (equação (3.9)) : o número de neurônios da camada escondida e seus vetores de pesos também são computados automaticamente pelo número de vetores-suporte e de seus valores, respectivamente. Já os parâmetros da função tangente hiperbólica, $\beta_0$ e $\beta_1$ são especificados a priori pelo usuário.

## 4.4 Propriedades Estatísticas do Hiperplano Ótimo

Com base na teoria do aprendizado estatístico, será aplicado o método de minimização do risco estrutural, descrito na Seção 3.4, para construir um conjunto de hiperplanos separadores. A

estratégia será variar a dimensão VC de modo que o risco empírico (erro de treinamento) e a dimensão VC sejam ambos minimizados ao mesmo tempo, numa solução de compromisso.

A seguir, são apresentados os teoremas pertinentes:

Teorema 4.1 Vapnik (1995):

*Sejam R o raio da menor esfera contendo todos os dados de entrada* $(x_i)_{1 \le i \le N}$ *, e "a" o centro desta esfera, então :* $| x_i - a | \le R$ .

*O subconjunto de hiperplanos canônicos :* $f(x,w,b)=sgn\{(w^T x)+b\}$

*satisfazendo a restrição :*

$$\|w\| \le \frac{1}{\rho^2} = A \qquad \textbf{(4.24)}$$

*tem uma dimensão VC, h, limitada superiormente por :*

$$h \le \min\{[R^2 A^2], m_0\} + 1 , \qquad \textbf{(4.25)}$$

*onde* $\rho$ *é a margem de separação* (4.12) *e* $m_0$ *é a dimensão do espaço de entrada*.

Na Seção 3.4.2, vimos que a dimensão VC de um conjunto de hiperplanos é igual a $m_0$+1, porém a dimensão VC de um subconjunto destes hiperplanos, satisfazendo a restrição (4.24) pode ser menor quando se constrói hiperplanos com valores pequenos de $[R^2A^2]$, e com isto a capacidade de generalização destes hiperplanos será elevada.

Este teorema também afirma que se pode exercer controle sobre a dimensão VC (complexidade) do hiperplano ótimo, independentemente da dimensão $m_o$ do espaço de entrada, escolhendo apropriadamente a margem de separação $\rho$. Em SVM, a própria estrutura impõe que o hiperplano ótimo seja o que minimiza a norma euclidiana do vetor de pesos *w* do hiperplano. Como conseqüência, é maximizada a margem de separação $\rho$ e ao mesmo tempo minimizada a dimensão VC.

Na Seção 3.4, foi visto que, para alcançar uma boa capacidade de generalização, pode-se selecionar uma estrutura particular com a menor dimensão VC e erro de treinamento, de acordo com o princípio da minimização do risco estrutural. Pelas equações (4.24) e (4.25), deduz-se que esta exigência pode ser satisfeita usando o hiperplano ótimo com a maior margem de separação,

ou equivalentemente, pela equação (4.12) pode-se utilizar o vetor de pesos ótimo $w^*$ tendo a mínima norma euclidiana.

Desta maneira, a escolha do hiperplano ótimo como uma superfície de decisão não é somente intuitivamente suficiente, mas também uma completa realização do princípio de minimização do risco estrutural.

Teorema 4.2 (Vapnik, 1995):

*Se os dados de treinamento são separados pelo hiperplano ótimo (ou hiperplano ótimo generalizado), então o valor da probabilidade de cometer um erro no conjunto de dados de validação é limitado superiormente pela razão do "número de vetores-suporte" pelo "número de amostras no conjunto de treinamento menos 1"* :

$$P(erro) \leq \frac{\text{número de vetores - suporte}}{(\text{número de amostras de treinamento}) - 1} .$$

Este limitante não depende da dimensão do espaço e da norma do vetor $w$ e, portanto, se o hiperplano ótimo pode ser construído de um pequeno número de vetores-suporte, relativo à cardinalidade do conjunto de treinamento, a capacidade de generalização será alta, mesmo para um espaço de dimensão infinita.

Portanto, em SVM, a complexidade da estrutura depende do número de vetores-suporte, e não da dimensão do espaço característico.

## 4.5 Fatores de Custo

A maioria das técnicas de classificação são desenvolvidas para situações padrões (convencionais) em que as amostras do conjunto de treinamento são supostas serem derivadas da mesma distribuição de probabilidade da população, e o custo de classificação incorreta para as duas classes é suposto ser o mesmo. Muitas situações reais, entretanto, não são padrões, e as técnicas normalmente não se adaptam a estas situações. As mais freqüentes violações da suposição padrão são:

1) Diferentes tipos de classificações incorretas podem ter diferentes custos. Um tipo de classificação incorreta é muitas vezes mais sério do que outro. Isto deve ser considerado quando da construção de regras de classificação. Em particular, quando o custo esperado de

futuras classificações, ao invés da razão esperada de classificação incorreta, é utilizado para medir o desempenho do classificador;

2) A população pode não ter a mesma distribuição que aquela em que os dados de treinamento foram selecionados:
    a) Uma possível razão pode ser que a população apresente comportamentos distintos ao longo do tempo, sendo impossível obter uma solução geral, pois o perfil de variação do comportamento depende de cada problema;
    b) Outra possível razão é um vício do processo de amostragem, que ocorre em algumas situações, com as amostras de treinamento sendo geradas a partir de uma estratégia não completamente aleatória. Aqui também é muito difícil obter uma solução;
    c) Outra situação em que ocorre um vício no processo de amostragem é quando se impõe que o conjunto de dados de treinamento contenha aproximadamente o mesmo número de amostras para cada classe. Às vezes a menor classe, com relação aos dados da população, é sobre-amostrada e a maior classe é sub-amostrada numa tentativa de se ter uma amostragem balanceada. Nesta situação, a proporção das duas classes no conjunto de treinamento não reflete a atual proporção das classes na população.

Para resolver as situações 1 e 2c, é necessário introduzir os fatores de custo $C_+$ e $C_-$ para serem capazes de ajustar o custo dos falso positivos (*FP*, uma amostra da classe negativa assinalada como pertencente à classe positiva) versus os falso negativos (*FN*, uma amostra da classe positiva assinalada como pertencente à classe negativa).

Modificando a equação (4.17), resulta o seguinte problema de otimização adaptado para a aplicação dos fatores de custo:

$$\begin{aligned} &\textit{Minimizar}: \quad V(w,b,\xi) = \frac{1}{2}\|w\|^2 + C_+ \sum_{i:y_i=+1} \xi_i + C_- \sum_{j:y_j=-1} \xi_j. \\ &\textit{Sujeito a}: \quad \forall_{i=1}^{N}: \quad y_i\left[w^T x_i + b\right] \geq 1 - \xi_i; \\ &\qquad\qquad\quad \forall_{i=1}^{N}: \quad \xi_i \geq 0. \end{aligned}$$

Transformando este problema de otimização de sua representação primal para sua correspondente representação dual, obtém-se:

$$
\begin{aligned}
&\textit{Maximizar}: \quad W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j). \\
&\textit{Sujeito a}: \quad \sum_{i=1}^{N} y_i \alpha_i = 0; \\
&\qquad\qquad \forall_{i=1}^{N} : 0 \le \alpha_i \le C_+ \text{, caso } \mathrm{y_i} = +1; \\
&\qquad\qquad \forall_{i=1}^{N} : 0 \le \alpha_i \le C_- \text{, caso } \mathrm{y_i} = -1.
\end{aligned}
\tag{4.26}
$$

Felizmente, não é necessário calcular os valores exatos dos custos $C_+$ e $C_-$. O que é realmente necessário é calcular a razão entre os dois custos:

$$RC = C_+ / C_- . \tag{4.27}$$

Em Lin *et al*. (2000), é proposto o cálculo destes fatores com base nos conceitos da Regra de Bayes para situações não padrões, através da seguinte fórmula:

$$C_- = c^+ \pi^- \pi_{\mathrm{s}}^+ \quad \text{e} \quad C_+ = c^- \pi^+ \pi_{\mathrm{s}}^- , \tag{4.28}$$

onde:

$c^+$ = custo do $FP$ ;

$c^-$ = custo do $FN$ ;

$\pi^+$ e $\pi^-$ = proporções populacionais das classes positiva e negativa;

$\pi_{\mathrm{s}}^+$ e $\pi_{\mathrm{s}}^-$ = proporções dos conjunto de treinamento das classes positiva e negativa.

Vemos que este procedimento de agregação dos fatores de custo (equação (4.26)) leva a uma formulação muito semelhante à equação (4.22), onde os dois custos são iguais a 1. Portanto a implementação desta funcionalidade é fácil e conveniente em muitos problemas práticos.

## 4.6 Convertendo a resposta da SVM em probabilidade

Através da função de decisão $f(x)$ (equação (4.23)), tendo sua representação gráfica na Figura 4.1, obtém-se a resposta da SVM cujo valor não calibrado pertence ao conjunto dos números reais. Utiliza-se esta resposta da seguinte maneira:

- A resposta da função de decisão $f(x)$ tendo o valor negativo, implica que o ponto $x$ pertence à classe negativa (−1);
- A resposta da função de decisão $f(x)$ tendo o valor positivo, implica que o ponto $x$ pertence à classe positiva (+1).

Muitas vezes, o interesse não está em apenas classificar cada amostra dos dados em uma das duas classes, e sim em obter a probabilidade condicional da amostra pertencer a uma determinada classe, dado o seu vetor de variáveis de entrada, $P(y = \pm 1 \mid x)$.

Sendo a resposta da SVM um valor não calibrado (ver Figura 4.2), faz-se necessário desenvolver métodos para calibrar a resposta da SVM para assim permitir interpretá-la como uma probabilidade condicional.

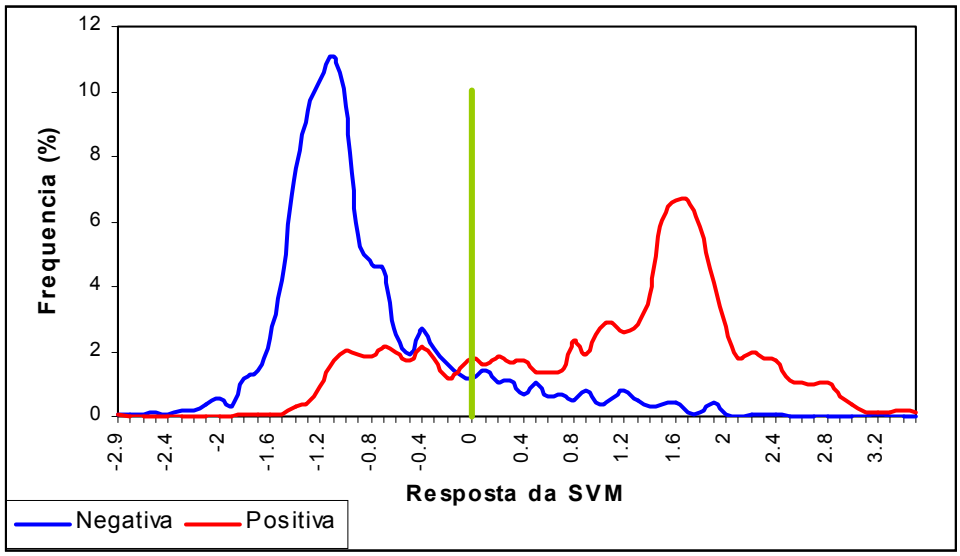


**Figura 4.2** : Histograma das probabilidades condicionais das classes (+1: positiva ; –1: negativa), $P(f(x) \mid y = \pm 1)$. A linha vertical em verde representa o ponto onde a função de decisão $f(x)$ é igual a 0. Os dados são provenientes de um problema de previsão de risco de inadimplência no mercado financeiro. Observa-se que as distribuições dos dois histogramas não se assemelham à distribuição gaussiana, pois não são distribuições simétricas, apresentando caudas longas apenas para um dos lados. Possivelmente, são provenientes de uma distribuição gamma.

A seguir, é apresentado um resumo de dois dos principais métodos desenvolvidos para o propósito de criar probabilidades condicionais para SVM :

1) Wahba (1999) propôs um método para criar probabilidades treinando diretamente a SVM com uma função de ligação logito:

$$p(x) = P(y = +1 \mid x) = \frac{1}{1 + \exp(-f(x))}$$

e com a função-objetivo a ser minimizada sendo o logaritmo negativo da função de máxima verossimilhança mais um termo que penaliza a norma de $f(x)$ em um RKHS ("Reproducing Kernel Hilbert Spaces" ; Wahba, 1999) denotado por $KH$ :

$$-\frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i+1}{2}\ln(p(x_i))+\frac{1-y_i}{2}\ln(1-p(x_i))\right)+C\left\|f(x_i)\right\|_{KH}^2 ;$$

2) Vapnik (1998) sugeriu um método para mapear a resposta da SVM em probabilidade pela decomposição do espaço característico em uma direção ortogonal ao hiperplano ótimo. A direção ortogonal para o hiperplano é parametrizada por $t$ (uma versão escalar de $f(x)$), enquanto todas as outras direções são parametrizadas por um vetor $u$. A probabilidade condicional depende de $t$ e $u$, P($y$ =+1| $t$, $u$). Vapnik propôs o ajuste da probabilidade como uma soma de co-senos:

$$P(y=+1\,|\,t,u)=a_o(u)+\sum_{i=1}^{n}a_i(u)\cos(nt) .$$

Porém, este método tem a limitação da solução do sistema linear acima ser calculada para cada iteração da SVM, elevando muito o custo computacional do algoritmo.

O método de maior utilização, pois tem o menor custo computacional e um bom desempenho, é o desenvolvido por John Platt (Platt, 1999b) que apenas treina os parâmetros de uma função sigmóide para mapear a resposta da SVM em probabilidade, não alterando em nada o mecanismo da SVM.

Platt emprega um modelo paramétrico para ajustar a probabilidade condicional $P(y=+1|\ f(x))$ utilizando os parâmetros que geram a melhor probabilidade.

Através da regra de Bayes para o cálculo da probabilidade a posteriori e da premissa de que as funções de densidade condicionaes das duas classes são funções exponenciais, Platt sugere a seguinte relação na forma sigmoidal:

$$p(x_i)=P(y=1\,|\,f(x_i))=\frac{1}{1+\exp(Af(x_i)+B)} . \qquad \textbf{(4.29)}$$

Este modelo sigmoidal é equivalente a assumir que a resposta da SVM é proporcional à :

$$\left(Af(x_i)+B\right)=\ln\left(\frac{p(x_i)}{1-p(x_i)}\right).$$

Os parâmetros $A$ e $B$ são ajustados utilizando os dados de treinamento ($f(x_i)$ , $t_i$), onde $t_i = (y_i+1)/2$, sendo $y_i = \pm 1$ , então $t_i = 0$ ou 1. Obtêm-se estes parâmetros pela minimização do logaritmo negativo da função de máxima verossimilhança.

$$Minimizar : E(p(x_i)) = -\sum_{i=1}^{N} t_i \ln(p(x_i)) + (1-t_i)\ln(1-p(x_i)) \quad . \qquad \textbf{(4.30)}$$

Esta minimização pode ser realizada utilizando-se muitos algoritmos já disponíveis, sendo que em Platt (1999b) está disponível o código de um algoritmo específico para este problema.

Portanto, a estratégia para converter a resposta da SVM em probabilidade é a seguinte:

- Treinar a SVM sem nenhuma modificação;
- Com uma matriz composta da resposta da SVM e sua respectiva classe [$f(x_i)$ ; $t_i$], treinar um algoritmo de otimização que resolva o problema (4.30), encontrando os parâmetros $A$ e $B$;
- Para predição, utiliza-se a função de decisão da SVM (equação (4.23)) e depois a equação (4.29) para obter a probabilidade da amostra pertencer à classe positiva. A probabilidade da amostra pertencer à classe negativa é o complementar da primeira.

# Capítulo 5

# Algoritmo de Implementação para Support Vector Machines

## 5.1 Introdução

O treinamento da SVM conduz a um problema de otimização quadrático com uma restrição de igualdade e varias restrições de desigualdade. Apesar do fato deste tipo de problema já ser muito bem conhecido, existem muitas questões a serem consideradas no escopo do treinamento da SVM, em particular para grande número de dados de treinamento, situação em que as técnicas de otimização padrões rapidamente tornam-se intratáveis em termos de memória e tempo de processamento.

Para tratar especificamente deste caso, será descrito neste capítulo o algoritmo de treinamento da SVM proposto por Joachims (Joachims, 1999a) denominado SVM$^{light}$. Este algoritmo é baseado na divisão (decomposição) do problema de otimização em uma série de pequenos problemas, de modo que cada pequeno problema possa ser eficazmente resolvido.

São os elementos-chave do algoritmo a estratégia para encontrar uma boa decomposição e um método para reduzir o tamanho do problema pela exclusão de variáveis irrelevantes.

O algoritmo SVM$^{light}$ está disponível na Web no site http://svmlight.joachims.org/ . Utilizado em diversas aplicações, principalmente em problemas de classificação de textos e reconhecimento de padrões, este algoritmo mostra-se, quando comparado ao conhecido algoritmo SMO "Sequential Minimal Optimization" (Platt, 1999a), o mais rápido e com o melhor desempenho para um grande número $N$ de dados de treinamento, por exemplo, $N$ =10.000.

## 5.2 Abordagem do Problema

O problema de otimização quadrático para o treinamento da SVM, mostrado no Capítulo 4 para o caso mais geral de classes não linearmente separáveis (equação (4.22)), será o foco desta abordagem. Por conveniência, este problema de maximização em sua representação dual será transformado em um problema de minimização, como segue:

**PO(1)** - Problema de Otimização 1:

$$Minimizar: \quad W(\alpha) = -\sum_{i=1}^{N} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j). \tag{5.1}$$

$$Sujeito\ \ a: \quad \sum_{i=1}^{N} y_i \alpha_i = 0; \tag{5.2}$$

$$\forall_{i=1}^{N} : 0 \le \alpha_i \le C, \tag{5.3}$$

onde $N$ é o número de dados de treinamento, $x \in R^m$, $y \in \{+1, -1\}$ e $K$ é um função kernel.

Definindo a matriz $Q$ como $(Q)_{ij} = y_i y_j K(x_i, x_j)$, podemos escrever o PO(1) como:

$$Minimizar: \ \ W(\alpha) = -\alpha^T + \frac{1}{2} \alpha^T Q \alpha. \tag{5.4}$$

$$Sujeito\ \ a: \ \ \alpha^T y = 0; \tag{5.5}$$

$$\vec{0} \le \alpha \le C\vec{1}\,. \tag{5.6}$$

O tamanho do problema de otimização depende do número de dados de treinamento $N$. Uma vez que o tamanho da matriz $Q$ é $N^2$, para problemas com $N$ da ordem de 10.000, a quantidade de memória requerida passa a ser intratável computacionalmente, pois muitas implementações padrões de algoritmos para resolverem problemas de otimização quadráticos necessitam armazenar $Q$ integralmente. Uma alternativa poderia ser recalcular cada termo da matriz $Q$ quando necessário, mas isto se torna muito custoso quando estes cálculos são muito freqüentes.

Uma maneira de treinar a SVM para problemas com muitos dados de treinamento é decompor o problema em uma série de pequenas tarefas. Esta decomposição divide o PO(1) em uma parte inativa e uma parte ativa chamada de "conjunto de trabalho". A principal vantagem

desta decomposição é que o algoritmo irá requerer uma quantidade de memória que se relaciona linearmente com o número de dados de treinamento e linearmente com o número de vetores-suporte. Uma desvantagem é que este algoritmo necessitará de mais tempo de processamento. Para implementar esta estratégia, será descrito o algoritmo $SVM^{light}$, que incorpora as seguintes idéias:

- Um método eficaz para selecionar o conjunto de trabalho;
- Sucessivas reduções do problema de otimização, explorando propriedades presentes em problemas tratados via SVM :
  - Muito menos vetores-suporte do que dados de treinamento;
  - Muitos vetores-suporte limitados, ou seja, com $\alpha_i = C$.
- Melhorias computacionais, como o armazenamento dos produtos internos kernel e sucessivas atualizações no gradiente e no critério de parada.

## 5.3 Algoritmo geral para a tarefa de Decomposição

Nesta seção, será apresentada uma versão generalizada da estratégia de decomposição proposta por Osuna *et al.* (1997a).

Em cada iteração, a variável $\alpha_i$ do PO(1) é dividida em duas categorias:

1. O conjunto $B$ de variáveis livres (conjunto de trabalho): são aquelas que podem ser atualizadas na iteração corrente. Este conjunto tem o tamanho constante $q$ muito menor do que $N$.
2. O conjunto $D$ de variáveis fixas: são aquelas fixadas temporariamente em um valor particular.

O algoritmo trabalha da seguinte maneira:

- Enquanto as condições de otimalidade são violadas:
  - Selecionar $q$ variáveis para o conjunto de trabalho $B$, e as $N–q$ variáveis restantes permanecem fixas;
  - Decompor o problema e resolver o subproblema de otimização quadrático, minimizando $W(\alpha)$ (5.1) em $B$.
- Terminar e retornar $\alpha$.

A questão que surge é como o algoritmo detecta que foi encontrado o valor ótimo para $\alpha$. Como no PO(1) é garantido haver uma matriz Hessiana $Q$ semi-definida positiva e com todas as restrições lineares, então PO(1) é um problema de otimização convexo. Para estas classes de problema, as condições de Kuhn-Tucker são condições necessárias e suficientes de otimalidade.

Denotando o multiplicador de Langrange para a restrição de igualdade (5.5) como $\lambda^{eq}$ e os multiplicadores de Lagrange para os limites inferiores e superiores das restrições (5.6) como $\lambda^{inf}$ e $\lambda^{sup}$, $\alpha$ é o ótimo do PO(1) se existirem $\lambda^{eq}$ , $\lambda^{inf}$ e $\lambda^{sup}$ que satisfaçam:

$$g(\alpha) + (\lambda^{eq} y - \lambda^{\inf} + \lambda^{\sup}) = \vec{0} \ ; \tag{5.7}$$

$$\forall_{i=1}^{N} : \lambda_i^{\inf}(-\alpha_i) = 0 \ ; \tag{5.8}$$

$$\forall_{i=1}^{N} : \lambda_i^{\sup}(\alpha_i - C) = 0 \ ; \tag{5.9}$$

$$\lambda^{\sup} \geq \vec{0} \ ; \tag{5.10}$$

$$\lambda^{\inf} \geq \vec{0} \ ; \tag{5.11}$$

$$\alpha^T y = 0 \ ; \tag{5.12}$$

$$\vec{0} \leq \alpha \leq C\vec{1} \ , \tag{5.13}$$

onde $g(\alpha)$ é o vetor das derivadas parciais de $\alpha$ em relação ao PO(1):

$$g(\alpha) = -\vec{1} + Q\alpha \ . \tag{5.14}$$

Se as condições de otimalidade não são satisfeitas, o algoritmo decompõe o PO(1) e resolve um problema menor de otimização quadrático. A decomposição assegura que isto conduzirá a um decréscimo na função-objetivo $W(\alpha)$ se o conjunto de trabalho $B$ satisfizer algumas condições (Osuna *et al*., 1997b).

Considere $\alpha$, $y$ e $Q$ apropriadamente particionados com respeito aos conjuntos de variáveis $B$ e $D$, na forma:

$$\alpha = \begin{vmatrix} \alpha_B \\ \alpha_D \end{vmatrix} \quad , \qquad y = \begin{vmatrix} y_B \\ y_D \end{vmatrix} \quad , \qquad Q = \begin{vmatrix} Q_{BB} & Q_{BD} \\ Q_{DB} & Q_{DD} \end{vmatrix} \ . \tag{5.15}$$

Então, uma vez que $Q$ é simétrica ($Q_{BD} = Q_{DB}$), podemos escrever o seguinte problema de otimização:

**PO(2)** - Problema de Otimização 2:

$$\textit{Minimizar}: \; W(\alpha_B) = -\alpha_B^T(\vec{1} - Q_{BD}\alpha_D) + \frac{1}{2}\alpha_B^T Q_{BB}\alpha_B + \frac{1}{2}\alpha_D^T Q_{DD}\alpha_D - \alpha_D^T. \quad (5.16)$$

$$\textit{Sujeito a}: \; \alpha_B^T y_B + \alpha_D^T y_D = 0 \; ; \quad (5.17)$$

$$\vec{0} \le \alpha \le C\vec{1} \; . \quad (5.18)$$

Sendo as variáveis em $D$ fixas, os termos ${\alpha_D}^T Q_{DD}\alpha_D$ e $\alpha_D$ são constantes e assim podem ser excluídos sem alterar a solução do PO(2).

Como $\alpha_i$ é ótimo no conjunto de trabalho $B$, é importante selecionar adequadamente bons conjuntos de trabalho.

PO(2) é um problema de programação quadrática onde se tem uma matriz semi-definida positiva. Agora, este problema é pequeno o bastante para ser resolvido pelos métodos padrões de otimização.

O algoritmo SVM$^{light}$ utiliza como método de otimização para resolver PO(2) o método de Hildreth e D'Espo descrito em Wismer & Chattergy (1978) (HIDEO), que é adaptado para tratar de problemas com matrizes semi-definidas pela exclusão de variáveis que correspondem a partes linearmente dependentes da matriz hessiana.

Outro método que pode ser utilizado é o método de pontos interiores primal-dual (Vanderbei, 1994), implementado por A. Smola (Smola, 1998) (LOQO). Outros métodos já propostos na literatura também podem ser facilmente incorporados na solução do PO(2).

## 5.4 Selecionando Bons Conjuntos de Trabalho

Na seleção do conjunto de trabalho, é desejável escolher um conjunto de variáveis para o qual o valor da função-objetivo $W(\alpha)$ irá progredir mais para o seu mínimo na iteração corrente. Para este propósito, a estratégia baseia-se no método de Zoutendijk (Zoutendijk, 1970), que utiliza uma aproximação de primeira ordem da função-objetivo (equação (5.14)) com a idéia de encontrar uma direção factível de descida $d$ que tenha somente $q$ elementos diferentes de zero. As variáveis correspondentes a estes elementos irão compor o conjunto de trabalho.

A seguir, será apresentado o problema de otimização tal que sua solução descreve quais variáveis entrarão no conjunto de trabalho em cada iteração ($t$):

**PO(3)** - Seleção do Conjunto de Trabalho:

$$Minimizar: \quad V(d) = g(\alpha^{(t)})^T d. \tag{5.19}$$

$$Sujeito\ a: \quad y^T d = 0 \ ; \tag{5.20}$$

$$d_i \geq 0 \quad para\ i: \ \alpha_i = 0 \ ; \tag{5.21}$$

$$d_i \leq 0 \quad para\ i: \ \alpha_i = C \ ; \tag{5.22}$$

$$-\vec{1} \leq d \leq \vec{1} \ ; \tag{5.23}$$

$$\left|\{d_i : d_i \neq 0\}\right| = q \ . \tag{5.24}$$

A função-objetivo (5.19) determina que a direção de descida é garantida.

As restrições (5.20), (5.21) e (5.22) asseguram que a direção de descida é projetada ao longo da restrição de igualdade (5.5) e obedecem às restrições ativas de desigualdade (5.6).

A restrição (5.23) normaliza a direção de descida .

A restrição (5.24) condiciona que a direção de descida tenha somente *q* variáveis, as quais serão incluídas no conjunto de trabalho *B*.

A convergência do algoritmo de otimização composto pela estratégia de seleção, pelas condições de otimalidade e pela decomposição tem que satisfazer às seguintes condições mínimas:

- terminar somente quando a solução ótima é encontrada;
- se não estiver na solução ótima, escolher um passo em direção ao ótimo.

A primeira condição pode ser facilmente avaliada checando as condições necessárias e suficientes de otimalidade (5.7) a (5.13), a cada iteração.

Na segunda condição, o passo em direção ao ótimo do PO(1) é garantido desde que a estratégia de seleção do conjunto de trabalho retorne as variáveis para o PO(2) e a solução deste levará a um decréscimo do valor de sua função-objetivo (5.16). Além disto, a solução do PO(2) é também factível para o PO(1) e também levará ao decréscimo do valor de sua função-objetivo (5.1). Isto significa que a função-objetivo (5.1) desce estritamente em cada iteração. Quanto ao uso da direção de descida *d*, encontrada utilizando o algoritmo de Zoutendijk, é fato que o algoritmo utilizado para resolver o PO(3) não converge para o ótimo no caso de algumas classes de problemas (Wolfe, 1972). Porém, a direção *d* não é diretamente utilizada em cada iteração, mas sim o conjunto de trabalho de tamanho *q* selecionado pelos componentes do vetor de direção *d* não nulos.

Uma alternativa ainda mais prática para selecionar o conjunto de trabalho, ou seja, a solução do PO(3), é utilizar a seguinte estratégia : sendo $z_i = y_i g_i(\alpha^{(t)})$, ordene $\alpha_i$ de acordo com $z_i$ em ordem decrescente e, sendo $q$ um número par, então selecione sucessivamente os $q/2$ elementos do topo da lista para o qual $0 < \alpha_i^{(t)} < C$, ou $d_i = -y_i$ obedece (5.21) e (5.22). De forma similar, selecione os $q/2$ elementos da parte inferior da lista para o qual $0 < \alpha_i^{(t)} < C$, ou $d_i = y_i$ obedece (5.21) e (5.22). Estas $q$ variáveis compõem o conjunto de trabalho.

## 5.5 Reduzindo o Número de Variáveis

Para muitos problemas, o número de vetores-suporte é muito menor do que o número de dados de treinamento. Se conhecermos a priori quais os dados de treinamento que serão vetores-suporte, é possível treinar a SVM somente com estes dados e ainda assim obter o mesmo resultado que treinando a SVM com todo o conjunto de dados de treinamento. Isto faria o PO(1) menor e mais rápido de ser resolvido, já que se economiza tempo e espaço significativos ao não calcular parte da matriz Hessiana $Q$, correspondente aos dados que não são vetores-suporte.

De modo similar, para problemas com muito ruído existem muitos vetores-suporte com $\alpha_i$ em seu limite superior $C$. Vamos chamar estes vetores-suporte de vetores-suporte limitados (SVL). Argumentos similares para dados que não são vetores-suporte aplicam-se a SVL. Caso se conheça a priori quais dados de treinamento serão SVL, o $\alpha_i$ correspondente poderia ser fixado em $C$, conduzindo a um novo problema de otimização, com menos variáveis.

Durante o processo do otimização, freqüentemente nas primeiras iterações, se consegue detectar que determinados dados são improváveis de se tornarem vetores-suporte ou aqueles que serão SVL. Eliminando estas variáveis do PO(1), se consegue um problema menor PO(1)' de tamanho $N$', e deste problema pode-se construir a solução do PO(1).

Seja:

$X$ : os índices correspondendo aos vetores-suporte não SVL;

$Y$ : os índices correspondendo aos SVL;

$Z$ : os índices dos dados que não são vetores-suporte;

a transformação do PO(1) para o PO(1)' pode ser feita usando a decomposição similar à aplicada ao PO(2). Tomando $\alpha$, $y$ e $Q$ apropriadamente particionados com respeito a $X$, $Y$ e $Z$, na forma:

$$\alpha = \begin{vmatrix} \alpha_X \\ \alpha_Y \\ \alpha_Z \end{vmatrix} = \begin{vmatrix} \alpha_X \\ C\vec{1} \\ \vec{0} \end{vmatrix} \quad , \quad y = \begin{vmatrix} y_X \\ y_Y \\ y_Z \end{vmatrix} \quad , \quad Q = \begin{vmatrix} Q_{XX} & Q_{XY} & Q_{XZ} \\ Q_{YX} & Q_{YY} & Q_{YZ} \\ Q_{ZX} & Q_{ZY} & Q_{ZZ} \end{vmatrix} . \qquad (5.25)$$

A decomposição de $W(\alpha)$ é apresentada como:

**PO(4)** - Problema de Otimização 4:

$$\textit{Minimizar}: \quad W(\alpha_X) = -\alpha_X^T \left(\vec{1} - (Q_{XY}\vec{1})C\right) + \frac{1}{2}\alpha_X^T Q_{XX}\alpha_X + \frac{1}{2}C\vec{1}^T Q_{YY}C\vec{1} - |Y|C. \quad (5.26)$$

$$\textit{Sujeito a}: \quad \alpha_X^T y_X + C\vec{1}^T y_Y = 0 \ ; \qquad (5.27)$$

$$\vec{0} \le \alpha_X \le C\vec{1} \ . \qquad (5.28)$$

Sendo $C\vec{1}^T Q_{YY}C\vec{1} - |Y|C$ um valor constante, então este termo pode ser descartado sem ocorrer mudanças na solução.

Por enquanto, ainda não está claro como o algoritmo pode identificar quais dados podem ser eliminados. É desejável encontrar condições que indiquem o mais cedo possível, no processo de otimização, que certas variáveis não irão se tornar vetores-suporte ou SVL. Visto que condições suficientes ainda não são conhecidas para esta tarefa, uma aproximação heurística baseada na estimativa dos multiplicadores de Lagrange é utilizada.

Na solução, um valor positivo do multiplicador de Lagrange de uma restrição de desigualdade indica que a variável está no ótimo quando está no limite superior. Seja $A$ o conjunto corrente dos $\alpha_i$ satisfazendo $0 < \alpha_i < C$, então resolvendo a equação (5.7) para $\lambda^{eq}$ e calculando a média em relação a todos os $\alpha_i$ do conjunto $A$, temos assim uma estimativa para $\lambda^{eq}$:

$$\lambda^{eq} = \frac{1}{|A|}\sum_{i \in A}\left[ y_i - \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) \right] . \qquad (5.29)$$

Como as variáveis $\alpha_i$ não podem estar simultaneamente nos limites superiores e inferiores, os multiplicadores das restrições de desigualdade podem ser estimados por :

$$\lambda_i^{\inf} = +y_i \left( \left[ \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) \right] + \lambda^{eq} \right) - 1 \ , \qquad (5.30)$$

$$\lambda_i^{\text{sup}} = -y_i \left( \left[ \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) \right] + \lambda^{eq} \right) + 1 \quad . \tag{5.31}$$

Considerando o histórico dos multiplicadores de Lagrange estimados nas últimas *h* iterações, se as estimativas de (5.30) ou (5.31) são positivas (ou acima de algum limiar positivo Θ) em cada uma das últimas *h* iterações, isto também será verdade na solução ótima. Portanto, estas variáveis são eliminadas usando a decomposição do PO(4), em que estas variáveis são fixas e o gradiente e as condições de otimalidade não são calculados. Isto conduz a uma redução substancial do número de estimativas dos produtos internos kernel.

Como esta heurística pode falhar, as condições de otimalidade para a exclusão de variáveis são checadas depois da convergência do PO(1)'.

## 5.6 Questões para uma Implementação Eficiente

### 5.6.1 Critério de Parada

Existem dois caminhos óbvios para definir um critério de parada que opere adequadamente na estrutura do algoritmo apresentado até então.

No primeiro, a solução do PO(3) pode ser utilizada para definir uma condição necessária e suficiente para a otimalidade, ou seja, se a equação (5.19) é igual a zero, o PO(1) é resolvido com o $\alpha^{(t)}$ corrente como solução. Porém, com este critério de parada fica difícil especificar uma precisão desejada de uma maneira intuitiva e significativa.

O segundo caminho, que é o utilizado pelo algoritmo SVM$^{\text{light}}$ como critério de parada, deriva das condições de otimalidade (5.7) a (5.13). Utilizando o mesmo raciocínio empregado nas equações (5.29) a (5.31), as seguintes condições com $\varepsilon = 0$ são equivalentes às condições (5.7) a (5.13):

$$\forall i \ com \ \ 0 < \alpha_i < C : \qquad \lambda^{eq} - \xi \le y_i - \left[ \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) \right] \le \lambda^{eq} + \xi \quad , \tag{5.32}$$

$$\forall i \ com \ \alpha = 0: \qquad y_i - \left( \left[ \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) \right] + \lambda^{eq} \right) \geq 1 - \xi \ , \tag{5.33}$$

$$\forall i \ com \ \alpha = C: \qquad y_i - \left( \left[ \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) \right] + \lambda^{eq} \right) \leq 1 + \xi \ , \tag{5.34}$$

$$\alpha^T y = 0 \ . \tag{5.35}$$

As condições de otimalidade (5.32), (5.33) e (5.34) são sugestivas pois elas refletem as restrições do problema de otimização primal (4.17). Na prática, estas condições não precisam ser satisfeitas com alta precisão. Utilizando uma tolerância de $\xi$ = 0.001 é aceitável para a maioria dos problemas. Utilizando uma precisão maior, pode não resultar em uma melhora no desempenho de generalização, e certamente vai promover um aumento considerável do tempo de treinamento (Joachims, 1999a).

### 5.6.2 Cálculo do Gradiente e do Critério de Parada

A eficiência do algoritmo de otimização depende muito de como as rotinas podem ser executadas eficientemente a cada iteração. As seguintes tarefas são necessárias a cada iteração :

- Calculo do vetor das derivadas parciais $g(\alpha^{(t)})$, para selecionar o conjunto de trabalho.
- Obtenção do valor das equações (5.32), (5.33) e (5.34) para o critério de parada.
- Obtenção das matrizes $Q_{BB}$ e $Q_{BN}$ para o PO(2).

Felizmente, devido à estratégia de decomposição, todas estas tarefas podem ser calculadas ou atualizadas sabendo somente $q$ linhas da matriz Hessiana $Q$. Estas $q$ linhas correspondem às variáveis do conjunto de trabalho corrente e seus valores são calculados logo depois da seleção do conjunto de trabalho e armazenados ao longo de toda a iteração.

O cálculo do gradiente $g(\alpha^{(t)})$ (equação (5.14)) e dos critérios de parada (5.32) a (5.34) pode ser realizado com facilidade caso se saiba o valor de $s_i^{(t)}$ introduzido a seguir:

$$s_i^{(t)} = \sum_{j=1}^{n} \alpha_j y_j K(x_i, x_j) \ , \tag{5.36}$$

Quando $\alpha^{(t-1)}$ é atualizado para $\alpha^{(t)}$, o vetor $s^{(t)}$ precisa também ser atualizado. Isto pode ser feito eficientemente e com boa precisão da seguinte maneira:

$$s_i^{(t)} = s_i^{(t-1)} + \sum_{j \in B} \left( \alpha_j^{(t)} - \alpha_j^{(t-1)} \right) y_j K(x_i, x_j) \quad . \tag{5.37}$$

Apenas as variáveis pertencentes ao conjunto de trabalho $B$ são necessárias para a atualização das linhas de $s^{(t)}$ e como conseqüência da matriz Hessiana $Q$. O mesmo também é verdade para $Q_{BB}$ e $Q_{BD}$, que são apenas subconjuntos das colunas destas linhas.

### 5.6.3 Armazenagem do cálculo dos Produtos Internos Kernel

Para os produtos internos kernel não-lineares, o passo mais custoso a cada iteração é a sua estimação para o cálculo da matriz Hessiana $Q$. Ao longo de todo o processo de otimização, eventuais vetores-suporte entram no conjunto de trabalho muitas vezes. Para evitar o recálculo de parte da Hessiana correspondente aos dados que freqüentemente entram e saem do conjunto de trabalho, o algoritmo armazena a estimativa de parte da Hessiana correspondente a estes dados. Isto permite um elegante relacionamento entre consumo de memória e tempo de treinamento (Joachims, 1999a).

Quando o espaço reservado para a armazenagem está cheio, o dado menos utilizado nas últimas iterações é removido para a armazenagem de dados da iteração corrente.

Todas as vezes que ocorre redução do número de variáveis (seção 5.5), os dados armazenados são novamente particionados.

## 5.7 Outras Maneiras de Implementação da SVM

O treinamento da SVM conduz a um problema de otimização quadrático em que os métodos padrões de otimização são aplicáveis, como o Quasi-Newton, Gradiente Conjugado e o método de Pontos Interiores Primal-Dual. Estes métodos podem executar o treinamento da SVM rapidamente, mas as implementações adotadas na literatura têm a desvantagem de armazenar a matriz $Q$ na memória. Para pequeno número de dados de treinamento estes métodos são a melhor escolha, porém para um grande número de dados eles tornam-se intratáveis em termos de memória e tempo de processamento.

Os métodos em que os componentes da matriz $Q$ são calculados e descartados durante o aprendizado da SVM se adaptam melhor a problemas com grande número de dados de treinamento. Para esta categoria, o método mais conhecido é o algoritmo Kernel-Adatron (Friess *et al*. , 1998) que, semelhante ao procedimento do método do gradiente descendente quando aplicado ao PO(1), iterativamente atualiza o valor de $\alpha_i$ utilizando a seguinte equação:

$$\alpha_i \leftarrow \alpha_i + \eta \frac{\partial W(\alpha)}{\partial \alpha_i};$$

$$\alpha_i \leftarrow \alpha_i + \eta \left( 1 - y_i \sum_{j=1}^{N} \alpha_j y_j K(x_i, x_j) \right) , \qquad \textbf{(5.38)}$$

onde $\eta$ é o tamanho do passo de descida do gradiente

O algoritmo Kernel-Adatron tem as limitações de resolver apenas problemas em que as classes são separáveis e que o hiperplano passe pela origem, ou seja, sem o intercepto $b$. Com estas limitações, a restrição de igualdade (5.2) do PO(1) não é mais necessária. Resta apenas resolver as restrições de desigualdade (5.3) com a seguinte estratégia:

- se o resultado de $\alpha_i$ na equação (5.38) for negativo, então force $\alpha_i$ a ser igual a 0;
- se o resultado de $\alpha_i$ na equação (5.38) for maior do que $C$, então force $\alpha_i$ a ser igual a $C$.

Esta estratégia para eliminar as duas restrições do PO(1) é o que torna viável a utilização de métodos de otimização irrestrita em contextos com restrições, como os métodos mencionados no começo desta seção.

Outra classe de métodos são os que trabalham com a estratégia de decomposição e divisão do conjunto de dados de treinamento em parte menores, como já estudado neste Capítulo. Quem primeiro explorou esta estratégia foi Boser *et al*. (1992) com o algoritmo denominado "chunking". Porém este algoritmo não era rápido. Nesta classe de algoritmos, os que são mais conhecidos são o SMO e o SVM$^{light}$ que já foi apresentado neste Capítulo.

O algoritmo SMO "Sequential Minimal Optimization" (Platt, 1999a), considerado um dos algoritmos mais competitivos e mais utilizados para o treinamento de SVM, pode ser visto com um caso especial do algoritmo SVM$^{light}$. No algoritmo SMO, o conjunto de trabalho é de tamanho $q$ fixo e igual a 2, permitindo que o PO(2) possa ser resolvido analiticamente, evitando a implementação de um algoritmo para resolver o problema quadrático PO(2) , LOQO ou HIDEO, por exemplo.

SMO difere também na estratégia de seleção do conjunto de trabalho, pois utiliza um conjunto de heurísticas motivadas pelas condições de Karush-Kuhn-Tucker. Na prática, estas heurísticas produzem o mesmo resultado da estratégia de seleção do algoritmo $SVM^{light}$ .

Porém, a estratégia de redução do número de variáveis (seção 5.5) e a estratégia de armazenagem (seção 5.6.3) não são partes do algoritmo SMO, e fazem com que o algoritmo $SVM^{light}$ seja mais rápido em termos de tempo de processamento do que o SMO.

Outras duas abordagens de como implementar a SVM merecem destaque, por abordarem diferentes características do problema. Elas estão descritas a seguir.

Mangasarian e Musicant (2000) exploraram uma nova simplificação da formulação da SVM para a sua implementação. Eles impuseram que o hiperplano passasse pela origem e que o erro de classificação incorreta, medido linearmente pelas variáveis de folga $\xi$, fosse medido pelo quadrado da mesma variável, $\xi^2$. Além disso, eles restringiram a implementação apenas para SVM lineares, sem a utilização de produtos internos kernels. Explorando estas modificações, eles apresentaram um algoritmo que a cada iteração requer a inversão de uma matriz de dimensão igual ao numero de atributos *m*, e não com a cardinalidade do conjunto de dados de treinamento *N*. Estas implementações tornaram o algoritmo muito rápido para dados pertencentes a um espaço de baixa dimensão. Porém este algoritmo é inapropriado quando o número de atributos *m* é grande.

Keerthi *et al*. (1999) propuseram um algoritmo diferente daqueles que trabalham com a formulação primal-dual. Eles mostraram que o problema de treinamento da SVM pode ser transformado em um problema de calcular a menor distância entre dois politopos convexos, cada um contendo os dados pertencentes a uma classe. Esta transformação requer que os dados de treinamento sejam linearmente separáveis, e que o erro de classificação incorreta seja medido pelo quadrado da variável de folga, $\xi^2$.

### 5.7.1 Razão da escolha do algoritmo $SVM^{light}$

O motivo da escolha do algoritmo $SVM^{light}$ para implementar o treinamento da SVM foi baseado nos seguintes fatos:

- É projetado para operar com grande número de dados de treinamento não tendo problema com quantidade de informação armazenada na memória;
- O tempo de processamento para grandes tarefas é muito satisfatório;

- Trabalha com problemas de todos os tipos: classes separáveis, classes não separáveis e ainda problemas com muita interseção (ruído) entre as classes;
- Não têm nenhuma restrição quanto ao intercepto *b*.

Estas características são todas necessárias para viabilizar as aplicações do Capítulo 7, onde os dados pertencem a um espaço de alta dimensão, o número de dados de treinamento é grande e a dificuldade para separar as classes é elevada devido à existência de interseção entre elas.

As características apresentadas acima, são também satisfeitas pelo algoritmo SMO, que também poderia ser utilizado, mas os testes que foram feitos comparando os dois algoritmos, SMO e $SVM^{light}$, junto aos dados das aplicações do Capítulo 7, mostraram que $SVM^{light}$ é mais rápido do que SMO.

# Capítulo 6

# Inferência Transdutiva aplicada a Support Vector Machines

## 6.1 Introdução

Neste capítulo, serão abordados os seguintes tópicos:

- Os princípios da inferência transdutiva;
- Aspectos da teoria do aprendizado estatístico relacionados à inferência transdutiva;
- Aplicação dos princípios da inferência transdutiva tendo SVM como técnica de classificação;
- Apresentação do algoritmo proposto por Joachims (1999b) para a implementação dos conceitos da inferência transdutiva associada à técnica SVM.

## 6.2 Inferência Transdutiva

A *inferência transdutiva* foi introduzida junto à teoria do aprendizado estatístico por Vapnik (1998) com a idéia principal de construir um classificador utilizando dois conjuntos de dados: o tradicional de treinamento, em que as amostras já estão previamente classificadas, e o conjunto de predição, em que as amostras não estão classificadas. O objetivo é classificar os dados pertencentes ao conjunto de predição. Treinando o classificador com estes dois conjuntos de dados, é possível classificar os dados do conjunto de predição diretamente em um único passo, e como principal vantagem teremos o aumento de informação disponível para o treinamento do algoritmo, e conseqüentemente uma melhora da generalização e desempenho do classificador.

A inferência transdutiva representa um modo alternativo para o método tradicional, a inferência indutiva, a qual necessita de dois passos para classificar as amostras do conjunto de predição, conforme ilustrado na Figura 6.1:

- O passo *indutivo* consiste em descobrir a dependência funcional entre as variáveis de entrada-saída;
- O passo *dedutivo*, utiliza esta dependência funcional para avaliar a saída dos pontos de interesse, ou seja, classificá-los.

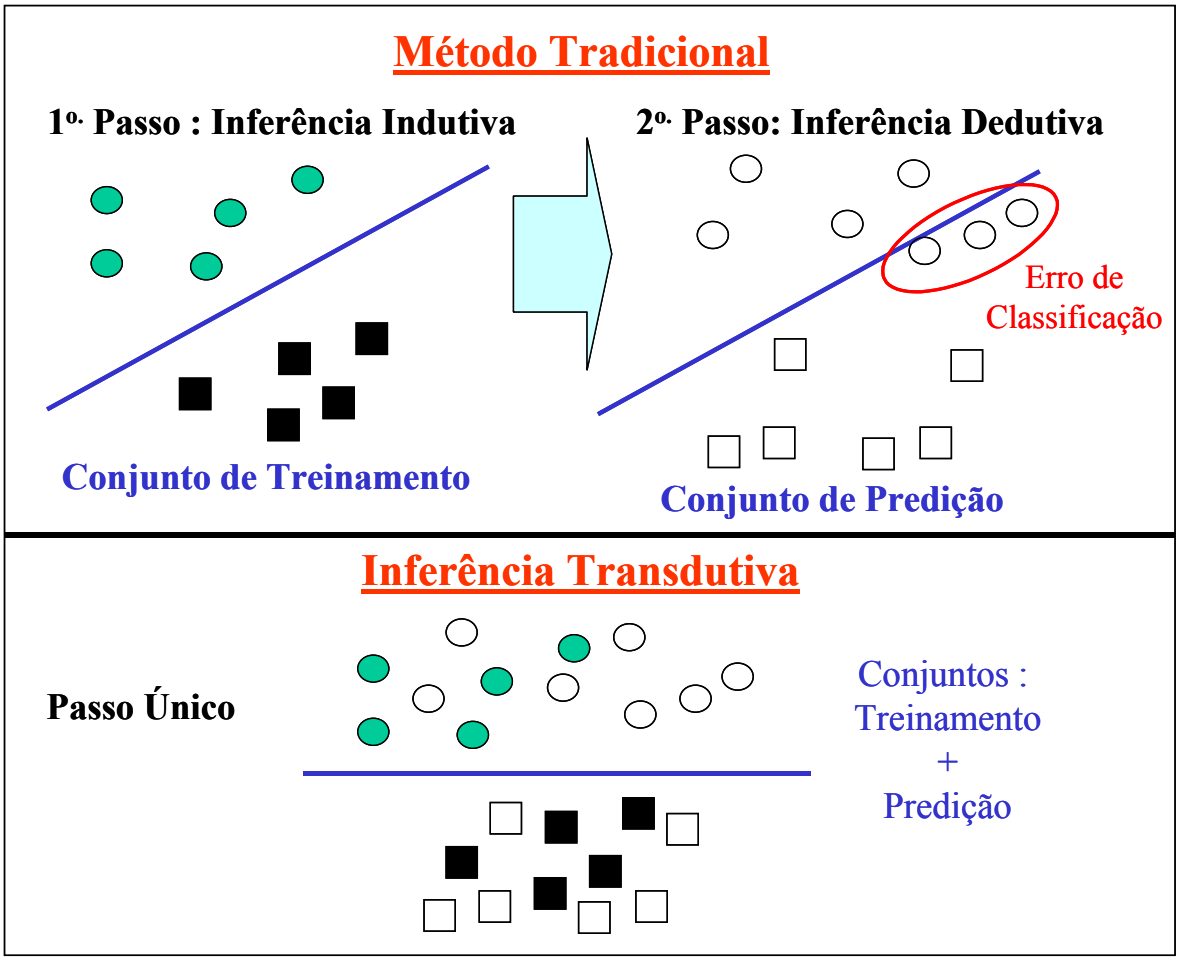


**Figura 6.1** : Diferença do método tradicional (indutivo-dedutivo) para a inferência transdutiva. No método tradicional, utilizando apenas o conjunto de treinamento, o hiperplano é construído para separar as duas classes (bolas verdes x quadrados pretos). Num segundo passo, este hiperplano é utilizado para deduzir (predizer) a classificação das amostras do conjunto de predição. Como ilustração, é mostrada a classe desconhecida a que pertencem as amostras do conjunto de predição, obedecendo à seguinte notação: bolas brancas pertencem à classe bola verde, enquanto que quadrados brancos pertencem à classe quadrado preto. Verifica-se que houve erro de classificação. Empregando os princípios da inferência transdutiva, o hiperplano é encontrado utilizando no treinamento os dois conjuntos de dados, treinamento e predição, sendo que os dados de predição ainda não estão classificados. Assim a classificação dos dados do conjunto de predição é feita em um único passo. Observa-se que agregando os dados do conjunto de predição ao treinamento do algoritmo, consegue-se um melhor desempenho, embora erros de classificação possam ainda existir (não foi o caso aqui).

Através da inferência indutiva, o aprendizado do classificador irá induzir a função de decisão de tal modo a minimizar a freqüência do erro no conjunto de treinamento. Porém, na maioria dos casos, o interesse não está diretamente ligado à escolha da função de decisão, e sim na classificação do conjunto de predição com o menor erro possível, o que constitui o princípio da inferência transdutiva.

Os conceitos da inferência transdutiva são ainda mais úteis quando o conjunto de treinamento consiste em apenas um pequeno número de amostras já classificadas e pertencentes a um espaço de alta dimensão. Para este caso, a solução do problema via um classificador que emprega o método indutivo vai estar demasiadamente suscetível ao sobreajuste (*overfitting*) dos dados do conjunto de treinamento. A idéia principal é explorar os dados ainda não classificados para gerar informação adicional sobre o problema, a qual será utilizada com o propósito de melhorar a generalização e aumentar o desempenho do classificador.

Um exemplo de como o conjunto de predição, ainda não classificado ou rotulado, consegue gerar informação adicional sobre o problema é ilustrado na Figura 6.2:

| Amostra | Atributos | | | | | | Conjunto | Classe |
|---|---|---|---|---|---|---|---|---|
| | c1 | c2 | c3 | c4 | c5 | c6 | | |
| A1 | 1 | 1 | 0 | 0 | 0 | 0 | treinamento | +1 |
| A2 | 1 | 0 | 1 | 0 | 0 | 0 | predição | +1 |
| A3 | 0 | 0 | 1 | 0 | 0 | 0 | predição | +1 |
| A4 | 0 | 0 | 0 | 1 | 0 | 0 | predição | -1 |
| A5 | 0 | 0 | 0 | 1 | 1 | 1 | predição | -1 |
| A6 | 0 | 0 | 0 | 0 | 0 | 1 | treinamento | -1 |

**Figura 6.2** : As amostras 1, 2 e 3 pertencem à classe positiva e as amostras 4, 5 e 6 pertencem à classe negativa. O conjunto de treinamento é formado pelas amostras 1 e 6, e o conjunto de predição é formado pelas amostras 2 a 5. Os atributos c1 ao c6 são variáveis fictícias onde o 1 significa a presença de determinada característica e o 0 a ausência da característica.

Pela Figura 6.2, observa-se que a amostra A2 apresenta o atributo c1 semelhante à amostra de treinamento A1, assim pode ser classificada como pertencente à classe positiva. Semelhante raciocínio pode ser feito com a amostra A5 com relação à amostra de treinamento A6. Desta forma, é repetido o que o treinamento indutivo faria de forma dedutiva nas amostras de predição.

E como ficariam classificadas as amostras A3 e A4? Comparando-as com as amostras A2 e A5, observa-se que elas apresentam, respectivamente, os atributos c3 e c4 semelhantes, por isto

podem ser classificadas como: A3 na classe positiva e A4 na classe negativa. Agora, é feito o que o princípio da inferência dedutiva faria tendo os dois conjuntos de dados sendo utilizados no treinamento do classificador.

Há muitas aplicações práticas em que os dados não classificados são muitos e a quantidade de dados previamente classificados é pequena. Isto pode acontecer pelo motivo da classificação dos mesmos ser muito custosa, difícil ou demorada de ser obtida. A seguir, são citadas algumas possíveis áreas de aplicação para os princípios da inferência transdutiva:

- Medicina: diagnósticos médicos e pesquisas para o desenvolvimento de novas drogas;
- Bioinformática: análise de dados de expressão gênica;
- Classificação de textos;
- *Database Marketing*: prospecção de clientes para novos produtos;
- Mercado Financeiro: previsão de inadimplência;
- Mercado Segurador: previsão de sinistro.

## 6.3 Aspectos da Teoria do Aprendizado Estatístico

Será formulado nesta seção o problema de estimação de valores de uma função para um grupo de pontos de interesse utilizando o método chamado de minimização da função-risco global, proposto por Vapnik (1998).

Seja a função distribuição de probabilidade P($x,y$) aplicada ao conjunto de pares ($X,Y$), e considere o conjunto independente e identicamente distribuído, contendo $N+K$ vetores, selecionados aleatoriamente deste conjunto de pares ($X,Y$):

$$\{x_1, x_2, ..., x_N, x_{N+1}, ..., x_{N+K}\}. \quad \textbf{(6.1)}$$

Existe uma função $y = \phi(x)$ que assinala um valor $y$ para cada vetor $x$ pertencente ao conjunto (6.1). Assim, para cada elemento do conjunto (6.1) temos os valores correspondentes na forma:

$$\{y_1, y_2, ..., y_N, y_{N+1}, ..., y_{N+K}\}. \quad \textbf{(6.2)}$$

Aleatoriamente, $N$ vetores são selecionados do conjunto (6.1), para os quais os correspondentes valores de $\phi(.)$ são indicados, formando o conjunto de pares:

$$\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}. \quad \textbf{(6.3)}$$

Os pares (6.3) formarão o tradicional conjunto de treinamento.

O conjunto de predição será formado pelos vetores remanescentes:

$$\{x_{N+1}, x_{N+2}, \ldots, x_{N+K}\}. \quad \textbf{(6.4)}$$

É preciso obter um algoritmo $M$ que, baseado no conjunto de pares de treinamento (6.3) e no conjunto de predição (6.4), irá estimar os valores da função $\phi(x)$ nos pontos de interesse (6.4) através da função:

$$F(x,w_M) = f(x,w_M(x_1, y_1; \ldots; x_N, y_N; x_{N+1}, \ldots, x_K)) ,$$

onde $w_M$ pertence a um conjunto $W$.

Esta função $F(x,w_M)$ deverá minimizar o valor da função-risco:

$$R(w_M) = \int \frac{1}{K} \sum_{i=N+1}^{N+K} L(y_i, F(x, w_M)) \;\; dP(x_1, y_1)...dP(x_{N+K}, y_{N+K}) ,$$

onde $L(y_i , F(x,w_M))$ é a função de perda que mede a discrepância entre $y$ e $F(x,w)$. Podendo ser, por exemplo, a função de perda quadrática ou simplesmente uma função indicadora que recebe o valor 0, quando $y = F(x,w)$, e 1, caso contrário.

Baseado nos elementos dos conjuntos de treinamento e de predição, e no conjunto pré-definido de funções $F(x,w)$, é preciso encontrar a função $F(x,w^*)$ que minimiza com probabilidade pré-fixada 1-$\alpha$ (conforme definido na Seção 3.4.3), o risco global de predizer os valores da função $\hat{y} = F(x,w)$ nos elementos do conjunto de predição, ou seja, o valor da função-risco global no conjunto de predição, definida como:

$$R_{pred}(w) = \frac{1}{K} \sum_{i=N+1}^{N+K} L(y_i, F(x,w)) .$$

Utilizando o método de minimização do risco estrutural, constrói-se limitantes na função-risco global, uniformemente sobre a classe de funções $F(x,w)$, baseado nos valores da função-risco empírico, de maneira análoga àquela utilizada na Seção 3.4.3.

Vapnik (1998) demonstrou que a função-risco global no conjunto de predição, para um conjunto de funções indicadoras com dimensão VC $h$, e com probabilidade pré-fixada 1-$\alpha$, é limitada por :

$$R_{pred}(w) \leq R_{emp}(w) + \varepsilon(N,K,h,\alpha) , \quad \textbf{(6.5)}$$

onde :

- $R_{emp}(w)$ é o risco empírico definido somente pelo conjunto de treinamento (equação (3.11));

- $\varepsilon\,(N,K,h,\alpha)$ é o limitante superior de confiança que depende da cardinalidade dos conjuntos de treinamento $N$ e de predição $K$, da dimensão VC $h$ e da probabilidade $\alpha$. Pela complexidade da demonstração deste limitante, optou-se por apenas mencioná-lo. Maiores detalhes podem ser encontrados em Vapnik (1998).

Considere o conjunto de funções classificadoras $F(x,w)$:

$$\mathscr{F} = \{ F(x,w): w \in W_j \} \quad , \; j = 1,2,..., q$$

definindo a seguinte estrutura:

$$\mathscr{F}_1 \subset \mathscr{F}_2 \subset \ldots \subset \mathscr{F}_{q-1} \subset \mathscr{F}_q .$$

É possível minimizar o lado direito da equação (6.5) encontrando um elemento $\mathscr{F}$, e uma função $F(x,w^*)$ para o qual a garantia do mínimo para o limitante da função-risco global é alcançada. Utilizando esta função $F(x,w^*_{emp})$, os valores de $\hat{y}$ são calculados para os elementos do conjunto de predição. Aparentemente, este esquema não difere em nada do método indutivo considerado na Seção 3.4.3 para a estimação de funções.

Entretanto, neste esquema de minimização do risco global estrutural, uma característica especial determina a diferença entre as soluções de estimação da função e aquela de estimação de valores da função para um grupo de pontos de interesse.

Para o problema de estimação de funções, é suficiente saber a classe de funções $F(x,w)$, $w \in W$, e o domínio de definição da função, para definir a estrutura sobre $F(x,w)$.

Para o problema de estimação de valores da função, precisa-se determinar a estrutura sobre $F(x,w)$ sabendo o conjunto de funções e o conjunto completo de dados (6.1).

Com isto, a inferência transdutiva obtém limitantes para o erro de predição melhores do que os limitantes obtidos pelos princípios da inferência indutiva. Como conseqüência, o método transdutivo deve apresentar uma capacidade maior de generalização.

## 6.4 Support Vector Machines associada à Inferência Transdutiva

Nesta seção, será demonstrado como a formulação padrão da técnica SVM pode ser generalizada para atender os princípios da inferência transdutiva.

Considere o seguinte problema: dado o conjunto de treinamento

$$\{(x_1, y_1), ..., (x_N, y_N)\} \qquad x \in R^m , \;\; y \in \{+1, -1\}, \tag{6.6}$$

e os dados de predição

$$\{ x_1^*,...,x_K^* \}, \qquad x^* \in R^m , \tag{6.7}$$

encontre o conjunto de funções lineares $y = (w^T x)+b$ que minimize o número de erros de classificação no conjunto de predição.

A melhor solução para este problema é fornecer a classificação para o conjunto de predição

$$\{ y_1^*,...,y_K^* \}, \qquad y^* \in \{+1, -1\} , \quad i = 1, ..., K, \tag{6.8}$$

de tal maneira que a seqüência completa

$$\left\{(x_1, y_1),...,(x_N, y_N),(x_1^*, y_1^*),...,(x_K^*, y_K^*)\right\} \tag{6.9}$$

tenha as duas classes (+1, −1) separadas com a máxima margem.

Por essa razão, deseja-se encontrar a classificação (6.8) para o conjunto das amostras de predição (6.7) para o qual o hiperplano ótimo

$$y = (w_0^T x) + b_0$$

maximize a margem de separação ao separar os dados (6.9) em duas classes. Aqui, o hiperplano ótimo é definido de modo que o conjunto de predição (6.7) é classificado de acordo com (6.8).

Portanto, o objetivo é encontrar as classificações (6.8) para as quais as seguintes desigualdades são validas:

$$y_i \left[(w^T x_i) + b\right] \geq 1, \qquad i = 1,..., N \ ; \tag{6.10}$$

$$y_j^* \left[(w^T x_j^*) + b\right] \geq 1, \qquad j = 1,..., K \ . \tag{6.11}$$

Assim, o problema de otimização, em sua representação primal, para encontrar o hiperplano ótimo para classes linearmente separáveis, utilizando os princípios da inferência transdutiva, é formulado como segue:

*A partir dos dados de treinamento linearmente separáveis* $( x_i , y_i )_{1 \leq i \leq N}$ , $x \in R^m$, $y \in \{+1, -1\}$, *onde x são os dados de entrada e y corresponde à resposta desejada, e dos dados de predição* $(x_j^*)_{1 \leq j \leq K}$ , *encontre o valor do vetor de pesos w, intercepto b e os valores da classificação do conjunto de predição* $(y_j^*)_{1 \leq j \leq K}$, $y^* \in \{+1, -1\}$, *que resolvem o seguinte problema :*

$$\begin{aligned} &\textit{Minimizar}: \quad V(w,b,y^*) = \frac{1}{2} \| w \|^2 . \\ &\textit{Sujeito a}: \quad \forall_{i=1}^{N}: \; y_i\left[w^T x_i + b\right] \geq 1; \\ &\qquad\qquad \forall_{j=1}^{K}: \; y_j^*\left[w^T x_j^* + b\right] \geq 1. \end{aligned} \tag{6.12}$$

Para o caso mais geral, de encontrar o hiperplano ótimo para classes não linearmente separáveis, o problema de otimização primal, utilizando os princípios da inferência transdutiva, é formulado como a seguir:

*A partir dos dados de treinamento* $(x_i, y_i)_{1 \leq i \leq N}$, $x \in R^m$, $y \in \{+1, -1\}$, *e dos dados de predição* $(x_j^*)_{1 \leq j \leq K}$, *encontre o valor do vetor de pesos w, intercepto b, variáveis de folga* $(\xi_i)_{1 \leq i \leq N}$, $(\xi_j^*)_{1 \leq j \leq K}$ *e os valores da classificação do conjunto de predição* $(y_j^*)_{1 \leq j \leq K}$, $y^* \in \{+1, -1\}$ *que resolvem o seguinte problema:*

$$\begin{aligned} &\textit{Minimizar}: \quad V(w,b,\xi,\xi^*,y^*) = \frac{1}{2} \| w \|^2 + C\sum_{i=1}^{N} \xi_i + C^* \sum_{j=1}^{K} \xi_j^*. \\ &\textit{Sujeito a}: \quad \forall_{i=1}^{N}: \; y_i\left[w^T x_i + b\right] \geq 1 - \xi_i; \\ &\qquad\qquad \forall_{j=1}^{K}: \; y_j^*\left[w^T x_j^* + b\right] \geq 1 - \xi_j^*; \\ &\qquad\qquad \forall_{i=1}^{N}: \; \xi_i \geq 0; \\ &\qquad\qquad \forall_{j=1}^{K}: \; \xi_j^* \geq 0, \end{aligned} \tag{6.13}$$

*onde os parâmetros C >0 e C*>0, são especificados pelo usuário. Estes parâmetros são conhecidos como parâmetros de penalização, e controlam a relação entre a complexidade do algoritmo e o número de amostras de treinamento classificadas incorretamente.*

Finalmente, formulamos o problema de otimização dual para o caso geral em que o hiperplano ótimo é construído no espaço característico, através de um mapeamento não-linear definido implicitamente por um produto interno kernel escolhido a priori.

*A partir dos dados de treinamento* $(x_i, y_i)_{1 \le i \le N}$, $x \in R^m$ *e* $y \in \{+1, -1\}$, *e dos dados de predição* $(x_j^*)_{1 \le j \le K}$, *e utilizando o espaço característico definido implicitamente por um produto interno kernel K, encontre os multiplicadores de Lagrange* $(\alpha_i)_{1 \le i \le N}$, $(\alpha^*_j)_{1 \le j \le K}$ *e os valores da classificação do conjunto de predição* $(y_j^*)_{1 \le j \le K}$, $y^* \in \{+1, -1\}$, *que resolvem o seguinte problema*:

$$\begin{aligned}
&\textit{Maximizar}: \quad W(\alpha, \alpha^*, y^*) = \sum_{i=1}^{N} \alpha_i + \sum_{j=1}^{K} \alpha_j^* \\
&-\frac{1}{2}\left[ \sum_{i,r=1}^{N} y_i y_r \alpha_i \alpha_r K(x_i, x_r) + \sum_{j,r=1}^{K} \alpha_j^* y_j^* y_r^* \alpha_r^* K(x_j^*, x_r^*) + 2 \sum_{i=1}^{N} \sum_{j=1}^{K} y_i y_j^* \alpha_i \alpha_j^* K(x_i, x_j^*) \right]. \\
&\textit{Sujeito a}: \quad \sum_{i=1}^{N} y_i \alpha_i + \sum_{j=1}^{K} y_j^* \alpha_j^* = 0 \ ; \\
&\qquad \forall_{i=1}^{N} : 0 \le \alpha_i \le C \qquad ; \qquad \forall_{j=1}^{K} : 0 \le \alpha_j^* \le C^*,
\end{aligned} \tag{6.14}$$

*onde os parâmetros C e C* , são especificados pelo usuário.*

A função de decisão do hiperplano dada pela SVM e utilizando os princípios da inferência transdutiva é:

$$y(x) = \operatorname{sgn}\left( \sum_{i=1}^{N_{sv}} y_i \alpha_i K(x_i, x) + \sum_{j=1}^{K_{sv}} y_j^* \alpha_j^* K(x_j^*, x) + b \right) , \tag{6.15}$$

onde $N_{sv}$ é o número de vetores-suporte pertencentes ao conjunto de treinamento, e $K_{sv}$ é o número de vetores-suporte pertencentes ao conjunto de predição.

Porém, a solução exata do problema primal (6.13), ou de seu dual (6.14), requer uma busca sobre todas as $2^K$ possibilidades de classificação do conjunto de predição, visando produzir a SVM com a máxima margem de separação baseada em todo o conjunto de dados $N+K$. Isto pode ser feito apenas para instâncias pequenas do conjunto de predição, de 3 a 7 amostras. Para um número grande de amostras de predição, deve-se utilizar algum procedimento heurístico ou de busca para encontrar uma boa solução, que até pode ser o ótimo do problema (6.13), ou (6.14).

## 6.5 Algoritmo de implementação para SVM Transdutiva (TSVM)

O treinamento da SVM com os princípios da inferência transdutiva (TSVM) conduz a um problema de otimização na forma de um *mixed integer quadratic program*. Em geral, este problema não é convexo e o número de variáveis inteiras é proporcional ao número de amostras do conjunto de predição. Assim, encontrar o ótimo global com os métodos de otimização padrões é possível somente para um número pequeno de amostras de predição.

Por isto, será descrito nesta seção o método para o treinamento da TSVM proposto por Joachims (1999b), que torna o problema transdutivo tratável, mesmo para grandes conjuntos de predição (por exemplo, $K$=10.000), encontrando uma solução aproximada e com convergência garantida.

Este algoritmo, TSVM, também está disponível na Web no site http://svmlight.joachims.org/ . Utilizado principalmente para o problema de classificação de textos, é considerado na literatura como o algoritmo de referência por outros pesquisadores desta área (Demiriz & Bennett, 2000) pelo fato de afrouxar as restrições quanto ao número de amostras de predição e ser o de melhor desempenho quando comparado aos demais algoritmos.

### 6.5.1 Abordagem do Problema

Como já mencionado, o problema de otimização combinatório, utilizando os princípios da inferência transdutiva para treinar a SVM (TSVM), foi formulado na seção anterior, problema de otimização (6.13). Iremos chamá-lo de PO(6.13).

Para um pequeno número de amostras de predição, o PO(6.13) pode ser resolvido otimamente simplesmente considerando todas as $2^K$ possibilidades combinatórias de classificação (6.8) para o conjunto de predição (6.7), e para cada uma destas possibilidades treinando a técnica SVM indutiva descrita no Capítulo 5. A combinação que encontrar o mínimo seria a ótima. Porém, esta abordagem torna-se intratável para conjuntos de predição com mais de 7 amostras.

Abordagens prévias utilizando o método de pesquisa branch-and-bound baseado nos algoritmos propostos em outros contextos por Vapnik & Sterin (1977) e Wapnik & Tscherwonenkis (1979) (trata-se de Vapnik & Chervonenkis, porém respeitou-se o artigo original) conseguiram elevar o número de amostras do conjunto de predição para não mais do que 100 amostras, além do que se perdia a tratabilidade computacional. Uma vez que os princípios da inferência transdutiva são cada vez mais úteis para grandes conjuntos de predição, esta

abordagem utilizando o método de pesquisa branch-and-bound tornou-se inapropriada para o treinamento da TSVM.

O algoritmo TSVM proposto por Joachims (1999b) é elaborado para trabalhar com conjuntos de predição que podem ter mais de 10.000 amostras. Este algoritmo encontra uma solução aproximada para o PO(6.13) utilizando uma forma de busca local.

Algoritmos de busca local começam com alguma instância inicial das variáveis e a cada iteração a instância corrente das variáveis é modificada no sentido de se mover em direção à solução ótima. Este processo iterativo é executado até que a melhora não seja mais possível.

Num primeiro momento, o caminho mais óbvio para aplicar este algoritmo de busca local é resolver o PO(6.13) da seguinte maneira: comece com alguma classificação para o conjunto de predição. Depois mude a classificação de alguma amostra do conjunto de predição a cada iteração, de maneira que o tamanho da margem de separação aumente.

Entretanto, num segundo momento surgem dois problemas com esta abordagem:

- Não há critério suficientemente óbvio para selecionar quais amostras irão mudar suas classificações a cada iteração. O encontro de uma amostra de maneira que a troca de sua classificação aumente a margem de separação requer efetuar a atualização e re-treinar a SVM. Isto torna o processo de seleção muito custoso;
- Existem muitos mínimos locais, que fazem com que o processo de pesquisa algumas vezes fique preso em um deles, não encontrando desta forma o mínimo global.

O algoritmo que será apresentado nesta seção (TSVM) ameniza estes problemas utilizando uma aproximação suave para a função-objetivo do PO(6.13). Esta aproximação é gradativamente diminuída conforme o processo de otimização progride, até que esta aproximação seja idêntica à função-objetivo original.

Empiricamente, a convergência do algoritmo sempre produziu soluções muito próximas do mínimo global, não apresentando casos em que a convergência se deu para mínimos locais longe do ótimo global (Joachims, 1999b).

### 6.5.2 O Algoritmo TSVM

O algoritmo TSVM é sumarizado na Figura 6.3 para o caso linear, sem a utilização de produto interno kernel, pois permite a apresentação do algoritmo tendo os problemas de

otimização em sua representação primal, tornando mais fácil a sua apresentação. A generalização para adaptá-lo ao uso de produtos internos kernel é direta.

O algoritmo TSVM (Figura 6.3) começa no passo 1 com o treinamento da SVM indutiva junto aos dados de treinamento. A classificação inicial no conjunto de predição é baseada na estratégia do método indutivo. As *num*+ (número de amostras de predição a serem classificadas na classe positiva) amostras de predição com os maiores valores da função de decisão $w^T x_j + b$ são classificadas na classe positiva ($y_j^* = +1$). As amostras remanescentes são classificadas na classe negativa ($y_j^* = -1$).

No passo 2, são fixados os valores dos parâmetros de custo $C^*_-$ (ajuste das amostras de predição da classe negativa na iteração corrente) e $C^*_+$ (ajuste das amostras de predição da classe positiva na iteração corrente) Estes dois parâmetros têm a finalidade do ajuste pelo algoritmo TSVM do parâmetro *num*+, inicialmente especificado pelo usuário.

A recursão 1 aumenta gradativamente, a cada iteração, a influência do conjunto de predição no treinamento da SVM, pelo aumento do valor dos parâmetros de custos $C^*_-$ e $C^*_+$ até o parâmetro do usuário $C$. Isto pode ser visto como uma lenta mudança entre o pleno treinamento indutivo ($C^*_- = C^*_+ = 0$) e o pleno treinamento transdutivo ($C^*_- = C^*_+ = C$).

A recursão 2 melhora iterativamente a solução do problema de otimização pela troca da classificação de um par de amostras de predição com classificações diferentes. O critério de seleção para a troca (passo 3.2) identifica duas amostras em que a troca de classificação na iteração corrente levará a um decréscimo da função objetivo do PO(6.16).

A função "resolve PO(6.16)" é utilizada muitas vezes no algoritmo (passos 1, 3.1 e 3.2.2). Ela se refere à resolução do seguinte problema de otimização primal :

$$\begin{aligned}
&\textit{Minimizar}: \quad V(w,b,\xi,\xi^*) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i + C^*_- \sum_{j:y_j^*=+1}\xi_j^* + C^*_+ \sum_{j:y_j^*=-1}\xi_j^*. \\
&\textit{Sujeito a}: \quad \forall_{i=1}^{N}: \quad y_i\left[w^T x_i + b\right] \geq 1-\xi_i; \\
&\qquad\qquad\quad \forall_{j=1}^{K}: \quad y_j^*\left[w^T x_j^* + b\right] \geq 1-\xi_j^*; \\
&\qquad\qquad\quad \forall_{i=1}^{N}: \quad \xi_i \geq 0; \\
&\qquad\qquad\quad \forall_{j=1}^{K}: \quad \xi_j^* \geq 0.
\end{aligned} \tag{6.16}$$

## ***Algoritmo TSVM***

Entrada : $(x_1, y_1), ..., (x_N, y_N)$ // conjunto de pares de treinamento

$x_1^*,...,x_K^*$ // conjunto de predição

Parâmetros do usuário : $C$ // parâmetro do PO(6.16)

$num+$ // número de amostras de predição a serem classificadas na classe positiva

**1)** $(w,b,\xi,\_)$ = resolve PO(6.16) $([(x_1, y_1)...(x_N, y_N)],[],C,0,0)$

$y_1^*,...,y_K^* = f(x_1^*,...,x_K^*, w, b, num+)$

// classifica o conjunto de predição utilizando $w$ e $b$. As $num+$ amostras de predição com os maiores valores da função de decisão $w^T x_j + b$ são classificadas na classe positiva ($y_j^* = +1$). As amostras remanescentes são classificadas na classe negativa ($y_j^* = -1$).

**2)** $C^*_- = 10^{-5}$ // algum número pequeno

$C^*_+ = 10^{-5} \times [num+ /(K - num+)]$

**// Recursão 1**

**3)** Enquanto $((C^*_- < C) \,||\, (C^*_+ < C))$ faça :

**3.1)** $(w,b,\xi,\xi^*)$ = resolve PO(6.16) $([(x_1, y_1)...(x_N, y_N)],[(x_1^*, y_1^*)...(x_K^*, y_K^*)],C, C^*_-, C^*_+)$

**// Recursão 2**

**3.2)** Enquanto $(\exists\, m,l : (y_m^* \times y_l^* < 0) \,\&\, (\xi_m^* > 0) \,\&\, (\xi_l^* > 0) \,\&\, (\xi_m^* + \xi_l^* > 2))$ faça :

**3.2.1)** $y_m^* = -y_m^*$ // troca da classificação das duas amostras, uma positiva e outra negativa

$y_l^* = -y_l^*$

**3.2.2)** $(w,b,\xi,\xi^*)$ = resolve PO(6.16) $([(x_1, y_1)...(x_N, y_N)],[(x_1^*, y_1^*)...(x_K^*, y_K^*)], C, C^*_-, C^*_+)$

**3.3)** $C^*_- = \min(2 \times C^*_-, C)$

$C^*_+ = \min(2 \times C^*_+, C)$

**4)** Saída : $y_1^*,...,y_K^*$ // classificação das amostras do conjunto de predição

**Figura 6.3** : Algoritmo de treinamento da support vector machines transdutiva proposto por Joachims (1999b)

O PO(6.16) é similar à SVM indutiva, sendo resolvido utilizando sua representação dual pelo algoritmo SVM$^{light}$ descrito no Capítulo 5.

Nas recursões 1 e 2, o PO(6.16) é resolvido utilizando a solução corrente como ponto inicial.

Para a chamada do PO(6.16) na recursão 2, o problema de otimização corrente difere do anterior apenas pela troca de classificação de duas amostras de predição. Por isto, a solução do problema de otimização não requer muito tempo de processamento. Raciocínio similar também é válido para a chamada do PO(6.16) na recursão 1, em que somente os parâmetros de custo $C^*_-$ e $C^*_+$ são atualizados.

A convergência do algoritmo TSVM é demonstrada formalmente, não havendo possibilidade de entrada em ciclos. Porém a convergência para o ótimo global não é garantida teoricamente. Mas empiricamente, o algoritmo sempre apresenta respostas próximas do ótimo global, evitando a convergência para mínimos locais longe do ótimo global (Joachims, 1999b).

### 6.5.3 Análise do funcionamento do algoritmo TSVM

Será apresentada nesta seção uma interpretação do mecanismo de funcionamento intuitivo do algoritmo TSVM. A Figura 6.4 mostra este mecanismo.

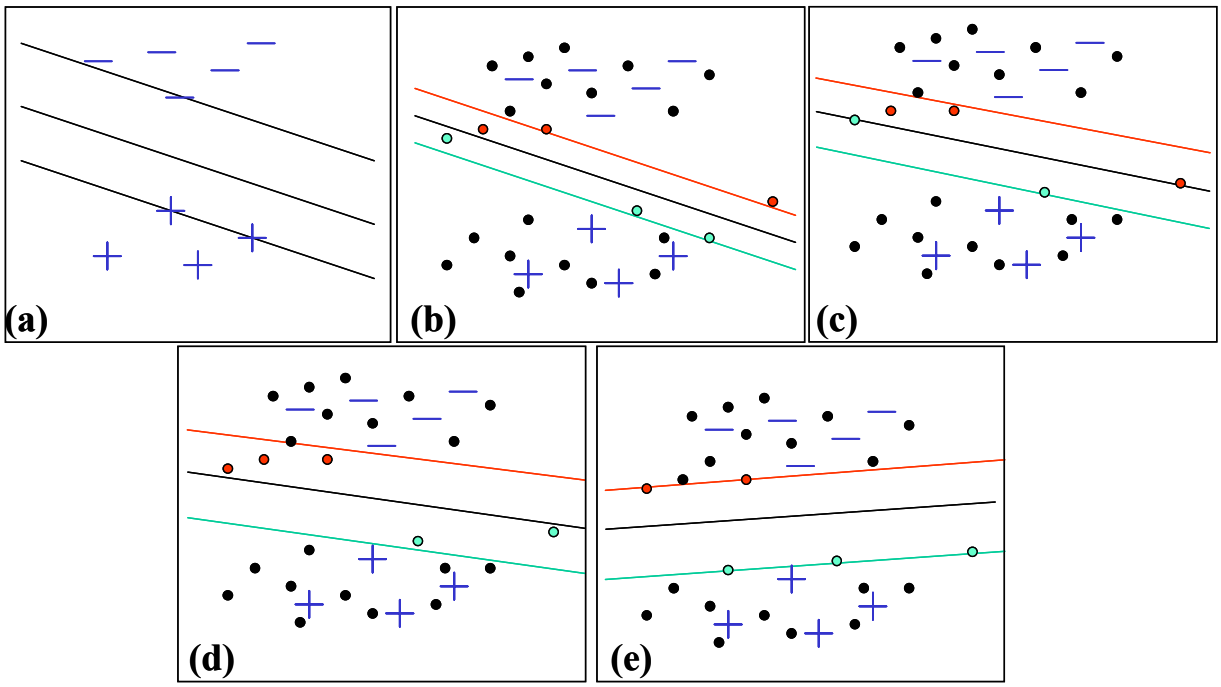


**Figura 6.4** : Exemplo intuitivo para ilustrar o funcionamento do algoritmo TSVM

Na Figura 6.4 (a) é mostrado o conjunto de treinamento, onde os sinais negativos representam as amostras de treinamento pertencentes à classe negativa, e os sinais positivos representam as amostras pertencentes à classe positiva. O hiperplano encontrado pela abordagem indutiva, utilizando SVM, é aquele que maximiza a margem de separação no conjunto de dados de treinamento.

Classificando o conjunto de predição com o hiperplano encontrado pela SVM indutiva, obtém-se a distribuição dos dados mostrada na Figura 6.4 (b). Passo 1 do algoritmo TSVM (Figura 6.3). As amostras de predição do lado da margem verde são classificadas como pertencentes à classe positiva, e as amostras do lado da margem vermelha são classificadas como pertencentes à classe negativa. As amostras de predição pintadas em verde e vermelho exercem uma força de atração (em uma interpretação física) em suas respectivas margem de separação.

O algoritmo TSVM entra na recursão 1, e com isto os valores dos parâmetros de custo, para o conjunto de predição, são incrementados fazendo com que as amostras de predição exerçam uma maior influência no treinamento da SVM, e com isto o hiperplano começa a executar uma rotação no sentido anti-horário, como é ilustrado na Figura 6.4 (c).

A rotação continua até que uma amostra de predição de cada classe ultrapasse a superfície de decisão. Quando isto acontece, as condições para a entrada na recursão 2 são satisfeitas, e as amostras trocam de classificação, como mostrado nas Figuras 6.4 (c) e (d).

O processo continua iterativamente até que não haja mais nenhuma amostra de predição para abandonar a região marginal (Figura 6.4 (e)) e assim a margem encontrada é a máxima, ou para os casos em que os dados são não linearmente separáveis, o limite do parâmetro $C$ é satisfeito.

### 6.5.4 Outras maneiras de implementação da Inferência Transdutiva aplicada a SVM

O primeiro algoritmo para SVM transdutiva, como já mencionado anteriormente, foi baseado nos algoritmos propostos em outros contextos por Vapnik & Sterin (1977) e por Wapnik & Tscherwonenkis (1979). Este algoritmo era baseado no método de pesquisa branch-and-bound. Porém não suportava mais do que 100 amostras de predição.

Depois de um longo período, Bennett & Demiriz (1998) propuseram uma abordagem baseada em uma modificação do problema para tornar sua resolução mais fácil. Ao invés de utilizarem a minimização da norma-2 do vetor de pesos da função-objetivo da SVM (equação

6.13), elas utilizaram a norma-1. Com isto, o problema foi transformado em um "*linear*" *mixed interger program*. Este problema foi resolvido utilizando o software de otimização CPLEX. Porém, este algoritmo é limitado para no máximo 70 amostras de predição.

Outra abordagem utilizando a mesma estratégia da norma-1 de Bennett & Demiriz (1998) foi proposta por Fung & Mangasarian (1999). Ao invés de utilizarem a formulação *mixed interger*, eles utilizaram um algoritmo de programação linear repetida. Com esta modificação, eles conseguiram aumentar um pouco o limite máximo do número de amostras de predição suportável para tornar o treinamento do problema tratável.

O algoritmo TSVM (Joachims, 1999b) descrito neste capítulo é considerado na literatura como de referência, justamente por tratar de problemas com mais de 10.000 amostras de predição. Além disto, é um algoritmo que tem apresentado convergência para uma solução sempre perto do ótimo global em um número reduzido de iterações, como já mencionado nas seções anteriores e como será verificado nas aplicações a serem apresentadas no próximo capítulo.

# Capítulo 7

# Aplicações

## 7.1 Aplicação 1: "Support Vector Machines Transdutiva para o Diagnóstico de Câncer e Classificação de Dados de Expressão Gênica"

### 7.1.1 Motivação

Esta é a primeira aplicação dos princípios da inferência transdutiva a problemas de Bioinformática (Semolini & Von Zuben, 2002), particularmente no contexto de classificação utilizando dados de expressão gênica, para os seguintes problemas:

- Classificação de categorias de diagnóstico de Câncer;
- Classificação de dados de expressão gênica da levedura de brotamento *Saccharomyces cerevisiae* em grupos funcionais.

A tarefa de classificação será realizada por meio dos princípios da inferência transdutiva em conjunto com a técnica SVM (TSVM), como descrito no Capítulo 6.

O treinamento da TSVM será baseado nos conjuntos de treinamento e predição. No caso do conjunto de treinamento, especialistas classificaram previamente os genes ou diagnósticos de câncer em suas correspondentes classes. Assim, dadas as amostras do conjunto de predição, o objetivo é associá-las em suas respectivas classes. Através da TSVM, poderemos classificar em um único passo cada amostra do conjunto de predição como pertencente ou não a uma determinada classe.

A técnica TSVM será comparada com a técnica tradicional, SVM indutiva, em uma série de experimentos.

Serão apresentados também resultados comprovando a existência de correlação entre a complexidade do problema de classificação, a porcentagem de amostras vetores-suporte e a evolução da diferença entre o desempenho do método transdutivo comparado ao indutivo.

### 7.1.2 Introdução

A tecnologia associada com *DNA microarray hibridization* analisa o mecanismo molecular da célula, capturando os níveis de expressão do RNA de milhares de genes instantaneamente, e permitindo aos biólogos formularem modelos de expressão gênica em uma escala nunca antes alcançada. Maiores detalhes desta tecnologia podem ser encontrados em Brown *et al*. (2000) e Pavlidis *et al*. (2001).

A tarefa de classificar genes em classes funcionais, utilizando dados de *microarray*, apoia-se no pressuposto de que genes com funções similares apresentam perfis similares de expressão através de um grande número de condições experimentais. Se este pressuposto é ou não válido, depende da categoria funcional que está sendo estudada. Kohane (2002) alerta para o fato de que se deve ter cuidado com esta pressuposição. Todavia, este é um ponto de partida conveniente, pelo menos no caso de análises preliminares.

Importantes resultados sobre classificação utilizando dados de *microarray* foram proporcionados por algoritmos de aprendizado não-supervisionado, como agrupamento hierárquico (Eisen *et al*., 1998) e mapas auto-organizáveis de Kohonen (Tamayo *et al*., 1999). Estes algoritmos agrupam os genes que apresentam padrões similares de expressão, que podem estar associados às mesmas funcionalidades.

Quando se começou a entender melhor a estrutura dos dados de *microarray*, os algoritmos de aprendizado supervisionado, como SVM, ganharam aplicabilidade graças à existência de dados previamente classificados em suas respectivas classes funcionais (Brown *et al*.,2000; Terrence *et al*., 2000; Pavlidis *et al*., 2001; Yeo & Poggio, 2001; Lee & Lee, 2002).

Por causa da maioria dos estudos com dados de expressão gênica envolverem somente 10 ou, excepcionalmente 100 amostras, cada uma com milhares de atributos para serem analisados, a TSVM é uma das mais promissoras técnicas para sintetizar a tarefa de classificação.

Aplicaremos a TSVM em dois problemas de Bioinformática com o propósito de classificação utilizando dados de expressão gênica. Nestes problemas, a técnica de classificação utilizará apenas uma pequena amostra de treinamento previamente classificada, e um segundo

conjunto de dados, o de predição, com o objetivo de ter suas amostras classificadas. Devido ao número reduzido de amostras de treinamento pertencentes a um espaço de alta dimensão, a utilização de técnicas baseadas nos princípios da inferência indutiva e dedutiva resultará no overfitting (sobre-ajuste) dos dados. Por esta razão, ao se extrair informação também do conjunto de predição, o que só é possível com a TSVM, pode-se obter uma melhor generalização, adicionando informação ao problema e aumentando o desempenho da técnica de classificação.

Os dois problemas que serão estudados nesta aplicação são:

- Classificação do tumor infantil *small round blue cell* em 4 categorias específicas de diagnóstico de câncer, baseado em seus dados de expressão gênica (Khan *et al*., 2001);
- Classificação dos genes da levedura de brotamento *Saccharomyces cerevisiae* em grupos funcionais utilizando dados de expressão gênica medidos de diferentes experimentos de *DNA microarray hybridization* (Eisen *et al*., 1998).

### 7.1.3 Análise dos Dados e Resultados

#### 7.1.3.1 Tumor infantil *Small Round Blue Cell*

Khan *et al*. (2001) classificou o tumor infantil *small round blue cell* (SRBCTs) em 4 classes: neuroblastoma (NB), rhabdomyosarcoma (RMS), linfoma Burkitt (BL, um subconjunto do linfoma não-Hodgkin) e a família Ewing de tumores (EWS). A classificação foi feita utilizando dados de expressão gênica do cDNA.

O conjunto de dados está disponível na Web no site http://www.nhgri.nih.gov/DIR/Microarray/Supplement/ , sendo composto de 6.567 genes e 88 amostras divididas em:

- NB : 18 amostras;
- RMS : 25 amostras;
- BL : 11 amostras;
- EWS : 29 amostras;
- 2 tecidos musculares normais;
- 3 células (sarcoma, osteosarcoma e prostate carcinoma).

Utilizando uma rede neural artificial, Khan *et al*. (2001) diagnosticaram com sucesso as 4 categorias de tipos de tumor.

Yeo & Poggio (2001) aplicaram, compararam e combinaram os métodos *k-Nearest Neighbor*, *weighted voting* e SVM indutiva, sendo esta última técnica a que apresentou o melhor desempenho.

Lee & Lee (2002) aplicaram a SVM indutiva para classificação de múltiplas classes, para classificar SRBCTs em 4 classes, e mostraram que somente 20 genes, aqueles com os maiores índices no método de seleção de atributos, dentre os 6567 genes, são capazes de classificar corretamente o SRBCTs.

De acordo com Lee & Lee (2002), os níveis de expressão dos genes foram transformados aplicando logaritmo na base 10 e padronizando os 20 vetores de atributos antes de aplicar SVM e TSVM.

Os SRBCTs pertencentes à classe em estudo serão classificados como positivos, e aqueles não pertencentes à classe em estudo serão classificados como negativos.

**Medida de Desempenho**

A medida de desempenho a ser maximizada pelas técnicas SVM e TSVM será a porcentagem de classificação correta no conjunto de dados de predição:

$$\%\ de\ classificação\ correta\ = (FP + FN) / \text{"}qtd.\ total\ de\ amostras\ de\ predição\text{"},$$

onde *FP* é a quantidade de falso positivos e *FN* é a quantidade de falso negativos.

Utilizando esta medida, pode-se comparar o desempenho de classificação de SVM e TSVM.

**Manipulação dos dados para a aplicação da TSVM**

Para avaliar o desempenho da TSVM, foi pressuposto que somente uma pequena parte das 88 amostras do conjunto de dados já estava previamente classificada, formando o conjunto de dados de treinamento. As amostras remanescentes formarão o conjunto de dados de predição. O objetivo desta aplicação consiste em classificar corretamente estas amostras de predição.

A diferença desta aplicação para os trabalhos já realizados com este conjunto de dados por Lee & Lee (2002), Khan *et al*. (2001) e Yeo & Poggio (2001) é que eles utilizaram as 83 amostras (excluíram as 5 amostras não pertencentes às 4 categorias) divididas em dois conjuntos de dados: o de treinamento com 63 amostras, e o de validação composto por 20 amostras, com o objetivo de classificar corretamente as 20 amostras de validação com a metodologia indutiva.

Nesta aplicação, partiremos de um conjunto pequeno de dados de treinamento e aumentaremos progressivamente seu tamanho, para com isto verificar os efeitos no desempenho da TSVM comparada à SVM indutiva (ver Tabela 7.1).

As amostras, selecionadas aleatoriamente, foram divididas de acordo com a distribuição mostrada na Tabela 7.1, onde apenas uma pequena parte das 88 amostras serão consideradas já classificadas, e o restante formará o conjunto de predição, de quantidade fixa de amostras para ser possível a comparação de desempenho das técnicas nos diferentes tamanhos do conjunto de treinamento.

**Tabela 7.1** : Estrutura dos Dados para a aplicação de TSVM e SVM

| | Conjunto de Treinamento | | | Conjunto de Treinamento | | | Conjunto de Treinamento | | | Conjunto de Treinamento | | | Conjunto de Predição | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Classes** | **Pos.** | **Neg.** | **Total** | **Pos.** | **Neg.** | **Total** | **Pos.** | **Neg.** | **Total** | **Pos.** | **Neg.** | **Total** | **Pos.** | **Neg.** | **Total** |
| **EWS** | 3 | 3 | **6** | 5 | 5 | **10** | 7 | 7 | **14** | 9 | 9 | **18** | 20 | 50 | **70** |
| **NB** | 3 | 3 | **6** | 6 | 6 | **12** | 6 | 12 | **18** | 6 | 18 | **24** | 12 | 52 | **64** |
| **BL** | 3 | 3 | **6** | 5 | 5 | **10** | 5 | 10 | **15** | 5 | 15 | **20** | 6 | 62 | **68** |
| **RMS** | 3 | 3 | **6** | 5 | 5 | **10** | 7 | 7 | **14** | 9 | 9 | **18** | 16 | 54 | **70** |

Com esta estrutura de dados, poderemos mensurar as diferenças de desempenho da TSVM comparada à SVM, principalmente no tocante ao acréscimo progressivo do total de amostras nos 4 conjuntos de treinamento, como mostra a Tabela 7.1. Com esta iniciativa, há 16 (4 classes × 4 conj. trein.) diferentes experimentos que serão repetidos 10 vezes, cada um caracterizado por diferentes conjuntos de treinamento e predição, selecionados aleatoriamente. Com isto o número total de experimentos será : 2 × 10 × 16 = 320 (160 para SVM e 160 para TSVM).

**Especificação dos Parâmetros Utilizados para TSVM e SVM**

As técnicas TSVM e SVM foram implementadas com os seguintes produtos internos kernel: função polinomial de graus $d$ = 1, 2 e 3 (ver equação (3.8)), e RBF (ver equação (3.7)) com o parâmetro $\sigma$ igual à mediana da distância euclidiana de cada amostra positiva em relação à amostra negativa mais próxima (Brown *et al*., 2000). As configurações dos produtos internos kernel que produziram os melhores resultados para os experimentos estão apresentadas na Tabela 7.2.

O parâmetro de penalização $C$ (Seção 4.3) foi escolhido experimentalmente, e os que produziram o melhor grau de generalização são mostrados na Tabela 7.2.

**Tabela 7.2** : Valores dos Parâmetros de Penalização e dos Produtos Internos Kernel

| Método | Param. | EWS | NB | BL | RMS |
|---|---|---|---|---|---|
| TSVM | Kernel | RBF $\sigma = 5,4$ | RBF $\sigma = 4,4$ | RBF $\sigma = 5,3$ | RBF $\sigma = 4,2$ |
| | *C* | 2 | 2 | 2 | 2 |
| SVM | Kernel | RBF $\sigma = 5,4$ | RBF $\sigma = 4,4$ | RBF $\sigma = 5,3$ | RBF $\sigma = 4,2$ |
| | *C* | 1 | 1 | 1 | 1 |

Para TSVM, o parâmetro do usuário *num+* (número de amostras de predição a serem classificadas na classe positiva) foi escolhido com o valor igual ao número de amostras positivas presentes no conjunto de dados de predição.

**Resultados**

As simulações foram executadas em um equipamento SunOS 5.6, com 256MB de memória e 167 MHz de CPU. O tempo médio de CPU gasto para cada uma das 160 simulações da TSVM foi de aproximadamente 2s, e menos de 0.5s em média para cada uma das 160 simulações da SVM.

Os resultados dos experimentos (cada um repetido 10 vezes com uma re-amostragem dos conjuntos de dados de treinamento e predição) comparando TSVM e SVM são mostrados na Tabela 7.3 e na Figura 7.1.

De acordo com os resultados da Tabela 7.3, e utilizando o teste estatístico não-paramétrico de Wilcoxon (Siegel, 1956) para amostras pareadas com nível de significância de 5% (0,05), vemos que para todas as 4 classes de diagnósticos, no $1^o$. e $2^o$. conjuntos de treinamento, a metodologia transdutiva (TSVM) foi significativamente superior à metodologia indutiva, com exceção do $1^o$. conj. de treinamento para a classe BL, onde houve uma tendência a ser significativo (p-valor < 0,1).

Estudos anteriores (Joachims, 1999b) mostraram que, com o aumento do conjunto de dados de treinamento, a diferença entre o desempenho das duas metodologias (indutiva e transdutiva) diminui proporcionalmente. Por isto, como já era esperado, não houve diferença significativa entre as 2 metodologias para o $3^o$. e $4^o$. conjuntos de treinamento (Tabela 7.3).

**Tabela 7.3** : Comparação do desempenho das metodologias TSVM e SVM para a classificação do conjunto de predição, medida pela % de classificação correta. São apresentados a média e o desvio padrão para as 10 simulações com amostras diferentes para cada um dos 16 experimentos de TSVM e SVM. Para comparar as duas metodologias, foi utilizado o teste estatístico não-paramétrico de Wilcoxon para amostras pareadas. Para a construção da última coluna, o nível de significância adotado foi de 0,05, onde S=significativo, T= tendência a ser significativo e NS = não significativo

| Classe | Conj. Trein. | | | Conj. Pred. | TSVM | | SVM | | Teste Wilcoxon p-valor | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Pos. | Neg. | | média | desvio | média | desvio | | |
| EWS | 1 | 3 | 3 | 70 | 100 | 0 | 96,1 | 4,9 | 0,004 | S |
| | 2 | 5 | 5 | 70 | 100 | 0 | 98,9 | 0,9 | 0,016 | S |
| | 3 | 7 | 7 | 70 | 99,6 | 1,0 | 99,0 | 0,7 | 0,219 | NS |
| | 4 | 9 | 9 | 70 | 100 | 0 | 98,7 | 0,8 | 0,008 | S |
| NB | 1 | 3 | 3 | 64 | 99,8 | 0,5 | 83,0 | 13,5 | 0,002 | S |
| | 2 | 6 | 6 | 64 | 100 | 0 | 93,4 | 8,7 | 0,008 | S |
| | 3 | 6 | 12 | 64 | 100 | 0 | 99,2 | 1,3 | 0,250 | NS |
| | 4 | 6 | 18 | 64 | 100 | 0 | 100 | 0 | - | NS |
| BL | 1 | 3 | 3 | 68 | 100 | 0 | 93,5 | 17,9 | 0,063 | T |
| | 2 | 5 | 5 | 68 | 100 | 0 | 98,5 | 1,2 | 0,016 | S |
| | 3 | 5 | 10 | 68 | 100 | 0 | 99,7 | 0,6 | 0,500 | NS |
| | 4 | 5 | 15 | 68 | 100 | 0 | 100 | 0 | - | NS |
| RMS | 1 | 3 | 3 | 70 | 98,3 | 2,2 | 89,7 | 13,2 | 0,004 | S |
| | 2 | 5 | 5 | 70 | 99,1 | 1,8 | 92,9 | 6,0 | 0,004 | S |
| | 3 | 7 | 7 | 70 | 98,4 | 2,1 | 97,4 | 2,2 | 0,438 | NS |
| | 4 | 9 | 9 | 70 | 98,1 | 2,0 | 97,4 | 1,5 | 0,250 | NS |

Graficamente (Figura 7.1), notamos o melhor desempenho da TSVM para os conjuntos de treinamento pequenos. Quando o tamanho do conjunto de treinamento aumenta progressivamente, a diferença de desempenho entre as metodologias TSVM e SVM se reduz proporcionalmente.

Para as 4 classes e 4 tamanhos de conjuntos de treinamento, o desempenho da TSVM foi de aproximadamente 100% de classificação correta no conjunto de predição, indicando uma baixa complexidade presente nos dados para a tarefa de classificação. Por isto, com apenas 6 amostras de treinamento a TSVM foi capaz de produzir um desempenho de 100%. Para a SVM alcançar este mesmo desempenho, é necessário mais do que 3 vezes o número de amostras: aproximadamente 20.

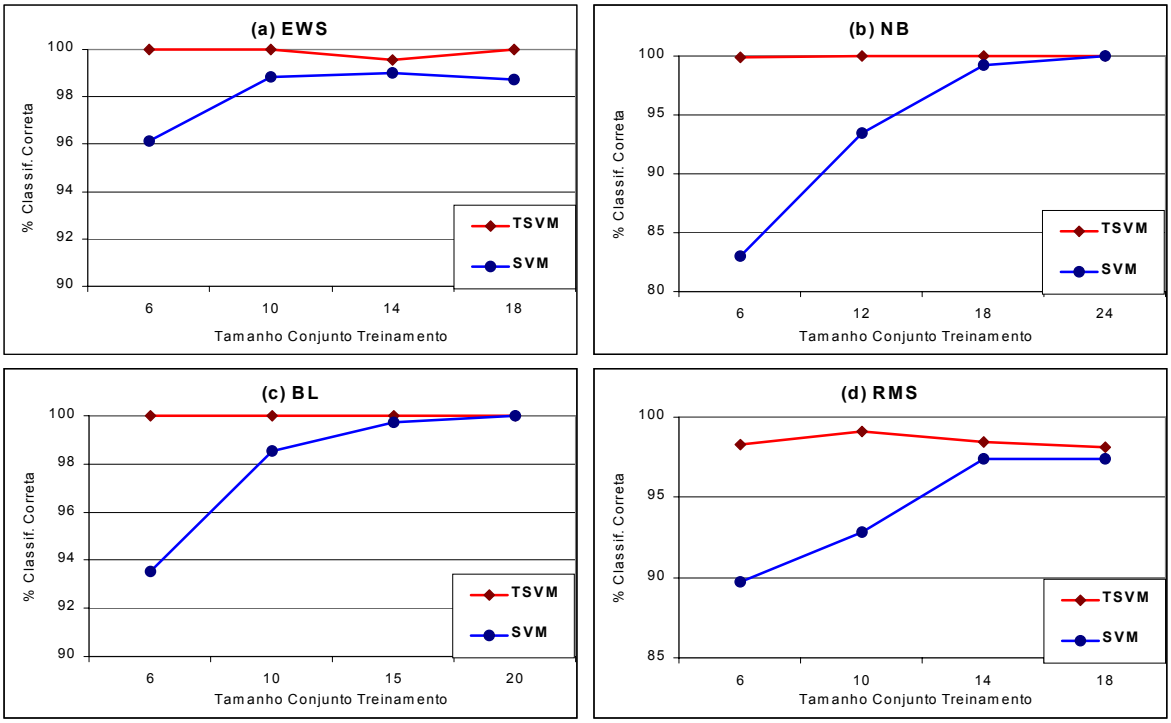


**Figura 7.1** : Comparação gráfica de desempenho das metodologias indutiva (SVM) e transdutiva (TSVM) para as 4 diferentes classes de diagnósticos, através da variação do tamanho do conjunto de treinamento (eixo *x*). Cada ponto representa a média de 10 simulações com diferentes amostragens para cada situação.

### 7.1.3.2 Dados de Expressão Gênica da levedura de brotamento *Saccharomyces cerevisiae*

Serão analisados os dados utilizados nos estudos de Brown *et al*. (2000), disponíveis na Web no site http://www.soe.ucsc.edu/research/compbio/genex . Os dados são compostos pela razão dos níveis de expressão de 2.467 genes da levedura de brotamento *Saccharomyces cerevisiae*, medidos em 79 diferentes experimentos de *DNA microarray hybridization*.

De acordo com Eisen *et al*.(1998), não é aconselhável trabalhar diretamente com a razão dos níveis de expressão, mas sim com o seu logaritmo normalizado:

$$X_i = \frac{\log(E_i / R_i)}{\sqrt{\sum_{i=1}^{79} \log^2 (E_i / R_i)}},$$

onde $E_i$ é o nível de expressão para o gene $X$ em cada um dos 79 experimentos, e $R_i$ é o nível de expressão para o gene $X$ em um estado de referência.

Estes dados foram gerados de um *array* impresso (*spotted array*) estudados durante a *diauxic shift* (transição da fase de consumo de glicose produzindo etanol, para a fase de utilização do etanol para a respiração), o ciclo de divisão celular mitótico, esporulação, temperatura e *reducing shocks*. Os genes foram classificados em 6 classes funcionais: ciclo do ácido tricarboxílico (TCA), respiração (Resp), gene citoplasmático ribosomal (Ribo), proteassomo (Prot), histonas (Hist) e proteína *helix-turn-helix* (domínio da proteína com propriedade de ligar DNA). Esta última classe foi excluída do experimento por ser uma classe de controle nos estudos de Eisen *et al*. (1998). Foi concluído que as outras 5 classes apresentavam similaridades significativas em seus padrões de expressão.

Os genes pertencentes à classe funcional em estudo serão classificados como positivos, e os genes que não pertencem à classe em estudo serão classificados como negativos.

**Medida de Desempenho**

A *função ganho adicional* a ser maximizada no conjunto de predição será medida de acordo com a expressão usada por Brown *et al*. (2000):

$$G = GN - (FP + 2 \times FN) , \qquad \textbf{(7.1)}$$

onde:

- *FP* é o número de falso positivos;
- *FN* é o número de falso negativos;
- *GN* é o ganho do aprendizado nulo que classifica todas as amostras do conjunto de predição como pertencentes à classe negativa. Portanto *GN* é dado por 2 vezes a quantidade de genes positivos do conjunto de predição.

Na equação (7.1) os falso negativos têm um peso maior do que os falso positivos devido ao fato do conjunto de dados (ver Tabela 7.4) apresentar o número de amostras positivas muito menor do que o número de amostras negativas.

Será considerado o índice de complexidade como o complementar da razão da função *G* em relação ao máximo valor possível desta função, obtido quando a classificação é feita sem erros, e assim *G*=*GN*. Por isto o índice de complexidade é dado por:

$$Complexidade = 1 - (G / GN) . \qquad \textbf{(7.2)}$$

Utilizando este índice, será possível comparar a complexidade de classificação das diferentes classes funcionais.

**Manipulação dos dados para a aplicação da TSVM**

Brown *et al*. (2000) demonstraram em seus estudos utilizando este conjunto de dados que a SVM, entre outras técnicas de classificação, é a que apresenta o melhor desempenho para classificação de genes em classes funcionais. A Tabela 7.4 apresenta o conjunto de dados de treinamento e o desempenho (medido utilizando o método de validação cruzada com 3 divisões do conjunto de dados) obtido utilizando a SVM para classificar os genes nas 5 classes funcionais.

**Tabela 7.4** : Conjunto de Dados de Treinamento e Desempenho obtidos por Brown *et al*. (2000) utilizando a SVM indutiva. A coluna *G* é a função ganho (equação (7.1)) obtida por Brown *et al*.. A coluna *Complexidade* foi medida utilizando a equação (7.2).

| Classes | Conj. Treinamento | | | *G* | *Comple-xidade* |
|---|---|---|---|---|---|
| | Pos. | Neg. | Total | | |
| TCA | 17 | 2.450 | **2.467** | 12 | 0,65 |
| Resp | 30 | 2.437 | **2.467** | 39 | 0,35 |
| Prot | 35 | 2.432 | **2.467** | 52 | 0,26 |
| Hist | 11 | 2.456 | **2.467** | 18 | 0,18 |
| Ribo | 121 | 2.346 | **2.467** | 229 | 0,05 |

Para avaliar o desempenho da TSVM, foi pressuposto, de maneira análoga à utilizada na análise dos dados da seção anterior, que somente uma pequena parte das 2.467 amostras do conjunto de dados já estavam previamente classificadas, as quais formarão o conjunto de dados de treinamento. As amostras remanescentes formarão o conjunto de dados de predição. O objetivo desta aplicação consiste em classificar corretamente estas amostras do conjunto de predição.

A diferença desta aplicação para o trabalho já realizado por Brown *et al*. (2000) com este conjunto de dados é que eles utilizaram as 2.467 amostras compondo o conjunto de treinamento, com o objetivo de classificar corretamente as mesmas 2.467 amostras (método de validação cruzada) com a metodologia indutiva. Nesta aplicação, partiremos de um conjunto pequeno de dados de treinamento e aumentaremos progressivamente seu tamanho, para com isto verificar os efeitos no desempenho da TSVM comparada à SVM indutiva (ver Tabela 7.5).

As amostras, selecionadas aleatoriamente, foram divididas de acordo com a distribuição mostrada na Tabela 7.5, onde apenas uma pequena parte das 2.467 amostras serão consideradas já classificadas, e o restante formará o conjunto de predição, de quantidade fixa de amostras para ser possível a comparação de desempenho das técnicas nos diferentes tamanhos do conjunto de treinamento.

O total de amostras da classe positiva nos 4 conjuntos de treinamento foi fixa, sendo 1/3 do total das amostras pertencentes à respectiva classe em todo o conjunto de dados. Esta estratégia foi utilizada devido à pouca quantidade de amostras positivas em cada uma das 5 classes em estudo (ver Tabela 7.4). Já o total de amostras da classe negativa nos 4 conjuntos de treinamento foi variável de acordo com as quantidades mostradas na Tabela 7.5.

**Tabela 7.5** : Estrutura dos Dados para a aplicação de TSVM e SVM

| | Conjunto de Treinamento 1 | | | Conjunto de Treinamento 2 | | | Conjunto de Treinamento 3 | | | Conjunto de Treinamento 4 | | | Conjunto de Predição | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Classes** | **Pos.** | **Neg.** | **Total** | **Pos.** | **Neg.** | **Total** | **Pos.** | **Neg.** | **Total** | **Pos.** | **Neg.** | **Total** | **Pos.** | **Neg.** | **Total** |
| **TCA** | 5 | 5 | **10** | 5 | 15 | **20** | 5 | 50 | **55** | 5 | 200 | **205** | 12 | 2.250 | **2.262** |
| **Resp** | 10 | 10 | **20** | 10 | 30 | **40** | 10 | 60 | **70** | 10 | 200 | **210** | 20 | 2.237 | **2.257** |
| **Prot** | 11 | 11 | **22** | 11 | 33 | **44** | 11 | 66 | **77** | 11 | 200 | **211** | 24 | 2.232 | **2.256** |
| **Hist** | 4 | 4 | **8** | 4 | 12 | **16** | 4 | 24 | **28** | 4 | 200 | **204** | 7 | 2.256 | **2.263** |
| **Ribo** | 40 | 5 | **45** | 40 | 40 | **80** | 40 | 100 | **140** | 40 | 200 | **240** | 81 | 2.146 | **2.227** |

Com esta estrutura de dados, pode-se mensurar as diferenças de desempenho da TSVM comparada com a SVM, principalmente em virtude do acréscimo progressivo do total de amostras nos 4 conjuntos de treinamento, como mostra a Tabela 7.5. Com esta iniciativa, há 20 ($5 \text{ classes} \times 4 \text{ conj. trein.}$) diferentes experimentos que serão repetidos 10 vezes, cada um caracterizado por diferentes conjuntos de treinamento e predição, selecionados aleatoriamente. Com isto o número total de experimentos será : $2 \times 10 \times 20 = 400$ (200 para SVM e 200 para TSVM).

**Especificação dos Parâmetros Utilizados para TSVM e SVM**

As técnicas TSVM e SVM foram empregadas com os seguintes produtos internos kernel: função polinomial de graus $d$ = 1, 2 e 3 (ver equação (3.8)), e RBF (ver equação (3.7)) com o parâmetro $\sigma$ igual à mediana da distância euclidiana de cada amostra positiva em relação à amostra negativa mais próxima (Brown *et al*., 2000). As configurações dos produtos internos kernel que produziram os melhores resultados para os experimentos estão apresentadas na Tabela 7.6.

O parâmetro de penalização $C$ (Seção 4.3) foi escolhido experimentalmente, e os que produziram o melhor grau de generalização também são mostrados na Tabela 7.6.

**Tabela 7.6** : Valores dos Parâmetros de Penalização e dos Produtos Internos Kernel

| Método | Param. | TCA | Resp | Prot | Hist | Ribo |
|---|---|---|---|---|---|---|
| TSVM | Kernel | RBF $\sigma = 4,1$ | RBF $\sigma = 3,7$ | RBF $\sigma = 3,1$ | polin. d = 2 | RBF $\sigma = 3,6$ |
| | *C* | 2 | 2 | 2 | 0.01 | 2 |
| SVM | Kernel | RBF $\sigma = 4,1$ | RBF $\sigma = 3,7$ | RBF $\sigma = 3,1$ | polin. d = 2 | RBF $\sigma = 3,6$ |
| | *C* | 1 | 1 | 1 | 0.001 | 1 |

Devido à função ganho adicional *G* (equação (7.1)) apresentar custos diferentes para os tipos de erros de classificação, utilizou-se os fatores de custo (Seção 4.5) para conseguir um melhor desempenho. A estratégia sugerida por Lin *et al*. (2000) para o cálculo dos custos $C_+$ e $C_-$ (equação (4.28)) não obteve sucesso para este conjunto de dados, possivelmente devido à pouca quantidade de amostras do conjunto de treinamento. A estratégia que obteve o melhor desempenho foi simplesmente utilizar os valores de $C_+ = 2$ e $C_- = 1$, os mesmos custos da função ganho adicional (equação (7.1)). Com isto, o parâmetro da razão de custos é dado por : $RC = C_+ / C_- = 2$.

Para TSVM, o parâmetro do usuário *num*+ (número de amostras de predição a serem classificadas na classe positiva) foi escolhido com o valor igual ao número de amostras positivas presentes no conjunto de dados de predição.

**Resultados**

As simulações foram executadas nas mesmas condições da análise anterior. O tempo médio de CPU gasto para cada uma das 200 simulações de TSVM foi de aproximadamente 95s, e menos de 1s em média para cada uma das 200 simulações de SVM.

Os resultados dos experimentos (cada um repetido 10 vezes com uma re-amostragem dos conjuntos de dados de treinamento e predição) comparando TSVM e SVM são mostrados na Tabela 7.7 e na Figura 7.2.

De acordo com os resultados da Tabela 7.7, e utilizado o teste estatístico não-paramétrico de Wilcoxon (Siegel, 1956) para amostras pareadas com nível de significância de 5% (0,05), vemos que, para todas as 5 classes funcionais no 1º. e 2º. conjuntos de treinamento, a metodologia transdutiva (TSVM) foi significativamente superior à metodologia indutiva. Para o 3º. conjunto de treinamento, TSVM foi significativamente superior para 2 classes, e com tendência a ser significativa (p-valor < 0,1) nas outras duas classes. E como já era esperado, não

**Tabela 7.7** : Comparação do desempenho das metodologias TSVM e SVM para a classificação do conjunto de predição, medida pela função ganho adicional (7.1). São apresentados a média e o desvio padrão para as 10 simulações com amostras diferentes para cada um dos 20 experimentos de TSVM e SVM. Para comparar as duas metodologias, foi utilizado o teste estatístico não-paramétrico de Wilcoxon para amostras pareadas. Para a construção da última coluna, o nível de significância adotado foi de 0,05, onde S=significativo, T= tendência a ser significativo e NS = não significativo.

| Classe | | Conj. Trein. | | Conj. | TSVM | | SVM | | Teste Wilcoxon | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Pos. | Neg. | Pred. | média | desvio | média | desvio | p-valor | |
| TCA | 1 | 5 | 5 | 2.262 | -1,8 | 3,2 | -563 | 366 | 0,002 | S |
| | 2 | 5 | 15 | 2.262 | 1,2 | 5,3 | -50,1 | 37,1 | 0,002 | S |
| | 3 | 5 | 50 | 2.262 | 3,5 | 4,3 | -1,5 | 2,9 | 0,023 | S |
| | 4 | 5 | 200 | 2.262 | 2,3 | 4,6 | 1,4 | 2,0 | 0,543 | NS |
| Resp | 1 | 10 | 10 | 2.257 | 12,1 | 5,8 | -196 | 165 | 0,002 | S |
| | 2 | 10 | 30 | 2.257 | 15,9 | 6,6 | -34,7 | 22,0 | 0,002 | S |
| | 3 | 10 | 60 | 2.257 | 16,6 | 7,5 | 1,2 | 11,4 | 0,004 | S |
| | 4 | 10 | 200 | 2.257 | 18,7 | 3,3 | 18,5 | 2,2 | 0,742 | NS |
| Prot | 1 | 11 | 11 | 2.256 | 30,2 | 8,3 | -113 | 110 | 0,002 | S |
| | 2 | 11 | 33 | 2.256 | 34,2 | 2,9 | 13,0 | 16,7 | 0,002 | S |
| | 3 | 11 | 66 | 2.256 | 34,2 | 3,6 | 31,1 | 3,6 | 0,055 | T |
| | 4 | 11 | 200 | 2.256 | 34,9 | 3,5 | 36,0 | 2,9 | 0,113 | NS |
| Hist | 1 | 4 | 4 | 2.263 | 0,1 | 5,3 | -62,3 | 66,0 | 0,004 | S |
| | 2 | 4 | 12 | 2.263 | 7,1 | 5,3 | -3,3 | 12,3 | 0,025 | S |
| | 3 | 4 | 24 | 2.263 | 9,3 | 2,8 | 2,7 | 6,7 | 0,059 | T |
| | 4 | 4 | 200 | 2.263 | 10,1 | 2,0 | 4,8 | 5,1 | 0,074 | T |
| Ribo | 1 | 40 | 5 | 2.227 | 131,0 | 2,7 | -289 | 203 | 0,002 | S |
| | 2 | 40 | 40 | 2.227 | 138,9 | 5,4 | 110,6 | 13,4 | 0,002 | S |
| | 3 | 40 | 100 | 2.227 | 141,9 | 5,1 | 131,1 | 4,6 | 0,002 | S |
| | 4 | 40 | 200 | 2.227 | 143,9 | 3,9 | 142,4 | 3,1 | 0,140 | NS |

houve diferença significativa entre as 2 metodologias para o 4$^{o}$. conjunto de treinamento.

Graficamente, pela Figura 7.2, notamos o melhor desempenho da TSVM para os conjuntos de treinamento pequenos. Quando o tamanho do conjunto de treinamento aumenta progressivamente, a diferença entre o desempenho das duas metodologias, TSVM e SVM, se reduz proporcionalmente.

Para a metodologia indutiva (SVM), a classe TCA (ver Tabela 7.7) apresenta, nos 3 primeiros conjuntos de treinamento, o resultado da função ganho adicional (7.1) inferior a 0, e na classe Resp, nos 2 primeiros conjuntos de treinamento, a função ganho adicional também inferior a 0. Na Tabela 7.4, vemos que estas 2 classes são as que apresentam os maiores índices de complexidade.

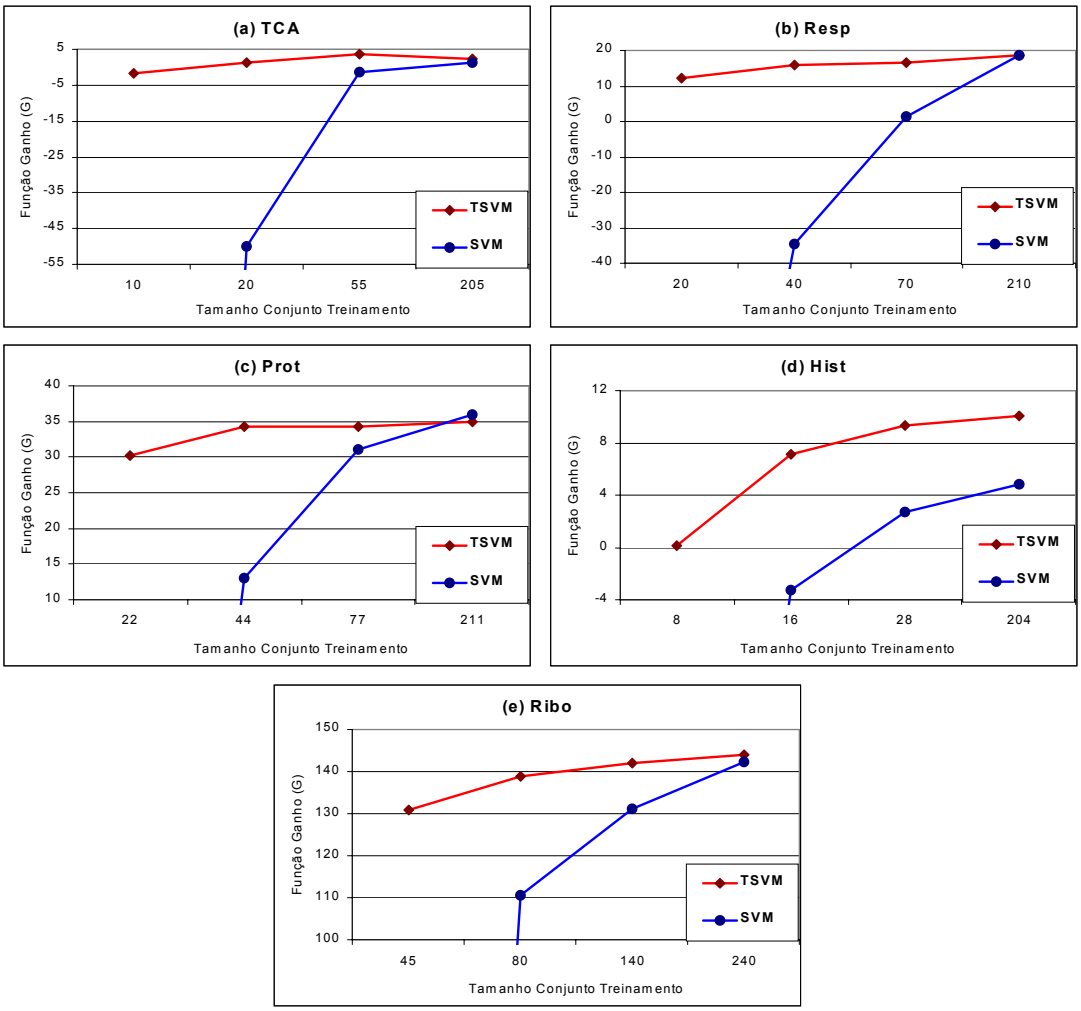


**Figura 7.2** : Comparação gráfica de desempenho das metodologias indutiva (SVM) e transdutiva (TSVM) para as 5 diferentes classes funcionais, através da variação do tamanho do conjunto de treinamento (eixo *x*). Cada ponto representa a média de 10 simulações com diferentes amostragens do conjunto de dados. Para as 5 classes, o resultado do método SVM no 1º. conjunto de treinamento ficou fora do intervalo mostrado nos gráficos. Estes resultados podem ser vistos na Tabela 7.7.

Para as 5 classes funcionais no 1º. conjunto de treinamento, o resultado da função ganho adicional na metodologia indutiva foi muito inferior a 0. Sendo que para a classe Hist, aquela com o menor número de amostras positivas no conjunto de treinamento, o 2º. conjunto de treinamento apresentou a função ganho adicional também inferior a 0 para a metodologia indutiva.

Estes resultados mostram a ineficácia da metodologia indutiva para tratar de problemas com alta complexidade e com conjunto de treinamento de tamanho reduzido, mesmo para a

técnica SVM, em que o problema da dimensionalidade para pequenos conjuntos de treinamento é reduzido pela utilização da representação dual do problema de otimização.

## Estudo dos Vetores-suporte

Na Seção 4.4, vimos que, em SVM, a complexidade da estrutura depende do número de vetores-suporte, e não da dimensão do espaço característico (dimensão 79 para esta aplicação).

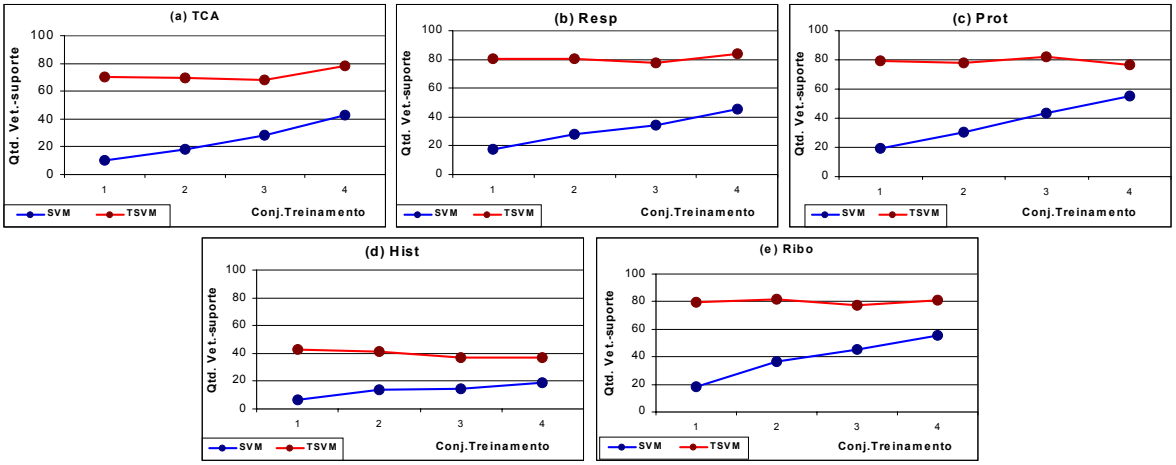


**Figura 7.3** : Quantidade de amostras que são vetores-suporte para cada experimento. Os pontos apresentados nos gráficos representam a média de 10 simulações com amostragens diferentes para cada experimento

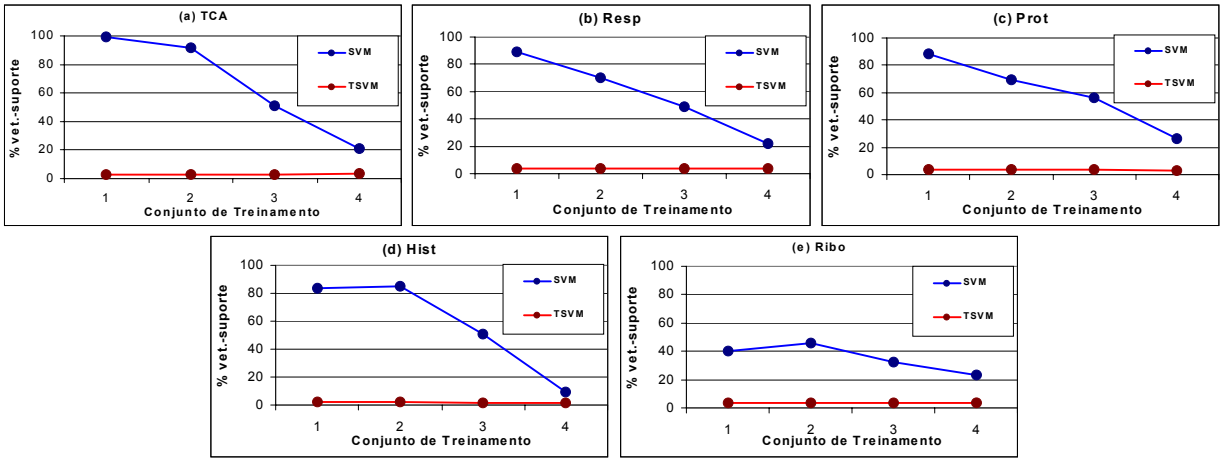


**Figura 7.4** : Porcentagem das amostras que são vetores-suporte para cada experimento. Os pontos apresentados nos gráficos representam a média de 10 simulações com amostragens diferentes para cada experimento.

Nas Figuras 7.3 e 7.4, são apresentadas a quantidade e a porcentagem de amostras do conjunto de dados (conj. de treinamento para SVM e conj. de treinamento + predição para TSVM), respectivamente, que se tornaram vetores-suporte para cada experimento realizado nesta aplicação.

A Figura 7.3, apresenta a quantidade de amostras do conjunto de dados que se tornaram vetores-suporte, porém como a quantidade de amostras do conjunto de treinamento é crescente, estes gráficos podem levar a conclusões erradas, por isto eles têm um caráter apenas exploratório.

Os gráficos da Figura 7.4, mostram que, para SVM, conforme o tamanho do conjunto de treinamento aumenta, a quantidade de vetores-suporte diminui proporcionalmente, e com isto a complexidade da estrutura (Seção 4.4) que é dependente desta proporção diminui, e conseqüentemente o desempenho da metodologia SVM aumenta progressivamente, como mostra a Tabela 7.7.

Para TSVM, a Figura 7.4 mostra que a proporção de vetores-suporte nos 4 conjuntos de treinamento + predição é constante, pois o aumento do conjunto de treinamento não é tão considerável, visto que o conjunto de predição, utilizado no treinamento e no cálculo da proporção é da ordem de 2.200 amostras.

Notamos também que os valores da proporção de vetores-suporte para SVM nos primeiros conjuntos de treinamento para as 5 classes são proporcionais ao índice de complexidade apresentado para cada classe na Tabela 7.4. Ou seja, quando a complexidade de classificação é alta, a proporção de amostras que são vetores-suporte é alta também, como mencionado na Seção 4.4.

### 7.1.4 Conclusões

De maneira similar aos resultados obtidos por Joachims (1999b) para o problema de classificação de textos, a aplicação associada com o problema de classificação de diagnósticos de câncer e classificação de classes funcionais utilizando dados de expressão gênica conduziram à afirmação de que a inferência transdutiva produz melhores classificadores do que a inferência indutiva/dedutiva, ambos os métodos utilizando SVM como classificador, na presença de um número reduzido de amostras de treinamento pertencentes a um espaço de alta dimensão. Quando o tamanho do conjunto de treinamento aumenta, o desempenho da inferência indutiva/dedutiva se aproxima do desempenho da inferência transdutiva.

No problema de classificação de genes em classes funcionais, a metodologia indutiva não foi capaz de classificar as 5 classes para o 1º. conjunto de treinamento com o mínimo de desempenho, função ganho adicional inferior a 0 (ver Tabela 7.7). Para as 2 classes com os maiores índices de complexidade (TCA e Resp, ver Tabela 7.4), o 2º. conjunto de treinamento também apresentou a função ganho adicional inferior a 0 na metodologia indutiva. A Figura 7.4 mostra que os valores da proporção de vetores-suporte para SVM nos primeiros conjuntos de treinamento para as 5 classes são proporcionais ao índice de complexidade da Tabela 7.4. Assim, quando a complexidade da classificação é alta, a proporção de amostras que são vetores-suporte é alta também, e a correlação com a ineficiência da técnica indutiva aplicada a pequenos conjuntos de treinamento é significativa.

No problema de classificação de diagnóstico de câncer, de baixa complexidade de classificação, apenas 6 amostras de treinamento e não mais do que 70 amostras de predição, são necessárias para a aplicação da metodologia transdutiva ter um desempenho de aproximadamente 100%.

Como conseqüência, a abordagem transdutiva utilizando SVM (TSVM) pode contribuir muito para o problema de classificação utilizando dados de expressão gênica, onde o conjunto de amostras previamente classificadas é geralmente de tamanho pequeno, e ainda existem muitas amostras para serem classificadas.

## 7.2 Aplicação 2: "Construindo Modelos de Concessão de Crédito Bancário para a Predição de Inadimplência, com variações da quantidade de Amostras de Treinamento"

### 7.2.1 Motivação

Nesta aplicação, utilizou-se os conceitos da inferência transdutiva associada à técnica SVM (TSVM) para resolver um problema de concessão de crédito bancário com o objetivo de minimizar as perdas de crédito com futuras inadimplências de clientes e, com isto, maximizar a receita final da instituição bancária.

O problema em questão se refere à previsão da classificação de futuros clientes em duas classes: *Mau* (cliente inadimplente no produto de crédito cheque especial) e *Bom* (cliente não inadimplente no produto de crédito cheque especial).

Algumas características tornam este problema de classificação de alta complexidade:

- Problema de difícil classificação (complexo) devido à não existência de um elenco adequado de atributos que consigam separar as amostras nas duas classes. Os vetores de atributos disponíveis para cada amostra freqüentemente possuem valores muito próximos para amostras pertencentes a classes diferentes;
- Conjunto de treinamento com número reduzido de amostras;
- O interesse não está em apenas classificar os futuros clientes nas duas classes (bom e mau), mas também em predizer a probabilidade do cliente tornar-se inadimplente.

Esta aplicação responde à questão proposta por Joachims (1999b): "Será possível utilizar a função de decisão, obtida com a ajuda dos princípios da inferência transdutiva, para predizer de forma indutiva futuras amostras de predição?".

Realizou-se uma comparação da TSVM com a SVM indutiva e com as técnicas de classificação Regressão Logística e Rede Neural Artificial, estudando o desempenho destas técnicas com a variação progressiva do tamanho do conjunto de dados de treinamento. Como será apresentado mais adiante, obteve-se resultados satisfatórios com a aplicação de TSVM, mostrando uma melhora significativa da capacidade de generalização e uma melhora no

desempenho da função de decisão quando aplicada à predição da probabilidade de inadimplência de novos clientes, sendo apenas treinada com um pequeno conjunto de amostras de treinamento.

### 7.2.2 Introdução

A atividade de concessão de crédito por instituições financeiras de grande porte, que focam seus negócios no segmento de mercado do varejo, exige que centenas de operações de crédito sejam avaliadas diariamente para serem aprovadas ou rejeitadas. Esta decisão é muito importante, pois as instituições evitam aprovar operações de clientes com alto risco de não pagarem seus compromissos, tornando-se inadimplentes.

Quando esta decisão de aprovação da operação de crédito é realizada por um analista de crédito, muito tempo é gasto pela unidade de negócio (exemplo: agência bancária) para responder aos seus clientes sobre a aceitação ou não do crédito, além do elevado gasto da instituição com analistas de crédito e do problema da avaliação conter um alto grau de subjetividade.

Por isto, foi necessário a criação de uma ferramenta que avaliasse em grande escala o risco de inadimplência associado a cada operação. Assim surgiram os modelos denominados "*Credit Scoring*" (Lewis, 1994 ; Mays, 1998), construídos tradicionalmente através de técnicas de estatística multivariada, principalmente a Regressão Logística (Hosmer & Lemeshow, 1989; McCullagh *et al.*, 1983), e mais recentemente através da técnica de inteligência artificial denominada Rede Neural Artificial (Bishop, 1995 ; Haykin, 1999).

Os modelos de *Credit Scoring* são implantados nos sistemas das instituições e interligados com todas as suas unidades de negócios. Assim, a resposta de uma análise de concessão de crédito é *on-line*, diminuindo os custos com pessoal e unificando a decisão de crédito, deixando de ser subjetiva.

Os modelos de *Credit Scoring* geralmente são construídos para serem específicos para a aprovação de operações em determinado produto de crédito, além de poderem ser específicos por região geográfica.

Neste problema real de criação do modelo de *Credit Scoring*, o conjunto de dados pertence à instituição financeira Unibanco. Iremos construir um modelo específico para o produto Cheque Especial (produto de crédito rotativo no qual o cliente possui um limite de crédito à sua disposição em sua conta corrente), e também específico para uma unidade de negócios (UN), onde o perfil dos clientes desta unidade diferem do perfil dos clientes do restante do banco.

Para cada cliente, o período de desempenho do modelo é de 12 meses após a contratação do produto cheque especial, definição padrão para a construção de modelos de *Credit Scoring* (Lewis, 1994 ; Mays, 1998). Portanto, o modelo estará prevendo a probabilidade do cliente tornar-se inadimplente no produto cheque especial por um período de até um ano após a contratação do produto.

Para selecionar as amostras (clientes) que formarão o conjunto de treinamento para a construção do modelo, é necessário retroagir a, no mínimo, um ano atrás, selecionar todos os clientes que contrataram o produto neste período passado, e classificá-los como:

- *Mau*: cliente que durante o período de desempenho do modelo (12 meses) tornou-se inadimplente no produto cheque especial, excedendo o seu limite de crédito e não regularizando o seu saldo devedor;
- *Bom*: cliente que durante o período de desempenho do modelo utilizou o produto de forma adequada;
- *Encerrado*: cliente que encerrou o contrato de cheque especial voluntariamente durante o período de desempenho do modelo;
- *Indeterminado*: cliente cujo comportamento na utilização do produto não permite vinculá-lo às classes anteriores.

A variável resposta do modelo será : Mau (resposta= −1) e Bom (resposta= +1). Os clientes classificados como Encerrados ou Indeterminados são excluídos do conjunto de dados.

### 7.2.3 Dados

O ponto chave desta aplicação é que os modelos de *Credit Scoring* serão criados variando o número de amostras do conjunto de treinamento (total de clientes utilizados no conjunto de treinamento) para com isto verificar os efeitos no desempenho da TSVM, quando comparada aos métodos indutivos, com o aumento progressivo do tamanho do conjunto de treinamento.

A Tabela 7.8 mostra as quantidades de clientes de cada classe utilizados nos conjuntos de treinamento, predição e validação, com o propósito de medir o desempenho da técnica TSVM para predição de novas amostras de predição através do conjunto de validação.

**Tabela 7.8** : Estrutura dos Dados para a aplicação da TSVM e alguns Métodos Indutivos.

| | Conjunto de Treinamento | | | Conjunto de Predição | | | Conjunto de Validação | | |
|---|---|---|---|---|---|---|---|---|---|
| Conjunto | Pos. | Neg. | Total | Pos. | Neg. | Total | Pos. | Neg. | Total |
| 1 | 5 | 5 | **10** | 500 | 500 | **1.000** | 500 | 500 | **1.000** |
| 2 | 10 | 10 | **20** | 500 | 500 | **1.000** | 500 | 500 | **1.000** |
| 3 | 25 | 25 | **50** | 500 | 500 | **1.000** | 500 | 500 | **1.000** |
| 4 | 50 | 50 | **100** | 500 | 500 | **1.000** | 500 | 500 | **1.000** |
| 5 | 100 | 100 | **200** | 500 | 500 | **1.000** | 500 | 500 | **1.000** |
| 6 | 200 | 200 | **400** | 500 | 500 | **1.000** | 500 | 500 | **1.000** |
| 7 | 500 | 500 | **1.000** | 500 | 500 | **1.000** | 500 | 500 | **1.000** |

Serão construídos 7 modelos de *Credit Scoring*, um para cada situação do conjunto de treinamento (ver Tabela 7.8), sendo cada modelo repetido 10 vezes, cada um caracterizado por diferentes amostragens dos conjuntos de treinamento, predição e validação, selecionados aleatoriamente. Com isto, teremos 70 experimentos para cada técnica utilizada.

O objetivo será classificar corretamente o conjunto de validação (as novas amostras a serem preditas) o qual terá tamanho fixo nos 7 conjuntos de treinamento para tornar possível a comparação de desempenho das técnicas nos diferentes tamanhos do conjunto de treinamento.

O conjunto de predição é formado pelos clientes contratados nos 12 últimos meses, sendo que não houve tempo suficiente para classificá-los como Bom ou Mau, pois o modelo prevê a probabilidade do cliente tornar-se inadimplente no produto cheque especial por um período de até um ano após a contratação do produto (para prever a probabilidade, será utilizado o algoritmo desenvolvido por Platt (1999b) e descrito na Seção 4.6, com o objetivo de converter a resposta da TSVM/SVM em probabilidade). Com isto, eles serão inseridos no treinamento da TSVM, não com o objetivo de serem classificados, como mencionado no Capítulo 6, mas com o propósito de melhorar a capacidade de generalização e aumentar o desempenho da função de decisão quando aplicada de forma indutiva na predição da probabilidade de inadimplência de futuros clientes, que são os alvos desta aplicação, predizendo o conjunto de validação.

Os atributos para a construção dos modelos são provenientes, em sua maioria, da ficha cadastral preenchida pelo cliente com seus dados pessoais ao abrir sua conta corrente no banco. Citamos como exemplos de atributos: sexo, estado civil, escolaridade, idade, endereço, local de trabalho, tempo de emprego, profissão, quantidade de dependentes, quantidade de veículos, ano do veículo, indicadores se possui telefone, investimentos e empréstimos em outros bancos, entre outras informações cadastrais. No total, foram utilizados 42 atributos, sendo 10 variáveis

contínuas e 32 variáveis binárias (onde o valor 1 indica a ocorrência de determinada característica do atributo, e o valor 0 a ausência).

### 7.2.4 Medida de Desempenho

A medida de desempenho a ser maximizada por TSVM e as demais técnicas será a porcentagem de classificação correta no conjunto de dados de validação:

$$\%\ \textit{de classificação correta} = (FP + FN) \,/\, \text{"}\textit{qtd. total de amostras de validação}\text{"},$$

onde $FP$ é a quantidade de falsos positivos e $FN$ é a quantidade de falsos negativos.

### 7.2.5 Especificações das Técnicas

TSVM e SVM:

Utilizou-se em todas as simulações de TSVM o algoritmo descrito no Capítulo 6, e para SVM o algoritmo $\text{SVM}^{\text{light}}$ (Joachims, 1999b), com a abordagem indutiva descrita nos Capítulos 4 e 5.

Os tipos de produtos internos kernel utilizados foram as funções polinomiais de graus $d$ = 1, 2 e 3 (ver equação (3.8)), e as RBFs (ver equação (3.7)) com o parâmetro $\sigma$ igual à mediana da distância euclidiana de cada amostra positiva em relação à amostra negativa mais próxima (Brown *et al*., 2000). As configurações dos produtos internos kernel que produziram os melhores resultados para os 7 modelos estão apresentadas na Tabela 7.9, assim como os parâmetros de generalização $C$ (Seção 4.3) que forneceram o melhor grau de penalização, escolhidos experimentalmente para cada modelo, são também mostrados na Tabela 7.9.

**Tabela 7.9** : Valores dos Parâmetros de Penalização e dos Produtos Internos Kernel.

| Método | Param. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| TSVM | Kernel | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 |
| | *C* | **1,5** | **1,5** | **1,5** | **1,5** | **1** | **1** | **1,5** |
| SVM | Kernel | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 | RBF σ = 1,86 |
| | *C* | **1** | **1** | **1** | **1** | **0,5** | **0,5** | **1** |

Para converter a resposta da TSVM e da SVM em probabilidade, foi utilizado o algoritmo desenvolvido por Platt (1999b), apresentado na Seção 4.6.

Para TSVM, o parâmetro do usuário *num+* (número de amostras de predição a serem classificadas na classe positiva) foi igual a 50% do número de amostras positivas do conjunto de dados de predição.

As simulações foram executadas em um equipamento SunOS 5.6, com 256MB de memória e 167 MHz de CPU. O tempo médio de CPU gasto para cada uma das 70 simulações de TSVM foi de aproximadamente 30s, e menos de 0.5s em média para cada uma das 70 simulações de SVM.

Regressão Logística:

Para regressão logística, que estima a probabilidade condicional da amostra pertencer à determinada classe, o ponto de decisão para classificar uma amostra como positiva ou negativa, visto que os conjuntos de dados são balanceados (mesma proporção de amostras nas duas classes) foi : $p(decisão) = 0{,}5$.

Portanto quando $p(x) > 0{,}5$ a amostra será classificada como pertencente à classe positiva, e caso contrário, pertencente à classe negativa.

As simulações utilizando regressão logística foram executadas em um Pentium 337 MHz com 64MB de memória, utilizando o software SAS. O tempo médio de processamento foi de menos de 1s para cada simulação.

Rede Neural Artificial:

Foi utilizada as redes com arquitetura *feedforward* com três camadas: camada de entrada, camada escondida (ou intermediária), e a camada de saída. O número de neurônios da camada escondida da rede, para as melhores configurações de cada um dos 7 modelos é apresentado na Tabela 7.10.

**Tabela 7.10** : Quantidade de neurônios da camada escondida

| Modelo | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| qtd neurônios | 1 | 1 | 2 | 2 | 2 | 3 | 3 |

A função-objetivo (ou função de erro) a ser minimizada com relação ao vetor de pesos foi à *função de máxima verossimilhança.* De acordo com Bishop (1995), para o problema de classificação podemos considerar o valor de saída da rede treinada com a função de máxima verossimilhança como a probabilidade condicional da amostra pertencer à classe positiva, de

modo análogo à regressão logística. Por isto, utilizou-se também $p(x) = 0,5$ como ponto de decisão para classificar uma amostra como positiva ou negativa.

O método de otimização utilizado para encontrar a solução do problema de minimização foi o método de *Levenberg-Marquardt*. Como não existe a garantia do método de otimização convergir para o mínimo global da função de máxima verossimilhança, e para evitar a convergência do algoritmo para um ponto de mínimo local longe do ótimo, utilizou-se para cada simulação 3 treinamentos preliminares, consistindo de 20% dos dados de treinamento, e apenas com 10 iterações, com o objetivo de escolher os valores iniciais do vetor de pesos igual ao da melhor das 3 preliminares, e com isto melhorar a convergência para um bom mínimo local, perto do ótimo global.

Utilizou-se o método de validação dos resultados conhecido por *validação cruzada* (Bishop, 1995), que consiste, no caso desta aplicação, em obter o menor valor da função-objetivo para o conjunto de dados de predição.

As simulações foram realizadas nas mesmas condições da Regressão Logística, gastando em média 125s para cada simulação.

### 7.2.6 Resultados

Os resultados dos experimentos (cada um repetido 10 vezes com uma re-amostragem dos conjuntos de dados de treinamento, predição e validação) comparando TSVM, SVM, Regressão Logística e Rede Neural Artificial são mostrados na Tabela 7.11 e Figura 7.5, onde a composição dos conjuntos de treinamento está especificada na Tabela 7.8.

Na Tabela 7.12, é apresentado os resultados do teste estatístico não-paramétrico de Wilcoxon (Siegel, 1956) para amostras pareadas com nível de significância de 5% (0,05), com o objetivo de verificar a existência de diferenças significativas entre todas as combinações 2 a 2 de todas as possibilidades de agrupamento entre as 4 técnicas.

Vemos pela Tabela 7.11 e Figura 7.5 que até o $4^{o}$. modelo, o desempenho melhor é da técnica TSVM e sempre com SVM com o segundo melhor desempenho. Para o $5^{o}$. modelo, houve um empate entre as duas metodologias, e para o $6^{o}$. e $7^{o}$. modelos, onde o tamanho do conjunto de treinamento é de 400 e 1.000 amostras, houve um decréscimo de desempenho de TSVM e SVM comparados com as outras 2 técnicas.

**Tabela 7.11** : Comparação do desempenho das 4 técnicas para a classificação do conjunto de validação, medidas pela % de classificação correta. São apresentados a média e o desvio padrão para as 10 simulações de cada uma das 28 situações, 7 modelos e 4 técnicas.

| | TSVM | | SVM | | Reg.Logist. | | Rede Neural | |
|---|---|---|---|---|---|---|---|---|
| | **média** | **desvio** | **média** | **desvio** | **média** | **desvio** | **média** | **desvio** |
| **1** | **61,1** | 2,8 | **58,9** | 3,5 | **53,7** | 3,9 | **54,1** | 3,6 |
| **2** | **60,9** | 3,2 | **57,9** | 3,8 | **49,9** | 5,5 | **56,5** | 4,6 |
| **3** | **61,4** | 4,0 | **60,1** | 4,1 | **53,0** | 4,2 | **59,8** | 3,4 |
| **4** | **64,0** | 2,2 | **63,0** | 2,4 | **59,8** | 2,7 | **60,8** | 3,3 |
| **5** | **64,1** | 2,1 | **64,2** | 2,2 | **62,7** | 2,5 | **62,3** | 2,2 |
| **6** | **65,6** | 1,5 | **66,2** | 1,1 | **66,1** | 1,3 | **64,1** | 1,8 |
| **7** | **66,7** | **1,6** | **67,2** | 1,8 | **68,3** | 1,6 | **67,4** | 2,6 |

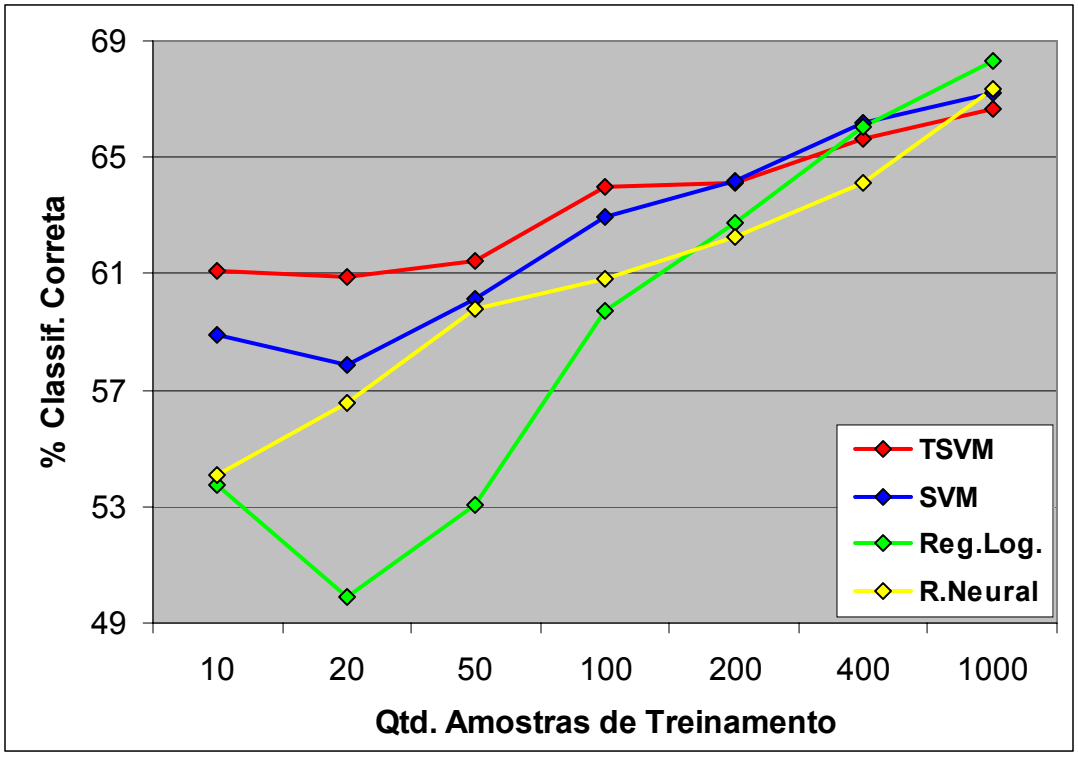


**Figura 7.5** : Comparação gráfica do desempenho das 4 técnicas no conjunto de validação para os 7 diferentes tamanhos do conjunto de treinamento (eixo *x*). Cada ponto representa a média de 10 simulações para cada uma das 28 diferentes situações, 7 modelos e 4 técnicas.

**Tabela 7.12** : Comparação das 4 técnicas utilizando o teste estatístico não-paramétrico de Wilcoxon para amostras pareadas para verificar a existência de diferenças significativas entre todas as combinações 2 a 2 de todas as possibilidades de agrupamento entre as 4 técnicas. O nível de significância adotado foi de 5% (0,05). É apresentado na tabela o p-valor do teste. As comparações feitas foram da técnica da linha superior do título da Tabela (em amarelo) em relação à técnica da segunda linha de título da Tabela (em verde). A cor **vermelha** representa as situações em que houve diferença significativa entre as 2 técnicas. A cor **azul** representa as situações em que houve diferença significativa entre as 2 técnicas, mas a melhor técnica é a da segunda linha de título da Tabela. A cor **amarelo ouro** representa a situação em que houve tendência a ser significativo, p-valor < 10% (0,1).

| | TSVM | | | SVM | | Reg.Log. |
|---|---|---|---|---|---|---|
| | SVM | Reg.Log. | R.Neural | Reg.Log. | R.Neural | R.Neural |
| 1 | 0,004 | 0,004 | 0,002 | 0,037 | 0,002 | 0,840 |
| 2 | 0,010 | 0,002 | 0,020 | 0,010 | 0,430 | 0,013 |
| 3 | 0,203 | 0,004 | 0,449 | 0,004 | 0,920 | 0,002 |
| 4 | 0,037 | 0,006 | 0,195 | 0,014 | 0,130 | 0,172 |
| 5 | 1,000 | 0,375 | 0,084 | 0,300 | 0,037 | 0,710 |
| 6 | 0,125 | 0,510 | 0,041 | 0,860 | 0,009 | 0,019 |
| 7 | 0,125 | 0,023 | 0,160 | 0,039 | 0,509 | 0,131 |

Observa-se nesta aplicação de predição de novas amostras (conjunto de validação) que o desempenho da TSVM para pequenos conjuntos de dados de treinamento é superior. Conforme o conjunto de treinamento cresce de tamanho, o desempenho da TSVM para predição diminui proporcionalmente em relação às outras técnicas indutivas.

Nota-se também que, entre as técnicas indutivas, SVM apresenta resultados superiores nos 5 primeiros conjuntos de treinamento, e um empate no 6º. conjunto de treinamento.

Pela Tabela 7.12, vemos que TSVM foi significativamente superior nos 2 primeiros conjuntos de treinamento, no 3º., 4º., 5º. e 6º. conjuntos de treinamento apenas em algumas situações houve diferenças significativas.

### 7.2.7 Conclusões

Para o problema da construção de modelos de concessão de crédito bancário (*Credit Scoring*), a aplicação de TSVM na forma indutiva para a predição de inadimplência de futuros clientes torna-se uma ótima alternativa para quando existem poucas amostras no conjunto de dados de treinamento. Esta afirmação encontra subsídios no desempenho da técnica TSVM

mostrado na Figura 7.5 e Tabelas 7.11 e 7.12, onde para conjuntos de treinamento variando de 10 a 100 amostras, TSVM foi melhor do que os três métodos indutivos utilizados.

Esta pioneira aplicação de TSVM na forma indutiva, motivada pelo questionamento de Joachims (1999b), "Será possível utilizar a função de decisão, obtida com a ajuda dos princípios da inferência transdutiva, para predizer de forma indutiva futuras amostras de predição?", obteve resultados satisfatórios, mostrando uma melhora significativa da capacidade de generalização e aumento do desempenho da função de decisão aplicada à predição de novas amostras, na presença de poucos dados previamente classificados. Este melhor desempenho da TSVM, quando da existência de poucas amostras no conjunto de treinamento, deve-se à contribuição dos dados do conjunto de predição, agregando informação adicional sobre o problema.

Além disto, tanto para TSVM como para SVM, a melhora significativa da capacidade de generalização e o aumento do desempenho da função de decisão, quando da existência de poucas amostras no conjunto de treinamento, deve-se também ao fato estudado na Seção 4.3, de que SVM proporciona um método que controla a complexidade da técnica independente da dimensão dos dados (dimensão 42 para esta aplicação). O problema da dimensionalidade para pequenos conjuntos de treinamento é reduzido pela utilização da representação dual do problema de otimização, que calcula os parâmetros do hiperplano ótimo tendo os dados de treinamento na forma de produto interno e assim formando uma matriz quadrada ("Matriz Kernel" para o caso de utilização de transformações não-lineares) de mesma dimensão da quantidade de dados de treinamento.

Portanto, para a construção de modelos de concessão de crédito bancário (*Credit Scoring*), a metodologia transdutiva será muito útil para quando o modelo for construído para unidades de negócios com pouco tempo de abertura, e conseqüentemente, poucos clientes com classificação. Neste tipo de aplicação em que o objetivo é predizer a classificação de futuros clientes, o conjunto de predição pode ser formado com os clientes que contrataram o produto de crédito nos últimos 12 meses e, sendo assim, não havendo tempo suficiente para classificá-los como Bom ou Mau.

# Capítulo 8

# Conclusões

Esta dissertação representou um esforço no sentido de contribuir na elaboração de um texto diferenciado para a introdução da técnica de aprendizado de máquina denominada Support Vector Machines (SVM). Houve uma preocupação constante em:

- Promover um nível de tratamento uniforme junto aos vários tópicos envolvidos;
- Manter uma coerência de notação ao longo de todo o texto;
- Abordar os conceitos da teoria do aprendizado estatístico com um nível de profundidade suficiente para permitir a exposição das garantias teóricas da técnica SVM;
- Abordar os fundamentos básicos de otimização e funções kernel com o propósito de contribuir para o entendimento das demonstrações matemáticas dos princípios da SVM;
- Apresentar todos os passos para se chegar ao processo final de otimização em representação dual, considerando todas as situações possíveis. As principais propriedades da solução dual foram destacadas, envolvendo vetores-suporte, matriz kernel, parâmetro de generalização e vínculos com a teoria do aprendizado estatístico;
- Apresentar as motivações e uma visão crítica associada a cada etapa dos algoritmos já propostos na literatura e que seriam utilizados nas aplicações;
- Abordar fatores de custo e conversão da resposta da SVM em probabilidade, devido à sua utilização nas aplicações;
- Interpretar geometricamente os principais aspectos conceituais envolvidos, particularmente em relação ao uso do hiperplano ótimo.

Após este investimento no posicionamento dos aspectos teóricos básicos envolvidos no desenvolvimento da pesquisa, no Capítulo 5 foi descrito em detalhes e de forma crítica o algoritmo $SVM^{light}$, com todos os seus passos, e uma visão geral de outros algoritmos que implementam SVM. Como já mencionado na seção introdutória desta dissertação, não faz parte

das contribuições desta pesquisa a proposição de novos algoritmos ou estratégias de implementação computacional. O que se buscou foi uma análise crítica e pragmática dos algoritmos existentes, de modo a viabilizar sua aplicação em novos contextos.

Em essência, todo o desenvolvimento da pesquisa relatado nas linhas acima corresponde às etapas intermediárias de preparação para o que seria abordado no Capítulo 6. O Capítulo 6 contribuiu para a formalização e apresentação didática da inferência transdutiva, metodologia que se mostra promissora para a tarefa de classificação, mas que ainda é pouco abordada na literatura. Foram tratados aspectos da teoria do aprendizado estatístico relacionados a esta inferência, além da aplicação dos princípios da inferência transdutiva tendo SVM como técnica de classificação. Finalmente, seguiu-se uma apresentação do algoritmo proposto por Joachims (1999b) para a implementação dos conceitos desta inferência associada com a técnica SVM.

Com a disponibilidade de ferramentas computacionais derivadas da teoria do aprendizado estatístico, devidamente analisadas, passou-se então à fase de aplicação, a qual foi dividida em duas partes. Na primeira aplicação, a utilização da inferência transdutiva associada à técnica SVM (TSVM) foi considerada junto aos seguintes problemas de classificação:

- diagnóstico de câncer;
- atribuição de classes funcionais utilizando dados de expressão gênica.

A análise de desempenho conduziu à afirmação de que, na presença de um número reduzido de amostras de treinamento pertencentes a um espaço de alta dimensão, a inferência transdutiva produz melhores classificadores do que a inferência indutiva/dedutiva, ambos os métodos utilizando SVM como classificador. Quando o tamanho do conjunto de treinamento aumenta, o desempenho da inferência indutiva/dedutiva se aproxima do desempenho da inferência transdutiva.

Esta primeira aplicação também trouxe a contribuição de apresentar um estudo inédito do reflexo do desempenho das técnicas SVM e TSVM associadas à complexidade do problema de classificação e à proporção de vetores-suporte, mostrando experimentalmente uma correlação elevada entre estes três itens.

A segunda aplicação, envolvendo o problema de construção de modelos de concessão de crédito bancário (*Credit Scoring*) para a predição de inadimplência de futuros clientes, foi pioneira por ser a primeira aplicação de TSVM na forma indutiva, motivada pelo questionamento de Joachims (1999b): "Será possível utilizar a função de decisão, obtida com a ajuda dos

princípios da inferência transdutiva, para predizer de forma indutiva futuras amostras de predição?". Os resultados obtidos foram satisfatórios, mostrando que, na presença de poucos dados previamente classificados, há uma melhora significativa da capacidade de generalização e aumento do desempenho da função de decisão aplicada à predição de novas amostras, quando comparado com as técnicas indutivas: SVM, Regressão Logística e Rede Neural Artificial.

## 8.1 Questões em Aberto e Perspectivas Futuras

Embora a eficiência da técnica SVM já tenha sido comprovada em muitas áreas de aplicação, algumas etapas desta metodologia requerem estudos mais avançados, como:

- Que tipo de características deve estar presente ou ausente no conjunto de dados para que SVM seja a melhor metodologia a ser utilizada? Parte da resposta a esta questão certamente pode ser extraída de uma análise dos aspectos envolvidos nas aplicações de SVM que estão disponíveis na literatura. Também as aplicações realizadas junto a este trabalho permitem extrair algumas conclusões. Exemplo: no caso da segunda aplicação, o desempenho de SVM é superado por outras abordagens quando se aumenta a cardinalidade do conjunto de dados de treinamento;
- Não existe um método de seleção de atributos (*feature selection*) específico para SVM indutiva, que seja simples e com desempenho garantido, ou seja, um método que seleciona um subconjunto dos atributos, preservando ou melhorando a habilidade em maximizar a margem de separação e o desempenho de generalização do hiperplano, reduzindo a dimensionalidade do espaço de entrada e como conseqüência diminuindo o tempo de processamento. Muitos estudos estão sendo feitos nesta área (Weston *et al*., 2000; Chapelle *et al*. ,2002);
- Não existem métodos de seleção dos parâmetros (o tipo de produto interno kernel e seus parâmetros específicos) que não consumam muito tempo de processamento. Geralmente, o que se faz é utilizar o método de validação cruzada, o qual se torna muito custoso em termos de tempo de processamento;
- Não há como interpretar os resultados produzidos por SVM. Questões como "*qual a influência de cada atributo no modelo final*" ainda não têm resposta. Nesta área, quase não existem estudos em andamento;

- Como tratar atributos categóricos? Geralmente, o que é feito é o mapeamento destes dados para atributos dicotômicos (0 ou 1) e a partir de então o tratamento é o mesmo adotado para os atributos contínuos. Poderia ser criado, por exemplo, um produto interno kernel específico para este tipo de atributo.

Uma direção de pesquisa que abre um novo horizonte para estudos, é a recente adaptação de técnicas de clusterização à utilização da formulação da SVM (Ben-Hur *et al*. , 2001). Os dados são mapeados por meio de um produto interno kernel do tipo RBF (equação (3.7)) para um espaço característico de alta-dimensão, onde uma esfera que circunda o conjunto de dados com o mínimo raio é encontrada. Quando o mapeamento é realizado de volta ao espaço original dos dados, esta esfera é separada em diversos contornos, cada um contendo um subconjunto dos dados. Estes contornos são interpretados como os clusters ou agrupamentos. O parâmetro de variância da RBF, $\sigma^2$, é que controla o número de clusters.

A teoria do aprendizado estatístico é considerada não pertencente a nenhuma área do conhecimento científico. É utilizada principalmente por estatísticos, matemáticos e cientistas da computação. Ela tem seus próprios objetivos, seus próprios paradigmas, e suas próprias técnicas. Apesar do fato das primeiras publicações apresentarem esta teoria como resultado da estatística, os estatísticos, que têm seus próprios paradigmas, nunca consideraram esta teoria como uma parte da estatística. Vapnik (1998) enfatiza que esta teoria está somente em sua infância, ainda existindo muitas áreas desta teoria para serem analisadas e que são importantes para o entendimento dos fenômenos do aprendizado e para as aplicações práticas. Vapnik sugere que esta teoria poderia ser chamada de "*Inferência para Dados Esparsos*". Todos estes tópicos, pouco explorados na literatura, abrem novos caminhos para investigações.

Sobre a teoria da inferência transdutiva presente na teoria do aprendizado estatístico, os argumentos de Vapnik (1998) são algumas vezes confusos. Nos trabalhos de outros autores presentes na literatura, a formulação teórica não é muito transparente. Isto pode explicar o porquê da existência de poucas pesquisas nesta área, apesar do sucesso das aplicações com este tipo de metodologia. Uma elaboração mais didática e ao mesmo tempo mais detalhada é recomendada para que pesquisadores tenham maiores argumentos para explorarem este tipo de inferência. Esforços neste sentido já foram adotados no contexto desta dissertação.

A ligação dos princípios da inferência transdutiva com os conceitos de probabilidade condicional e da fórmula de Bayes, pode auxiliar na explicação dos aspectos teóricos do aprendizado estatístico, presentes na utilização da inferência transdutiva.

Para o algoritmo TSVM proposto por Joachims e descrito no Capítulo 6, a crítica que surge é sobre o parâmetro do usuário *num*+, que apresenta grande influência na solução final. Outra abordagem poderia ser utilizada para resolver a maximização da margem de separação nos dados de treinamento e predição, sem a necessidade de estipulação a priori da quantidade de amostras de predição que pertencerão a cada classe. A própria solução que maximiza a margem, independente desta restrição do parâmetro *num*+, poderia ser utilizada, como sugerido por Vapnik (1998).

## 8.2 Possíveis extensões deste trabalho

A solução exata do problema de otimização resultante da aplicação dos princípios da inferência transdutiva a SVM, equação (6.13) ou (6.14), que em geral é não convexa, requerem uma busca sobre todas as $2^K$ possibilidades de classificação do conjunto de predição para produzir a SVM com a máxima margem de separação baseada em todo o conjunto de dados. Assim, é possível encontrar o ótimo global com os métodos de otimização padrões somente para um número pequeno $K$ de amostras de predição. Para um número grande de amostras de predição, deve-se utilizar algum procedimento de busca iterativa que conduza a uma boa solução, ou seja, um ótimo local do problema de otimização, que até pode ser o ótimo global. Joachims utilizou um método de busca local para resolver este problema, outros autores utilizaram o método de busca *branch-and-bound*, o qual apresenta problemas de tratabilidade com o aumento no número de amostras de treinamento. Como uma alternativa a estas abordagens, é possível implementar um algoritmo evolutivo para resolver o problema de otimização em questão, utilizando técnicas de computação evolutiva com etapas de busca local. No entanto, não está descartada também a utilização de estratégias de busca que empregam outras heurísticas e meta-heurísticas, como por exemplo a busca tabu (Glover & Laguna, 1997).

# Referências Bibliográficas

**Aizerman, M.A., Braverman, E.M. and Rozonoer L.I. (1964a)**, Theoretical foundations of the potential function method in pattern recognition liarning. Automation and Remote Control, vol. 25, pp. 821-835.

**Aizerman, M.A., Braverman, E.M. and Rozonoer L.I. (1964b)**, The probability problem of pattern recognition learning and the method of potential functions. Automation and Remote Control, vol. 25, pp. 1175-1193.

**Bazarra, M., Sherali, H. and Shetty C. M. (1993)**, Nonlinear Programming - Theory and Algorithms, 2nd Edition. John Wiley and Sons.

**Bennett, K. and Demiriz, A. (1998)**, Semi-supervised support vector machines. In Proceedings of Neural Information Processing Systems (NIPS), pp. 368-374.

**Bennett, K., Wu, D. and Auslender, L. (1998)**, On support vector decision trees for database marketing. Research Report No. 98-100, Rensselaer Polytechnic Institute, Troy, NY.

**Bennett, K. and Campbell, C. (2000)**, Support Vector Machines: Hype or Hallelujah ?. SIGKDD Explorations, vol. 2, pp. 1-13.

**Ben-Hur, A., Horn, D., Siegelmann, H.T. and Vapnik, V.N. (2001)**, Support Vector Clustering. Journal of Machine Learning Research 2, pp. 125-137.

**Bishop, C. M. (1995)**, Neural Networks for Pattern Recognition. Oxford University Press.

**Blum, A and Mitchell, T (1998)**, Combining labeled and unlabeled data with co-training. Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98), pp. 92-100. ACM Press.

**Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992)**, A training algorithm for optimal margin classifiers. In D. Haussler,, editor. Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pp. 144-152. ACM Press.

**Brown, M.P.S., Grundy, W.N., Lin, D., Cristianini, N., Sugnet, C.W., Furey, T.S., Ares, Jr M. and Haussler, S. (2000)**, Knowledge-based analysis of microarray gene expression data by using support vector machines. Procedings of the National Academy of Sciences of the USA, 97(1), pp. 262-267.

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

**Cataltepe, Z. and Magdon-Ismail, M. (1998)**, Incorporating test inputs into learning. In Jordan, M.I. , Kearns, M.J. and Solla, S. A. , editors, Advances in Neural Information Processing Systems, vol. 10. The MIT Press.

**Courant, R. and Hilbert D. (1970)**, Methods of Mathematical Physics, vol. I and II, New York: Wiley Interscience.

**Chapelle, O., Haffner, P. and Vapnik, V.N. (1999)**, SVMs for histogram-based image classification. IEEE Transaction on Neural Networks, vol 10.

**Chapelle, O., Vapnik, V., Bousquet, O. and Mukherjee, S. (2002)**, Choosing multiple parameters for support vector machines. Machine Learning, 46(1), pp. 131–159.

**Chu, W., Keerthi, S.S. and Jin Ong, C. (2001)**, A Unified Loss Function in Bayesian Framework for Support Vector Regression. Proc. 18th International Conf. on Machine Learning, pp. 51-58.

**Cristianini, N. and Shawe-Taylor, J. (2000)**, An Introduction to Support Vector Machines. Cambridge U.P.

**Demiriz, A. and Bennett, K. (2000)**, Optimization approaches to semi-supervised learning. In M. Ferris, O. Mangasarian, and J. Pang, editors, Applications and Algorithms of Complementarity, chapter 1. Kluwer Academic Publishers, Boston.

**Dumais, S., Platt, J., Heckerman, D. and Sahami M. (1998),** Inductive learning algoritms and representations for text categorization. In 7th International Conference on Information and Knowledge Management, pp. 148-155.

**Eisen, M., Spellman, P., Brown, M.P.S. and Botstein, D. (1998)**, Cluster analysis and display of genome-wide expression patterns. Procedings of the National Academy of Sciences of the USA, 95, pp. 14863-14868.

**Friess, T. T., Cristianini, N. and Campbell, C. (1998)**, The kernel adatron algorithm: a fast and simple learning procedure for support vectror machines. 15th Intl. Conf. Machine Learning. Morgan Kaufman Publishers, pp. 188-196

**Fung, G. and Mangasarian, O. (1999)**, Semi-supervised support vector machines for unlabeled data classification, Technical Report, Data Minimg Institute.

**Glover, F. and Laguna, M. (1997)**, Tabu Search, Kluwer Academic Publisher.

**Haykin, S. (1999)**. Neural Networks: A Comprehensive Foundation. 2nd edition, Prentice-Hall.

**Hosmer, Jr. D. W. and Lemeshow, S. (1989)**, Applied Logistic Regression. John Wiley and Sons, New York.

**Jaakkola T., Diekhans M. and Haussler, D. (1999)**, A discriminative framework for detecting romote protein homologies. MIT Preprint, 1999.

**James, B.R. (1981),** Probabilidade: um curso a nível intermediário. Instituto de Matemática Pura e Aplicada - CNPq.

**Joachims, T. (1998)**, Text categorization with support vector machines. In Proceedings of European Conference on Machine Learning (ECML), pp 137-142.

**Joachims, T. (1999a)**, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Scholkopf , C. Burges and A.Smola (ed.), MIT-Press, pp. 169-184.

**Joachims, T. (1999b)**, Transductive Inference for Text Classification using Support Vector Machines. Proceedings of the International Conference on Machine Learning (ICML), pp. 200-209.

**Joachims, T. (2000)**, Estimating the Generalization Performance of a SVM Efficiently. Proceedings of the International Conference on Machine Learning (ICML), pp. 431-438.

**Khan, J., Wei, J., Ringner, M., Saal, L., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Atonescu, C., Peterson, C. and Meltzer, P. (2001)**, Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature Medicine 7, pp. 673-679.

**Keerthi, S.S., Shevade, S.K., Bhattacharyya, C. and Murthy, K. (1999)**, A fast iterative nearest point algorithm for support vector machine classifier design. Technical Report TR-ISL-99-03, Indian Institute of Science, Bangalore, India.

**Keerthi, S.S., Duan, K., Shevade S.K. and Poo A.N. (2002)**, A Fast Dual Algorithm for Kernel Logistic Regression. Accepted for ICML(2002)

**Kohane, I. S. (2002)**, Bioinformatics, in Biostatistical Genetics and Genetic Epidemiology, John Wiley & Sons.

**Lee, Y. and Lee, C.-K. (2002)**, Classification of Multiple Cancer Types by Multicategory Support Vector Machines Using Gene Expression Data. UW-Madison Statistics Dept Technical Report 1051.

**Lewis, E. M. (1994)**. An Introduction to Credit Scoring. Fair Isaac & Co., Inc.: California.

**Lima, C.A.M., Coelho, A.L.V. and Von Zuben, F.J. (2002)**, Ensembles of Support Vector Machines for Regression Problems. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'2002), vol. 3, pp. 2381-2386, in the 2002 IEEE World Congress on Computational Intelligence (WCCI'2002), Honolulu, Hawaii.

**Lin, Y., Lee, Y. and Wabba, G. (2000)**, Support vector machines for classification in nonstandard situations. Technical Report 1016. Department of Statistics, University of Wisconsin.

**Luenberger, D. (1984)**, Linear and Nonlinear Programming. Addison-Wesley

**Mangasarian, O.L. and Musicant, D.R. (2000)**, Active support vector machine classification. Technical Report 00-04, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin.

**Mays, E. (1998)**, Credit Risk Modeling: Design and Application. Amacon, New York

**Mercer, J. (1909)**, Functions of positive and negative type and their connection with the theory of integral equations. Philos, Trans. Roy. Soc. London (A), vol. 209, pp. 415-446.

**McCullagh, P. and Nelder, J.A. (1983)**, Generalized Linear Models. 2 ed. Chapman and Hall, London.

**Miller, D. and Uyar, S. (1997)**, A mixture of experts classifier with learning based on both labelled and unlabelled data. In Mozer, M.C. , Jordan, M.I. and Petsche, T. , editors, Advances in Neural Information Processing Systems 9, pp. 571-578. MIT Press.

**Muller, K., Smola, A., Ratsh, G., Scholkopf, B., Kohlmorgen, J. and Vapnik, V.N. (1999)**, Predicting time series with support vector machines. Advances in Kernel Methods - Support Vector Learning, pp. 243-253. MIT Press.

**Nigam, K, McCallum, A., Thrun, S., and Mitchell, T. (1998)**. Learning to classify test form labeled and unlabeled documents using EM. Machine Learning, 39(2/3).

**Osuna, E., Freund, R. and Girosi, F. (1997a)**, An improved training algorithm for support vector machines. In Principe, J., Giles, L., Morgan, N. and Wilson, E., editors, Neural Networks for Signal Processing VII - Proceedings of the 1997 IEEE Workshop, pp. 276-285, New York. IEEE.

**Osuna, E., Freund, R. and Girosi, F. (1997b)**, Training support vector machines: An application to face detection. In Proceedings of Computer Vision and Pattern Recognition, pp. 130-136.

**Pavlidis, P., Weston, J., Cai, J. and Grundy, W.N. (2001)**, Combining microarray expression data and phylogenetic profiles to learn functional categories using support vector machines. In RECOMB, pp. 242-248.

**Platt, J.C. (1999a)**, Fast training of support vector machines using sequential minimal optimization. In Sholkopf, B., Burges, C., and Smola, A., editors, Advances in Kernel Methods - Support Vector Learning, chapter 12. MIT-Press.

**Platt, J.C. (1999b)**, Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In Advances in Large Margin Classifiers. MIT-Press.

**Pontil, M. and Verri, A. (1998)**, Object recognition with support vector machines. IEEE Trans. on PAMI, 20, pp. 637-646.

**Sauer, N. (1972)**, On the densities of families of sets. Journal of Combinatorial Theory, Series A, vol. 13, pp. 145-172.

**Semolini, R. and Von Zuben, F.J. (2002)**, Transductive Support Vector Machines for Cancer Diagnosis and Classification of Microarray Gene Expression Data. In Proceedings of I Brazilian Workshop on Bioinformatics, pp. 102-104.

**Siegel, S. (1956)**, Nonparametric Statistics for Behavioral Sciences. McGraw-Hill

**Smola, A. (1998)**, Learning with Kernels. PhD thesis, Technische universitat Berlin.

**Suykens, J.A.K., Lukas, L. and Vandewalle, J. (2000)**, Sparse approximation using least squares support vector machines. In IEEE International Symposium on Circuits and Systems ISCAS'2000.

**Suykens, J.A.K., Brabanter, J. De, Lukas, L. and Vandewalle, J. (2002)**, Weighted least squares support vector machines: robustness and sparse approximation. Journal: Neurocomputing, vol 48, pp 85-105.

**Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. and Golub, T. (1999)**, Interpreting patterns of gene expression with self-organizing maps. Procedings of the National Academy of Sciences of the USA, 96, pp. 2907-2912.

**Terrence, S.F., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M. and Haussler, D. (2000)**, Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data. UCSC-CRL-00-04.

**Van Gestel, T., Suykens, J.A.K., Lanckriet, G., Lambrechts, A., De Moor, B. and Vandewalle, J. (2002)**, Bayesian framework for least squares support vector machine classifiers, gaussian processes and kernel fisher discriminant analysis. Neural Computation, 14(5), pp.1115-1147.

**Vanderbei, R. (1994)**, Loqo: An interior point code for quadratic programming. Technical report, Princeton University.

**Vapnik, V.N. and Chervonenkis, A. (1971)**, On the uniforme convergence of relative frequencies of events to their probabilities. Theoretical Probability and Its Applications, vol. 17, pp. 264-280.

**Vapnik, V.N. and Chervonenkis, A. (1974)**, Theory of Pattern Recognition. Nauka, Moscow.

**Vapnik, V.N. and Sterin, A. (1977),** On structural risk minimization or overall risk in a problem of pattern recognition. Automation and Remote Control, 10(3), pp. 1495-1503.

**Vapnik, V.N. (1982)**, Estimation of Dependences Based on Empirical Data, New York: Springer-Verlag.

**Vapnik, V.N. (1992)**, Principles of risk minimization for learning theory. Advances in Neural Information Processing Sytems, vol. 4, pp. 831-838, San Mateo, CA: Morgan Kaufmann.

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

**Vapnik, V. N. (1998)**, Statistical Learning Theory. John Wiley and Sons, New York.

**Vapnik, V.N. and Mukherjee, S. (1999)**, Support vector method for multivariant density estimation. In Neural Information Processing Sytems.

**Wahba, G. (1999)**, Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods - Support Vector Learning, pp. 69-88, Cambridge, MA. MIT Press.

**Wahba, G., Lin, Y., Lee, Y. and Zhang, H. (2001)**, Optimal Properties and Adaptive Tuning of Standard and Nonstandard Support Vector Machines. Proceedings of the Mathematical Sciences Research Institute, Berkeley Workshop on Nonlinear Estimation and Classification.

**Wapnik, W. and Tscherwonenkis, A. (1979)**, Theorie der Zeichenerkennung. Akademie Verlag, Berlin.

**Weston, J. and Watkins, C. (1999)**, Multi Class Support Vector Machines, in Proceedings of ESANN99, ed. M. Verleysen, D. Facto Press, Brussels, pp. 219-224.

**Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. and Vapnik, V. (2000)**, Feature selection for support vector machines. In Advances in Neural Information Processing Systems.

**Wismer, D., and Chattergy, R. (1978)**, Introduction to Nonlinear Optimization. North-Holland, New York.

**Wolfe, P. (1972)**, On the convergence of gradient methods under constraint. IBM Journal on Research and Development, 16, pp. 407-411.

**Wu, D., Bennett, K.P., Cristianini, N. and Shawe-Taylor, J. (1999)**. Large margin trees for induction and transduction. In Proc. 16th International Conf. On Machine Learning, pp. 474-483. Morgan Kaufmann, São Francisco, CA.

**Yeo, G and Poggio, T. (2001)**, Multiclass classification of SRBCTs. Technical Report AI Memo 2001-018 CBCL Memo 206, MIT.

**Zoutendijk, G. (1970)**, Mathematical Programming Methods. North-Holland.

# Índice Remissivo de Autores


Aizerman *et al*. (1964a), 19

Aizerman *et al*. (1964b), 19

Bazarra *et al*. (1993), 16

Bennett & Demiriz (1998), 83, 84

Bennett *et al*. (1998), 4

Bennett & Campbell (2000), 6

Ben-Hur *et al*. (2001), 6 ,116

Bishop (1995), 103, 107, 108

Blum & Mitchell (1998), 4

Boser *et al*. (1992), 3, 37, 66

Brown *et al*. (2000), 4, 86, 89, 92, 93, 94, 95, 106

Burges (1998), 37

Cataltepe & Magdon-Ismail (1998), 3

Courant & Hilbert (1970), 22

Chapelle *et al*. (1999), 3

Chapelle *et al*. (2002), 115

Chu *et al*. (2001), 5

Cristianini & Shawe-Taylor (2000), 3, 5, 37

Demiriz & Bennett (2000), 3, 4, 78

Dumais *et al*. (1998), 4

Eisen *et al*. (1998), 86, 87, 92, 93

Friess *et al*. (1998), 66

Fung & Mangasarian (1999), 4, 84

Glover & Laguna (1997), 117

Haykin (1999), 37, 103

Hosmer & Lemeshow (1989), 103

Jaakkola *et al*. (1999), 4

James (1981), 26, 31

Joachims (1998), 4

Joachims (1999a), 7, 55, 64, 65

Joachims (1999b), 3, 4, 7, 69, 78, 79, 81, 82, 84, 90, 100, 102, 106, 111, 114

Joachims (2000), 45

Khan *et al*. (2001), 87, 88

Keerthi *et al*. (1999), 67

Keerthi *et al*. (2002), 6

Kohane (2002), 86

Lee & Lee (2002), 86, 88

Lewis (1994), 103, 104

Lima et al. (2002), 6

Lin *et al*. (2000), 51, 96

Luenberger (1984), 16

Mangasarian & Musicant (2000), 67

Mays (1998), 103, 104

Mercer (1909), 22

McCullagh & Nelder (1983), 103

Miller & Uyar (1997), 3

Muller *et al*. (1999), 6

Nigam *et al*. (1998), 4

Osuna *et al*. (1997a), 57

Osuna *et al*. (1997b), 4, 58

Pavlidis *et al*. (2001), 86

Platt (1999a), 55

Platt (1999b), 53, 54, 66, 105, 106
Pontil & Verri (1998), 3
Sauer (1972), 32
Semolini & Von Zuben (2002), 7, 85
Siegel (1956), 90, 96, 108
Smola (1998), 59
Suykens *et al*. (2000), 5
Suykens *et al*. (2002), 6
Tamayo *et al*. (1999), 86
Terrence *et al*. (2000), 86
Van Gestel *et al*. (2002), 5
Vanderbei (1994), 59
Vapnik & Chervonenkis (1971), 28, 29
Vapnik & Chervonenkis (1974), 3, 10, 20
Vapnik & Sterin (1977), 78, 83
Vapnik (1982), 27, 31-33
Vapnik (1992), 32-34
Vapnik (1995), 2, 3, 5, 26, 37, 43, 48, 49
Vapnik (1998), 2, 3, 6, 24, 27, 31-34, 37, 53, 69, 72-74, 116, 117
Vapnik & Mukherjee (1999), 6
Wahba (1999), 52
Wahba *et al*. (2001), 3
Wapnik & Tscherwonenkis (1979), 78, 83
Weston & Watkins (1999), 6
Weston et al. (2000), 115
Wismer & Chattergy (1978), 59
Wolfe (1972), 60
Wu *et al*. (1999), 3
Yeo & Poggio (2001), 86, 88
Zoutendijk (1970), 59