Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

RECONHECIMENTO DE FALA USANDO MÁQUINAS DE VETOR DE SUPORTE (SVMs)

SANDRO TOMASSONI COELHO

NOVEMBRO / 2005

Reconhecimento de fala usando máquinas de vetor de suporte (SVMs)

Sandro Tomassoni Coelho

Dissertação apresentada ao Instituto Nacional de Telecomunicações, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Carlos Alberto Ynoguti

Santa Rita do Sapucaí
2005

Dissertação defendida e aprovada em 16/11/2005, pela comissão julgadora:

(Carlos Alberto Ynoguti/DCOM-INATEL)

(José Carlos Pereira/SEL-EESC-USP)

(Antonio Marcos Alberti/DTE-Inatel)

Coordenador do Curso de Mestrado

Dedico esta tese a minha
esposa Andréa por sua
paciência e seu amor.

Agradecimentos

Agradeço à Deus por mais esta conquista na minha vida, ao meu orientador e amigo Carlos Alberto Ynoguti pela sua amizade, paciência, profissionalismo e humanidade, agradeço também à minha esposa Andréa pelo seu amor, carinho e paciência, à minha filha Yasmin pela agradável inspiração que me trouxe seu recente nascimento, agradeço à todos os meus colegas e amigos pelo auxílio, apoio e amizade, à meus pais e meus irmãos pelo seu amor e carinho, aos meus familiares pela amizade e compreensão, ao Prof. Dr Clodoaldo da Unicamp pelo auxílio na área de SVM, aos funcionários do Inatel, à Finatel pela bolsa de estudos, e às pessoas simples de Santa Rita que me deram exemplo de tantas coisas boas como a humildade e a caridade, desejo tudo de bom para todos.

Índice

Lista de Figuras

Lista de Tabelas

Resumo

Neste trabalho será focado o estudo de uma técnica de aprendizado de máquina, chamada Máquina de Vetor de Suporte, aplicada problema de reconhecimento de fala para classificar palavras isoladas. Esta técnica vem despertando muito interesse na comunidade científica por apresentar desempenho superior as técnicas já existentes em inúmeras aplicações. Foi proposta uma nova metodologia para a classificação de múltiplas classes e verificado o desempenho do sistema após a normalização dos dados e a utilização da técnica de análise de componente principal para redução de dados.

Abstract

The aim in this work it will be the study of one recent technique of machine learning, called Support Vector Machines, applied to isolated word, speech recognition problem. A new methodology was proposed for the classification of several classes. Also, the system performance was evaluated after the data normalization and the use of the Principal Component Analysis technique for data reduction.

Capítulo 1

Introdução

A fala é a forma mais natural de comunicação entre as pessoas. De fato, é o meio mais utilizado para as atividades do dia-a-dia, proporcionando um meio rápido e eficiente para comunicar idéias e desejos, pensamentos e necessidades.

A interação entre homem e máquina faz uso de outros tipos de interface, tais como botões, chaves, teclas e outros dispositivos, que dependem de um treinamento a priori, em graus variados, dependendo da complexidade do equipamento e da tarefa a ser realizada.

As tecnologias de processamento da fala aparecem como uma alternativa bastante atraente: em um cenário ideal, ao invés de manipular botões ou chaves por exemplo, o usuário simplesmente expressa sua vontade usando a fala, a máquina "entende" o comando, e executa a tarefa desejada.

Em alguns casos, a máquina poderia "conversar" com o usuário, gerando respostas que seriam "faladas" por ela. Desta forma, as possíveis formas de comunicação entre homem e máquina, através da fala, podem ser o reconhecimento da fala, no sentido homem-máquina e a síntese de fala, no sentido máquina-homem. No reconhecimento de fala a máquina reconhece o que foi falado pela pessoa e na síntese da fala a máquina produz a fala para se comunicar.

Nesta linha, algumas aplicações possíveis para o reconhecimento de fala são: controle de máquinas, interface com o computador, ditado, auxílio a deficientes, quiosques de informação, conversa homem-máquina, dentre outras.

O processo de reconhecimento de fala consiste em converter uma seqüência de eventos acústicos em uma seqüência de palavras ou frases. Estas, depois de reconhecidas, podem ser a saída final do sistema ou a entrada de um sistema de processamento de linguagem natural.

Pode-se projetar desde sistemas bastante simples, que reconheçam apenas alguns comandos (por exemplo, para movimentar uma cadeira de rodas), até sistemas altamente complexos, que operam com fala espontânea em ambientes

ruidosos. Obviamente o grau de dificuldade para cada uma destas tarefas é bem diferente, sendo comum a seguinte classificação: [1]

Parâmetros	Faixa
Modo de pronúncia	De palavras isoladas à fala contínua
Estilo de pronúncia	De leitura a fala espontânea
Treinamento	De dependente de locutor a independente de locutor
Vocabulário	De pequeno ($<$ 20 palavras) a grande ($>$ 20000 palavras)
Modelo de linguagem	De estados finitos a sensível a contexto
Perplexidade	De pequena ($<$ 10) a grande ($>$ 100)
SNR	De alta ($>$ 30 dB) a baixa ($<$ 10 dB)
Transdutor	De microfone com cancelamento de ruído a telefone

Tabela 1.1: *Parâmetros que caracterizam um sistema de reconhecimento de fala.*

Descrição dos parâmetros:

- **Modo de pronúncia :** no reconhecimento de palavras isoladas existe a necessidade de uma pausa entre as palavras.
- **Estilo de pronúncia :** a fala espontânea é muito mais difícil de reconhecer que a de leitura pois, como é mais relaxada, tem mais coarticulações.
- **Treinamento:** no modo dependente de locutor, o sistema precisa ser treinado individualmente com cada usuário, enquanto que no modo independente não, pois já foi treinado anteriormente com vários locutores.
- **Vocabulário:** a complexidade da tarefa cresce com o aumento do vocabulário e com a presença de palavras parecidas.
- **Modelo de linguagem:** quando a fala é composta de uma seqüência de palavras, são determinadas quais as palavras que podem seguir uma dada palavra. Já para modelos que se aproximam da linguagem natural as definições são feitas a nível de gramática sensível a contexto.
- **Perplexidade:** é uma medida da complexidade da tarefa e envolve vocabulário e modelo de linguagem. Pode ser definida grosseiramente como o número de palavras que pode seguir uma determinada palavra depois que o modelo de linguagem foi aplicado.
- **Razão sinal ruído - SNR:** Quanto maior o nível de ruído do ambiente, mais difícil é a tarefa de reconhecimento.
- **Transdutor:** O tipo e a posição do microfone também interferem no reconhecimento.

Na Figura 1 tem-se um panorama geral do desempenho dos sistemas de reconhecimento automático de fala para diversas tarefas, com graus de dificuldade variados[2].

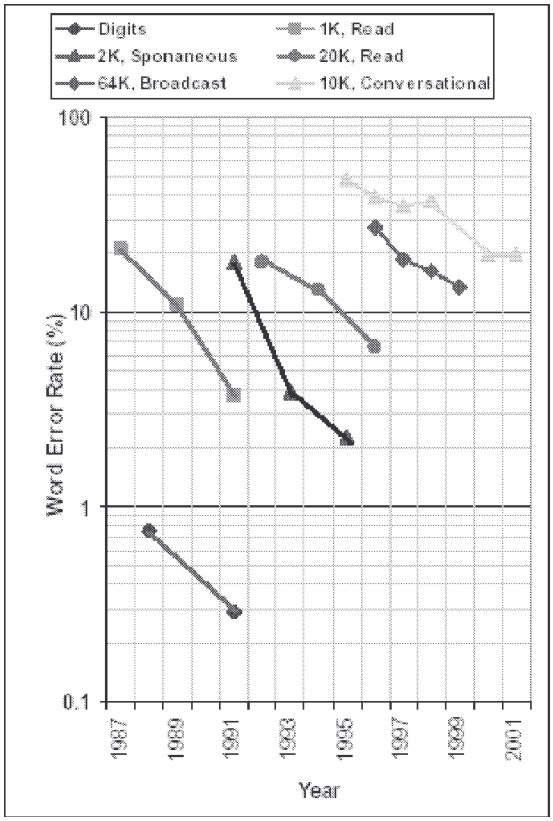


Figura 1.1: *Desempenho dos sistemas de reconhecimento em taxa de erro de palavra.*

1.1 Sistema de reconhecimento de fala

Um sistema de reconhecimento de fala é composto das seguintes partes [3](Figura 1.2):

- O módulo de extração de parâmetros converte o sinal acústico da fala em uma sequência de vetores acústicos. Tal conversão tem por objetivo remover as redundâncias do sinal de voz e extrair informações úteis para o reconhecimento (geralmente informações no domínio da frequência).
- O banco de dados armazena os padrões a serem reconhecidos. Estes podem ser palavras inteiras, no caso de vocabulários pequenos, ou subunidades fonéticas, no caso de vocabulários maiores. Em alguns tipos de aplicações

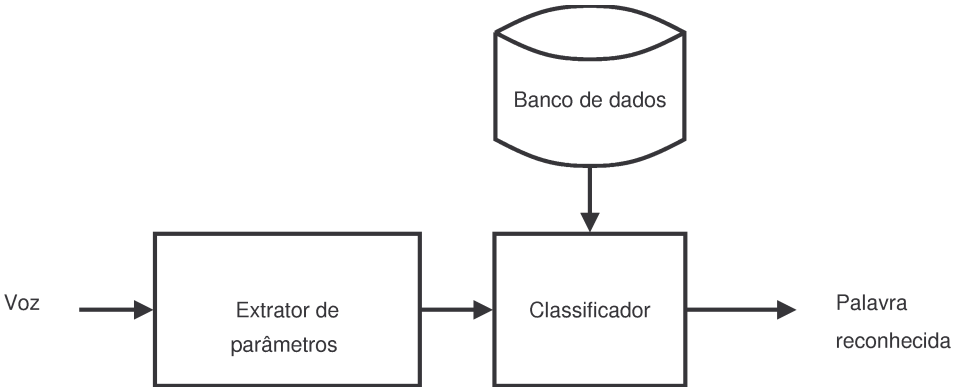


Figura 1.2: *Partes de um sistema de reconhecimento de fala.*

esta base de dados pode conter também informações lexicais, gramaticais e semânticas também.

- O classificador tem por função mapear a locução a ser reconhecida em um dos padrões armazenados no banco de dados. A saída deste é a saída do sistema em si.

Muitas pesquisas têm sido feitas em cada um destes blocos funcionais em busca de um melhor desempenho, tanto na taxa de acertos quanto no tempo de processamento. Neste trabalho, o foco será no bloco classificador, que será então descrito com maiores detalhes.

1.2 Classificadores

Dentre os classificadores existentes, os tradicionalmente utilizados em reconhecimento de fala são os Modelos Ocultos de Markov (*Hidden Markov Models* - HMM) [3] e as Redes Neurais (*Artificial Neural Networks* - ANN) [4]. Mais recentemente, sistemas usando Modelos de Misturas de Gaussianas (*Gaussian Mixture Models* - GMM) [5] e Máquinas de Vetor de Suporte (*Support Vector Machines* - SVM) [6] também foram propostos.

Evidentemente o uso de novos paradigmas só se justifica na medida em que apresentam melhores resultados ou modelamentos diferentes, que sejam promissores sob algum ponto de vista. A seguir será feita uma análise destes paradigmas com o objetivo de elucidar seus princípios.

Os HMMs fornecem um modelo estatístico da fala que modela, tanto as suas variações acústicas, quanto temporais. O critério utilizado para o treinamento destes modelos é o da maximização da verossimilhança (*Maximum Likelihood* - ML) do modelo gerar a locução correspondente. Logo, neste tipo de classificador,

só há a maximização do modelo que está sendo treinado, sem preocupação com os modelos concorrentes.

As redes neurais dão um passo além neste ponto: o critério de otimização utilizado é o da maximização a posteriori (*Maximum a Posteriori* - MAP), que consiste em maximizar a probabilidade do modelo correto, e simultaneamente minimizar a probabilidade dos modelos rivais. Com isto tem-se (pelo menos em teoria) um maior poder de discriminação. Entretanto, as redes neurais sofrem de um problema de sobreajuste aos dados de treinamento (*overfitting*): se treinada em excesso, a rede fica "viciada" nos dados de treinamento, e perde sua capacidade de generalização. Este problema tem sido contornado com conjuntos de validação cruzada, para monitorar a capacidade de generalização, durante o treinamento.

Já nas máquinas de vetor de suporte, além da minimização do erro de treinamento, leva-se em consideração a capacidade da máquina para minimizar a taxa de erro de generalização, obtendo assim maior resistência ao *overfitting*. Desta forma, esta técnica parece incorporar em seus fundamentos, otimizações para todos os problemas apresentados: discriminabilidade e capacidade de generalização. Este foi o principal motivador para a sua escolha neste trabalho.

1.3 Máquinas de Vetor de Suporte

Esta técnica de aprendizado de máquina foi criada por Vladmir Vapnik [7] e apresentada pela primeira vez na (*Conference on Computational Learning Theory*-COLT'92). Apesar disto, os elementos que a constituem já existiam antes e eram usados em aprendizado de máquinas, desde os anos 60. Entretanto, foi apenas em 1992 que todas essas características foram utilizadas juntas para formar um classificador de máxima margem, a SVM básica, e só em 1995 a versão de margens flexíveis foi introduzida por Cortes e Vapnik [8, 9]

Máquina de Vetor de Suporte (SVM, do inglês *Support Vector Machines* – SVM) é uma técnica que, além de poder ser aplicada ao reconhecimento de fala, tem sido utilizada com sucesso em diversas aplicações e vem recebendo bastante atenção nos últimos tempos. Isto porque, apesar de ter sido introduzida há poucos anos, esta técnica já apresenta um desempenho superior à maioria dos outros métodos em uma ampla variedade de aplicações. Em seu tutorial, Burges [10] apresenta diversos exemplos de aplicações.

Após as primeiras publicações, muitos pesquisadores se interessaram em trabalhar, tanto com os algoritmos, quanto com as análises teóricas destes sistemas. Isto criou uma nova direção de pesquisa, a qual mescla conceitos de disciplinas tão diferentes como otimização em aprendizado de máquina, estatística e análise

funcional. A extensão do algoritmo para o caso de regressão foi feita já no ano de 1995.[9] Uma boa e extensa base teórica sobre a área de SVMs pode ser encontrada no tutorial de Burges [10] e também em [7] e [11]. Foram feitos trabalhos de generalização do método, e extensões para o caso de múltiplas classes (Weston e Watkins [12]; Platt, Cristiannini e Shawe-Taylor [13]; Vapnik [7, 14]). Uma extensa bibliografia sobre SVM pode ser encontrada em [15].

SVMs são uma classe muito específica de algoritmos, caracterizada pelo uso de *kernels*, a ausência de mínimos locais e solução esparsa.

Quando aplicada ao reconhecimento de padrões, ao contrário de outras técnicas como Modelos Ocultos de Markov (HMM) e Redes Neurais (ANNs) que minimizam o erro de treinamento, a técnica de SVM baseia-se na minimização do risco estrutural que procura minimizar o limite superior do erro de generalização.

Logo a técnica de SVM é uma implementação do princípio de minimização estrutural de risco (SRM - *Structural Risk Minimization*).

1.4 Aplicações de SVM

Esta poderosa técnica pode ser usada para reconhecimento de fala, ou seja, classificação de padrões, como no caso desta tese, e ainda para regressão [16] que tem por exemplo a predição de séries temporais [17], entre outras aplicações. Além disso a técnica de SVM também pode ser usada para detecção de novidade (*Novelty Detection*)[18, 19, 20]. A detecção de novidade envolve o modelamento do comportamento normal de um sistema permitindo a detecção de qualquer divergência com a normalidade. Isto pode ser aplicado na detecção de defeitos em máquinas ou para destacar características anormais em dados médicos.

De outra forma pode-se dizer que a detecção de novidade é a habilidade para detectar uma nova classe ou sub-classe, podendo ser uma característica útil em um sistema de aprendizagem com inteligência artificial. Ligeiras modificações na distribuição dos dados podem indicar, por exemplo, uma nova classe ou dado ou uma modificação no padrão de uma classe que já foi modelada.

No caso de classificação, a técnica de SVM fez muito sucesso em áreas como: reconhecimento de face [21], classificação de textos [22, 23] e reconhecimento de dígitos escritos à mão [24] e em outras inúmeras aplicações. Uma lista de possíveis aplicações de SVM pode ser encontrada em [25].

Em muitas aplicações a SVM mostrou resultados acima dos métodos já estabelecidos como: redes neurais e funções de base radial. Vide ainda [8] e [10] para outros exemplos.

Nos anos que se seguiram após a criação do SVM, ocorreram inúmeros avanços, tanto com relação às aplicações, quanto com relação à melhoria da técnica.

Algumas das aplicações de SVM na área de reconhecimento de fala :

- Reconhecimento de fala contínua usando Máquinas de Vetor de Suporte :

  Em 2000, Clarkson apresenta, em um artigo, a técnica de SVM sendo usada para a classificação fonética pela primeira vez [26]. Foi feita também a classificação de vogais. O banco de vozes utilizado foi o TIMIT, sendo que o treinamento foi feito utilizando as sentenças 'sx' e 'si', obtendo assim 3696 locuções de 168 locutores. Para teste foram utilizadas 192 locuções de 24 locutores não incluídos no conjunto de treinamento.

  No mesmo ano, Picone e co-autores [27] utilizaram um híbrido HMM/SVM para o reconhecimento de voz e conseguiram um desempenho de 11,6 % (taxa de erro de teste), sendo que com um sistema com HMM, mistura de gaussianas e utilizando trifones, o desempenho foi de 12,7 %. No sistema híbrido foi utilizado apenas um quinto dos dados utilizados pelo sistema HMM. O banco de vozes utilizado foi o OGI *Alphadigits corpus* que tem um vocabulário de 36 palavras (26 letras e 10 dígitos), faladas através de um telefone e com independência de locutor. O conjunto de treinamento total é composto de 46.730 sentenças com 6 palavras em média por sentença, mas no sistema híbrido foi utilizado apenas 9.000 sentenças no conjunto de treinamento e 1000 no de teste.

- Medidas de confiança para *word spotting* (detecção de palavra chave) usando SVM [28]:

  Benayed e co-autores propuseram um método para *word spotting* utilizando uma forma alternativa de detecção de palavras que não pertençam ao vocabulário utilizado no sistema. Para isto, medidas de confiança são calculadas através de informações a nível fonético, fornecidas por um HMM (probabilidade posterior de cada quadro da locução). Foi utilizado para treinamento uma parte do banco de vozes SPEECHDAT [29]. Diversas línguas fazem parte deste banco, sendo que foi utilizada a parte em francês. Ele foi gravado através da rede telefônica a 8 kHz e contém 8800 sentenças, pronunciadas por 800 locutores e para teste foram usadas 4780 sentenças, pronunciadas por 1000 locutores (diferentes dos locutores de treinamento).

- Segmentador automático de fala utilizando SVM [30]:

  Foi proposto aqui um método que combina características fonéticas e SVM para fazer a segmentação de fala contínua em cinco classes : vogal, consoante sonora, fricativa, ruído de parada e silêncio. Foi utilizado o banco de vozes TIMIT.

- Reconhecimento de fala através de imagens produzidas durante a emissão dos fones [31]:

  Nesta aplicação o reconhecimento de fala é feito através de imagens. Cada palavra do vocabulário é modelada por uma seqüência de imagens as quais são classificadas com SVM. Foi utilizado o banco de vozes audiovisual da língua inglesa TULIPS [32].

1.5 Objetivo e estrutura da tese

O objetivo desta tese é o estudo teórico, implementação e teste de um sistema de reconhecimento de palavras isoladas, independente de locutor, e vocabulário pequeno, usando Máquinas de Vetor de Suporte (SVM).

Foram realizados testes com os *kernels* função de base radial gaussiana (Radial base function - RBF) e polinomial. As aproximações "um vs todos" e "um vs um" foram utilizadas para trabalhar com múltiplas classes. E para classificação foram utilizadas as aproximações *max win* (máximo de vitórias), DAG(*Directed Acyclic Grafh*), e uma nova, inspirada em DAG, como contribuição original desta tese.

A estrutura da dissertação é a seguinte: o Capítulo 2 trata sobre algumas definições necessárias ao entendimento da tese, o Capítulo 3 trata sobre SVMs, o Capítulo 4 sobre *kernels*, o Capítulo 5 sobre os resultados das simulações e *toolboxes* utilizadas e o Capítulo 6 das conclusões e possíveis extensões deste trabalho.

Capítulo 2

Teoria do Aprendizado Estatístico

Neste capítulo serão dadas algumas definições necessárias ao entendimento de Máquinas de Vetor de Suporte.

2.1 Teoria do Aprendizado Estatístico

A teoria do aprendizado estatístico, desenvolvida por Vapnik [7], define os conceitos em que a técnica de SVM se baseia, além de demonstrar matematicamente os principais resultados.

2.1.1 Dimensão VC

Assim denominada em homenagem a seus criadores Vapnik e Chervonenkis, a dimensão VC pode ser definida como a medida da capacidade (ou complexidade) de uma classe de funções que é, por definição, o maior número de amostras de treinamento que possam ser corretamente classificadas. A denominação "classe de funções"é só uma outra forma de se referir a uma "máquina de aprendizado" [10].

O exemplo para o caso de uma reta (pertencente à classe das funções lineares) é mostrado na figura (2.1) [10]. Verifica-se pela figura que a dimensão VC neste caso é 3, pois este é o maior número de amostras que podem ser separadas por uma reta, para qualquer padrão de classificação binária que as amostras podem admitir. Por indução, a dimensão VC para funções lineares no $\Re^n$, com $n \geq 2$, é $n+1$.

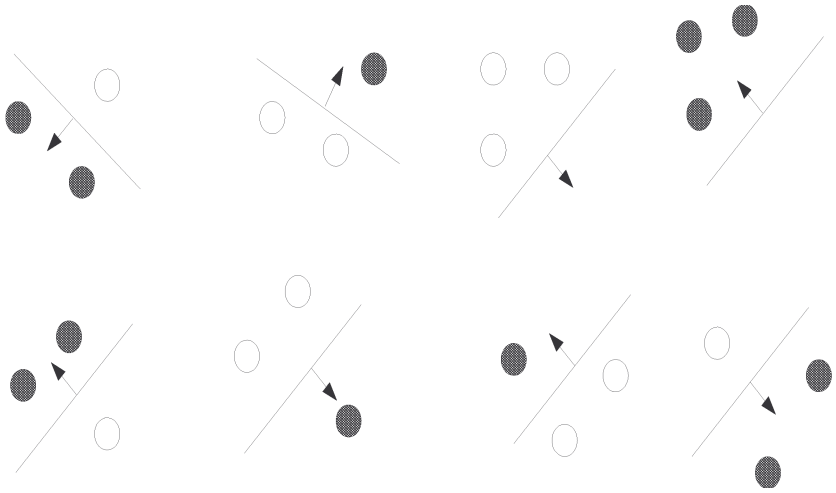


Figura 2.1: *Possíveis separações de três pontos por uma reta (o hiperplano neste caso). Se $n = 2$ for a dimensão dos pontos de dados, a dimensão VC será igual a $n + 1 = 3$.*

O conceito de capacidade da máquina (ou dimensão VC) está ilustrado na figura (2.2). Na figura (2.2 A) o conjunto de treinamento foi classificado, por exemplo, por uma função linear, na figura (2.2 B), por uma função polinomial e na figura (2.2 C), por uma função de base radial gaussiana. Estas funções serão vistas em detalhes no capítulo sobre *Kernels*. O exemplo (A) apresenta a menor capacidade, boa generalização e o maior erro de treinamento; o exemplo(B) tem capacidade intermediária, boa generalização e além disso apresenta um erro de treinamento pequeno; e o exemplo (C), apesar de ter alta capacidade e o menor erro de treinamento, tem baixa generalização.

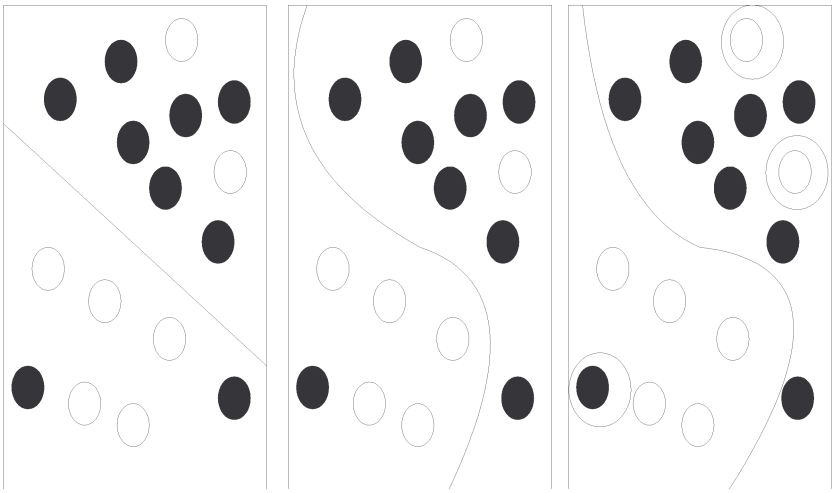


Figura 2.2: *Exemplos de classificação binária com diferentes funções.*

Diferentes classes de funções (lineares, exponenciais, polinomiais, etc) têm diferentes capacidades (diferentes valores de dimensão VC). Mais capacidade indica que é possível construir máquinas mais complexas, mas com tendência ao *overfitting* (sobreajuste ou ajuste excessivo, que causa perda de generalização) e menos capacidade indica que não haverá *overfitting*, mas haverá muita restrição no que aquela máquina pode fazer. O objetivo, então, é obter um compromisso entre a capacidade e a generalização da máquina. Logo, a chave para resolver esta questão é utilizar a minimização estrutural do risco para encontrar um valor ótimo para esta capacidade.

Antes de abordar a minimização estrutural do risco será necessário definir alguns conceitos.

2.1.2 Minimização do risco empírico

Seja uma máquina de aprendizado com um conjunto de parâmetros ajustáveis α e um conjunto de treinamento dado pelos pares $(\mathbf{X_1}, y_1)$,$(\mathbf{X_2}, y_2)$,...,$(\mathbf{X_N}, y_N)$, onde $\mathbf{X_1}$, $\mathbf{X_2}$, ..., $\mathbf{X_N}$ representam os vetores (amostras) de entrada e y_1, y_2, ..., $y_N \in [-1, +1]$ suas respectivas classes. Em um problema de classificação envolvendo duas classes, também chamado de problema de classificação binária, a máquina deve encontrar valores de α de modo a aprender o mapeamento $\mathbf{X} \mapsto y$. Isto irá resultar em um possível mapeamento $\mathbf{X} \mapsto f(\mathbf{X}, \alpha)$ que define a máquina. Logo, tem-se que encontrar uma função $f(\mathbf{X}, \alpha)$ que minimize o seguinte funcional de risco:

$$R(\alpha) = \int |y_i - f(\mathbf{X_i}, \alpha)| dP(\mathbf{X}, y) \tag{2.1}$$

onde $|y_i - f(\mathbf{X_i}, \alpha)|$ é uma função de perda para as amostras de treinamento incorretamente classificadas; $R(\alpha)$ é o risco esperado, o qual mede a capacidade que uma determinada função $f(\mathbf{X_i}, \alpha)$ tem para predizer a classe y de uma amostra $\mathbf{X}$; e $P(\mathbf{X}, y)$ é a distribuição de probabilidade conjunta.

Entretanto, não é possível calcular o funcional de risco $R(\alpha)$ pela equação (2.1), pois a distribuição de probabilidade conjunta $p(\mathbf{X}, y)$ não é conhecida. Devido a isto, foi utilizada uma estimativa de risco baseada nos dados de treinamento, chamada de funcional de risco empírico, que pode ser calculada pela minimização da seguinte expressão:

$$R_{emp}(\alpha) = \frac{1}{N} \sum_{i-1}^{N} |y_i - f(X_i, \alpha)| \tag{2.2}$$

onde N é o tamanho do conjunto de treinamento, α o conjunto de parâmetros ajustáveis e R_{emp} é o risco empírico (erro de treinamento), calculado para um

número fixo de amostras, supondo que estas amostras sejam independentes e identicamente distribuídas (iid), de acordo com uma distribuição de probabilidade $P(\mathbf{X}, y)$.

Pode-se observar que a distribuição de probabilidade conjunta $P(\mathbf{X}, y)$ não aparece na equação (2.2).

Este princípio de minimização de risco é chamado de Minimização do Risco Empírico (ERM - *Empirical Risk Minimization*) e é um dos procedimentos mais comumente usados em aprendizado de máquina. Apesar disto ter conduzido a classificadores eficientes, há um problema com este tipo de procedimento: se a capacidade da máquina é alta (se ela consegue classificar sem erros uma grande quantidade de amostras de treinamento), haverá uma tendência ao ajuste excessivo com os dados de treinamento(*overfitting*). Neste caso a máquina perderá sua capacidade de generalização. Ou seja, neste caso, ela só consegue classificar corretamente as amostras que forem iguais às do conjunto de treinamento. Logo, a medida da qualidade de uma máquina de aprendizado não é indicada em sua totalidade, quando só a minimização do risco empírico é considerada.

A análise a ser feita agora é verificar em que condições a estimativa dada por $R_{emp}(\alpha)$ na equação (2.2) se aproxima do valor desejado $R(\alpha)$, dado pela equação (2.1).

Pela lei dos grandes números, o valor de $R_{emp}(\alpha)$ converge em probabilidade para o valor esperado $R(\alpha)$, quando o tamanho do conjunto de treinamento N tende para ∞ [7]. Desta forma, a consistência da minimização do risco empírico baseia-se na suposição de que o risco esperado $R(\alpha)$ e o risco empírico $R_{emp}(\alpha)$ convergem em probabilidade para o mínimo valor de risco $R(\alpha)_{min}$.

Logo, a formulação de que o risco esperado $R(\alpha)$ e o risco empírico $R_{emp}(\alpha)$ convergem em probabilidade para o mínimo valor de risco $\inf(R(\alpha))$, fica:

$$\begin{cases} P(|R_{emp}(\alpha) - \inf(R(\alpha))|) > \varepsilon \rightarrow 0 \text{ quando } N \rightarrow \infty \\ P(|R(\alpha) - \inf(R(\alpha))|) > \varepsilon \rightarrow 0 \text{ quando } N \rightarrow \infty \end{cases} \tag{2.3}$$

onde inf é o ínfimo, e indica o maior dos limitantes inferiores.

Vapnik [7] mostra que a consistência será válida se e somente se a convergência em probabilidade for substituída pela convergência uniforme em probabilidade (a dedução detalhada destas equações também pode ser vista em [9]).

É possível impor uma condição estrita de que a relação probabilística

$$P(\sup|R_{emp}(\alpha) - R(\alpha)| > \varepsilon) \rightarrow 0 \text{ quando } N \rightarrow \infty \tag{2.4}$$

é válida. Nesta equação, sup indica o menor dentre os limitantes superiores. Se a equação (2.4) é satisfeita, ocorre a convergência uniforme das variáveis α do risco

empírico $R_{emp}(\alpha)$ para o seu valor esperado $R(\alpha)$.

2.2 Minimização do Risco Estrutural

Também foi mostrado por Vapnik que a condição necessária e suficiente para consistência do princípio de minimização do risco empírico é a limitação da dimensão VC de uma classe de funções $f(\alpha)$.

Um limitante superior válido para a variação do risco esperado $R(\alpha)$ em função do risco empírico $R_{emp}(\alpha)$ é, com probabilidade de pelo menos $1-\eta$, dado por

$$R(\alpha) \leq R_{emp} + \sqrt{\frac{h(\log(2N/h)+1) - \log(\eta/4)}{N}} \tag{2.5}$$

onde N é o número de amostras de treinamento, η o nível de confiança e h é um escalar inteiro positivo, chamado de dimensão VC (Vapnik - Chervonenkis) [7, 33] e mede a capacidade da máquina, ou seja, o maior número de amostras de treinamento que podem ser corretamente classificadas com uma determinada função. Esta equação é proveniente da teoria de convergência uniforme.

Da equação (2.5) pode-se verificar que para minimizar o risco esperado, e conseqüentemente melhorar o desempenho de generalização da máquina, a soma do risco empírico e da razão h/N (referente ao primeiro e ao segundo membro do lado direito da equação) tem que ser minimizada. Nesta observação baseia-se o princípio de minimização do risco estrutural (princípio SRM - *Structural Risk Minimization*).

Seja uma estrutura na qual o conjunto com possíveis classes (com todas as hipóteses) foi dividido em subconjuntos da seguinte forma:

$$F_1 \subset F_2 \subset ... \subset F_k \subset ..., \tag{2.6}$$

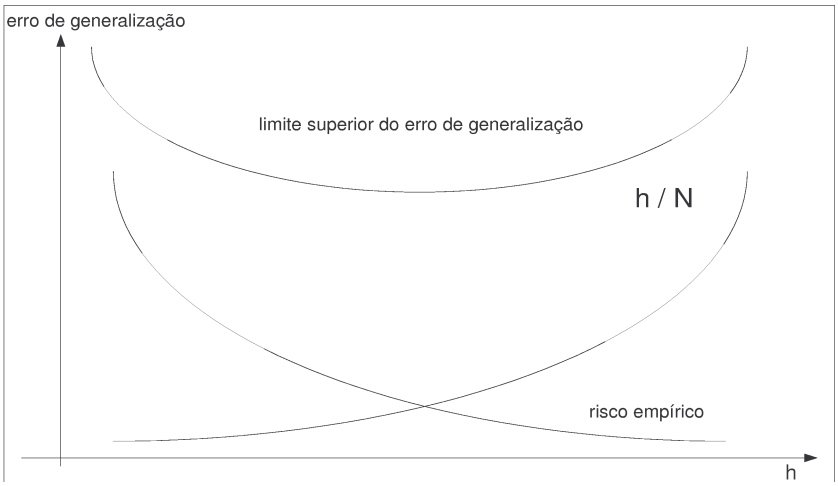
onde h_k é a dimensão VC de cada subconjunto F_k, com a propriedade $h_k \leq h_{k+1}$.

A minimização estrutural do risco consiste em encontrar o subconjunto de funções que minimiza o limite superior de risco (erro de generalização). Isto pode ser feito, treinando-se uma série de máquinas, uma para cada subconjunto, onde o objetivo é simplesmente minimizar o risco empírico [10]. Será escolhida a máquina em que a soma do risco empírico e da razão h/n for a menor.

O termo h/n indica que a capacidade da máquina é diretamente proporcional a dimensão VC, representada por h, e inversamente proporcional ao número de amostras de treinamento N.

A Figura (2.3) ilustra a utilização da minimização do risco estrutural. Na figura, o eixo horizontal corresponde à dimensão VC e o eixo vertical ao erro de generalização. Pode-se observar que o limite superior de risco será minimizado

quando a soma da razão h/N e do erro de treinamento (risco empírico) for minimizada. Pela tabela que acompanha o gráfico [34], pode-se observar a utilização da minimização do risco estrutural para escolher a melhor máquina de aprendizado com a dimensão VC. O termo f_i foi explicado na equação (2.6) e refere-se a um subconjunto.



i	f_i	risco empírico ou erro de treinamento	h / N	limite superior do erro de generalização	escolha
1	f_1				
2	f_2				
3	f_3				
4	f_4				
5	f_5				
6	f_6				

Figura 2.3: *Ilustração do princípio de minimização do risco estrutural. Na tabela a variável i indica o índice da função f_i, h é a dimensão VC e N o número de amostras de treinamento.*

Capítulo 3

Máquinas de Vetor de Suporte

3.1 Hiperplano ótimo para padrões linearmente separáveis

Inicialmente será mostrada a classificação de apenas duas classes de dados linearmente separáveis (i.e. podem ser separados por um hiperplano sem que ocorram erros de classificação) como ilustrado na Figura 3.1.

Em $\Re^2$ o hiperplano pode ser, por exemplo, uma reta (Figura 3.1) e em $\Re^3$, um plano comum ou uma superfície (Figura 3.3).

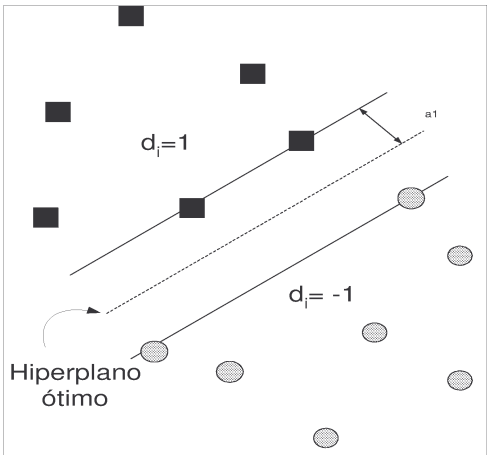
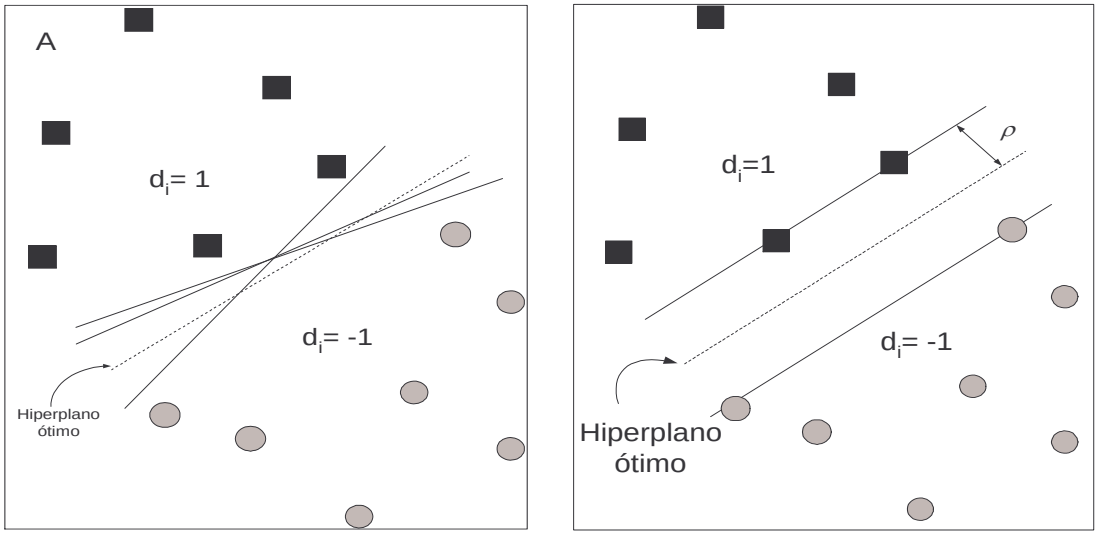


Figura 3.1: *Hiperplano ótimo para padrões linearmente separáveis.*

Pode-se observar pela Figura 3.1 que as amostras de treinamento são bidimensionais e já foram previamente classificadas. Estas amostras são dadas por $(\mathbf{X_i},\ d_i)_{i=1}^{l}$, onde $\mathbf{X_i}$ representa uma amostra, d_i sua respectiva classe e l o número



(a) Vários hiperplanos são possíveis.

(b) Hiperplano ótimo e hiperplanos canônicos.

Figura 3.2: *Escolha do hiperplano ótimo. Os hiperplano que estão a uma distância ρ do hiperplano ótimo são chamados de hiperplanos canônicos.*

de amostras. Pode-se observar também que os dados podem ser separados por uma reta r de equação geral $ax + by + c = 0$.

O conceito de hiperplano ótimo é ilustrado na Figura 3.2. Pode-se observar na Figura 3.2a que existem vários possíveis hiperplanos que resolvem o problema de classificação. Como o objetivo é encontrar o hiperplano que classifique bem futuras amostras, pode-se escolher, de forma intuitiva, o hiperplano ótimo como aquele que se localiza a uma distância igual entre as duas classes. Logo, o hiperplano ótimo será o que maximiza a distância entre o hiperplano e cada uma das classes (maximização da margem de separação). Na Figura 3.2b, a variável ρ representa a distância entre o hiperplano ótimo e a amostra mais próxima. A distância entre as classes é dada por 2ρ. Desta forma, a possibilidade de erros na predição de futuras amostras será minimizada.

Um exemplo com dados em três dimensões é dado na Figura 3.3. Neste caso verifica-se que será necessário uma superfície para separar os dados. Logo, se n é a dimensão do espaço na qual o hiperplano está, a dimensão do hiperplano será $n - 1$.

Na Figura 3.3, $g(x)$ representa o hiperplano pois, segundo Duda & Hart [35], o hiperplano pode ser considerado como uma função discriminante $g(x)$ que divide o espaço (em que estão as amostras) em duas regiões dadas por $g(x) > 0$ e $g(x) < 0$, cada região caracterizando uma classe. Se a amostra cair sobre o hiperplano $g(x) = 0$, define-se a qual classes ela pertencerá pela notação $g(x) \geq 0$

e $g(x) < 0$ ou ainda $g(x) > 0$ e $g(x) \leq 0$.

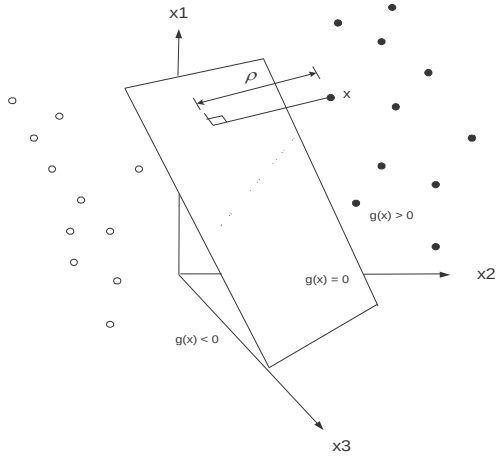


Figura 3.3: *Um hiperplano com duas dimensões é necessário para se classificar dados com três dimensões.*

Na Figura 3.2, pode-se observar que a margem de separação 2ρ é a menor distância entre amostras de classes diferentes. A maximização desta margem é o objetivo a ser alcançado no treinamento da SVM, ou seja, a maximização de ρ. Serão feitas, então, algumas considerações com o objetivo de calcular ρ, o que equivale a calcular a distância entre a amostra mais próxima do hiperplano, representada por um ponto $P(x_0, y_0)$, e o hiperplano, representado por uma reta r de equação $ax + by + c = 0$ (Figura 3.4).

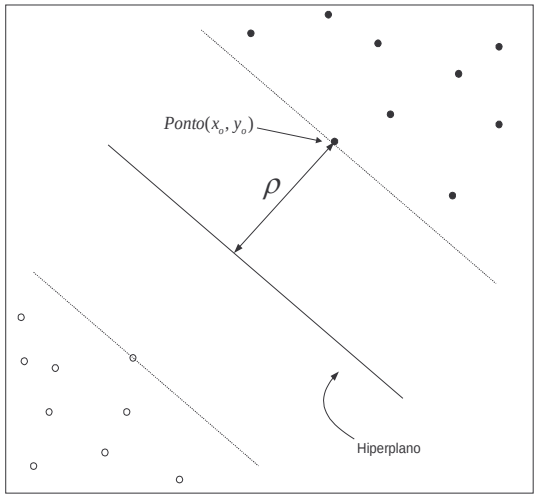


Figura 3.4: *Distância entre o hiperplano e a amostra mais próxima.*

Fazendo uma análise do ponto de vista geométrico, a equação que calcula a distância do ponto à reta é dada por:

$$d = \frac{|ax_o + by_o + c|}{\sqrt{a^2 + b^2}} \tag{3.1}$$

A partir da equação (3.1) pode-se calcular que a distância da origem (ponto) até o hiperplano (reta) é

$$d = \frac{b_0}{\|\mathbf{W}\|} \tag{3.2}$$

onde b_0, neste caso, é o *bias* ótimo e $\|\mathbf{W}\|$ é o modulo do hiperplano ótimo. Esta distância está representada na Figura 3.5.

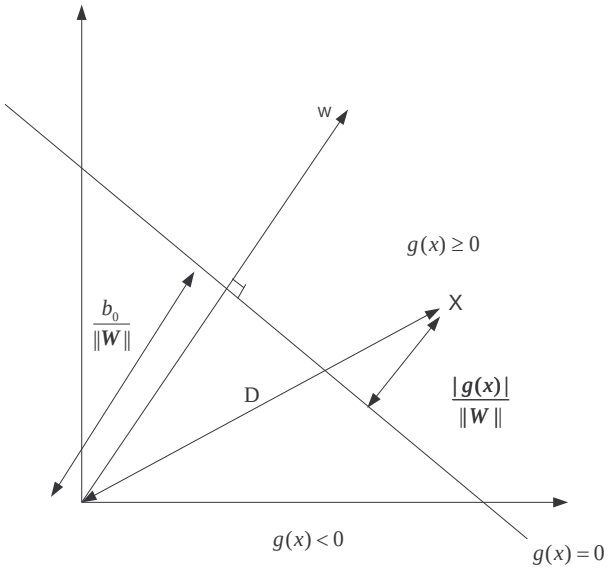


Figura 3.5: *Distância D de uma amostra $\mathbf{X}$ até a origem. Distância $b_o/\|\mathbf{W}\|$ do hiperplano à origem e distância $g(x)/\|\mathbf{W}\|$ da amostra ao hiperplano.*

A equação de um hiperplano é dada por

$$\mathbf{W}^T\mathbf{X} + b = 0, \tag{3.3}$$

onde $\mathbf{X}$ é o vetor de entrada, $\mathbf{W}$ é um vetor de pesos ajustável que será calculado durante o treinamento da máquina e b é o *bias* (distância do hiperplano até a origem). [36]

Vapnik [7] mostrou que, após o cálculo do hiperplano ótimo (definição de $\mathbf{W_o}$ e b_o), todas as amostras satisfazem as seguintes inequações:

$$\begin{aligned} \mathbf{W}_o^T\mathbf{X} + b_o \geq 0 \text{ se } d_i = +1 \\ \mathbf{W}_o^T\mathbf{X} + b_o \leq 0 \text{ se } d_i = -1 \end{aligned} \tag{3.4}$$

O sinal de igualdade foi incluído nas inequações para indicar que se a equação do hiperplano é igual a 1 ou -1, a amostra se localiza sobre um dos hiperplanos canônicos e pertence a uma das classes (o que caracteriza um hiperplano canônico é o fato dele se localizar a uma distância ρ do hiperplano ótimo).

Estas duas inequações podem ainda ser unidas em uma só inequação:

$$d(W^T X_i + b) \geq 1 \text{ para } i = 1,...,N \tag{3.5}$$

Segundo esta inequação a amostra mais próxima do hiperplano ótimo, denominada vetor suporte, está à uma unidade de distância do mesmo. Desta forma, tem-se:

$$\rho = \frac{g(x)}{\|\mathbf{W}\|} = \frac{|\mathbf{W}^T\mathbf{X} + b|}{\|\mathbf{W}\|} = \begin{cases} \frac{1}{\|\mathbf{W}_0\|} & \text{se } d_i = 1 \\ \frac{-1}{\|\mathbf{W}_0\|} & \text{se } d_i = -1 \end{cases} \tag{3.6}$$

onde $||.||$ é a norma euclidiana, ou norma dois, do vetor de pesos, ou em termos mais familiares, é o módulo do vetor de pesos e $\mathbf{W}_0$ representa o vetor de pesos ótimo. Assim, ρ fornece uma medida algébrica da distância de uma amostra $\mathbf{X}$ até o hiperplano ótimo.

Logo, a margem de separação mede

$$2\rho = \frac{2}{\|\mathbf{W_o}\|} \tag{3.7}$$

e o hiperplano ótimo será aquele que minimiza

$$\Phi(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|^2 \tag{3.8}$$

onde a constante $1/2$ foi incluida por conveniência de representação.

Pode-se observar que, na Equação (3.8), escolheu-se minimizar $\|\mathbf{W}\|^2$ que é uma função convexa no lugar de $\|\mathbf{W}\|$ [6]. Isto se deve ao fato de que existem técnicas computacionais eficientes para a otimização de funções convexas e suas restrições [37, 38], como a resolução através dos multiplicadores de Lagrange, utilizados neste trabalho.

Antes de continuar, será feita a definição de função convexa. A Figura 3.6 ilustra o conceito de função convexa. Pode-se observar que, no caso de uma função convexa, uma reta ligando dois pontos desta função ficará sempre acima da curva. Uma definição mais formal é dada a seguir.

Uma função é convexa se $f(\lambda y + (1-\lambda)z) \leq \lambda f(y) + (1-\lambda)f(z)$ para qualquer y e z e para $0 \leq \lambda \leq 1$. Por exemplo, na Figura, é válida a seguinte inequação: $f((y+z)/2) \leq f(y)/2 + f(z)/2$.

Feita a definição de função convexa, a fundamentação teórica sobre SVM

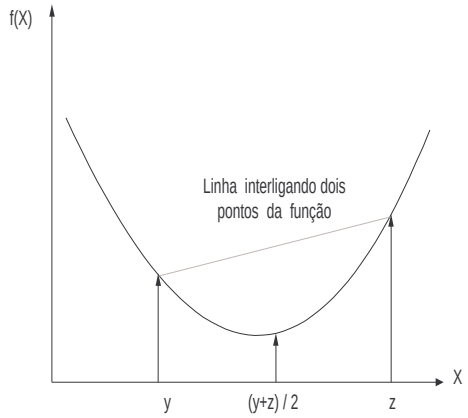


Figura 3.6: *Definição de função convexa. Uma linha interligando dois pontos quaisquer de uma função convexa ficará sempre acima da função.*

prosseguirá.

A minimização da Equação (3.8) é equivalente à implementação do princípio SRM (minimização estrutural do risco). Para analisar a aplicação deste princípio na escolha do hiperplano ótimo, considera-se uma variável A como o maior valor que a norma do vetor $\mathbf{W}$ pode assumir [7], ou seja,

$$\|\mathbf{W}\| \leq A \tag{3.9}$$

Logo, a maior distância do hiperplano até uma amostra é limitada por

$$\rho \geq \frac{1}{A} \tag{3.10}$$

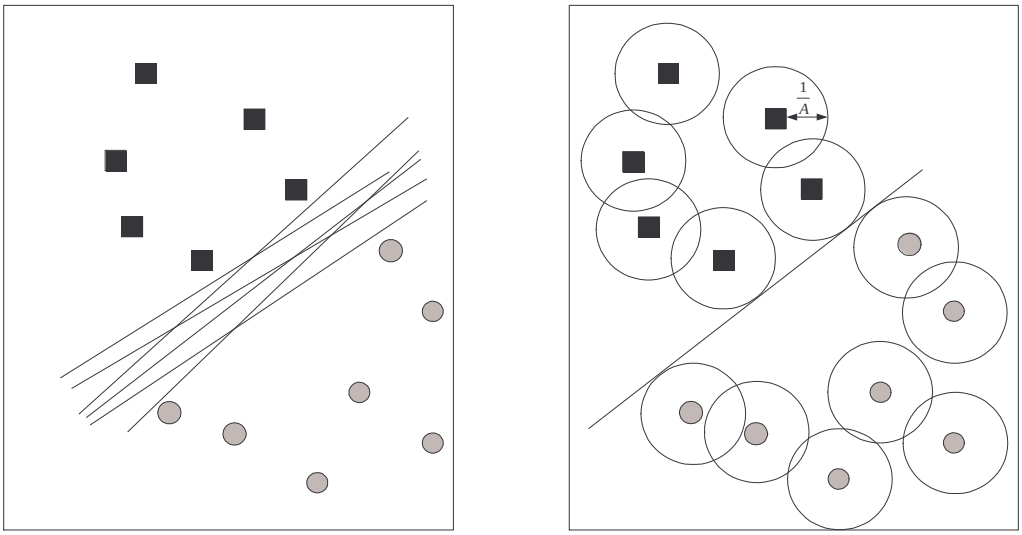
Desta forma, o hiperplano ótimo não poderá ficar a uma distância menor que $\frac{1}{A}$ de uma amostra. A Figura 3.7 ilustra esta idéia. Observa-se que na Figura 3.7a, vários hiperplanos são possíveis. Entretanto, na Figura 3.7b pode-se verificar que apenas um destes hiperplanos satisfaz a restrição dada na Equação (3.10). A melhor generalização possível é conseguida por esta escolha.

Os vetores de suporte neste caso serão aqueles que estão a uma distância de $1/A$ do hiperplano.

Vapnik [7] utiliza o valor de A, além do raio da menor hiperesfera que circunda os dados de treinamento R, para formular o limite superior da dimensão VC, dada por h:

$$h \leq \min(R^2A^2, n) + 1 \tag{3.11}$$

onde n é a dimensão do espaço em que se localiza o hiperplano.



(a) (b)

Figura 3.7: *A distância mínima entre uma amostra e o hiperplano ótimo deve ser de* $1/A$.

Como a variável A refere-se ao valor de $\|\mathbf{W}\|$, a Equação (3.11) fica:

$$h \leq \min(R^2\|\mathbf{W}\|^2,\ n) + 1 \tag{3.12}$$

Esta equação aplica o princípio SRM na seleção de uma máquina de aprendizado.

Pela Equação (3.3) pode-se verificar, que para obtenção do hiperplano ótimo, é necessário o cálculo do vetor de pesos ótimo $\|\mathbf{W_o}\|$ e do *bias* ótimo b_o referentes às amostras de treinamento.

Pela análise das equações (3.4) e (3.6), observa-se que o vetor de pesos e o *bias* ótimo devem satisfazer à restrição: [36]

$$\begin{aligned} \mathbf{W}_0^T\mathbf{X}_i + b_0 &\geq 1, \text{ se } d_i = 1 \\ \mathbf{W}_0^T\mathbf{X}_i + b_0 &\leq -1, \text{ se } d_i = -1 \end{aligned} \tag{3.13}$$

As amostras (ou vetores de entrada) que satisfazem uma ou outra linha da Equação (3.13) são denominadas de vetores de suporte, como mostrado na Figura 3.8. Assim surgiu o nome "Máquina de Vetor de Suporte". Estas são as amostras que mais perto se encontram da superfície de decisão (hiperplano) e que definem a localização ótima desta superfície, além de serem as amostras mais difíceis de classificar [36].

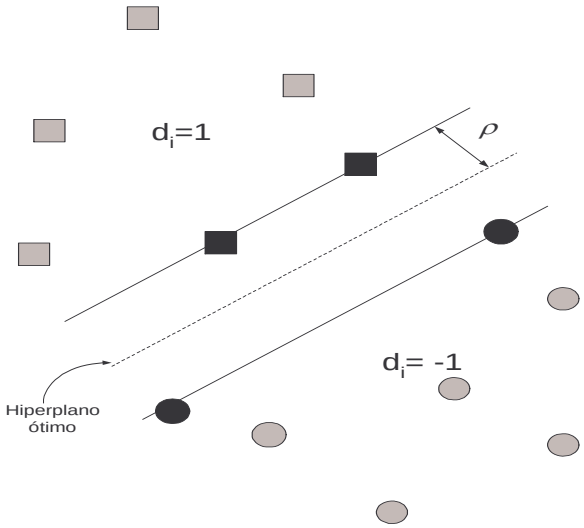


Figura 3.8: *Hiperplano ótimo para padrões linearmente separáveis. Os vetores suporte estão em preto. A distância ρ representa a margem de separação. A distância mínima entre dois pontos de dados de classes diferentes é 2ρ para este caso mais simples de classes linearmente separáveis. Os hiperplanos que passam pelos vetores suporte são chamados de hiperplanos canônicos.*

Tendo obtido o hiperplano ótimo, o vetor de pesos $\mathbf{W}_o$ fornece a máxima separação possível entre duas classes de exemplos. Pode-se, então, alcançar esta condição ótima minimizando a norma euclidiana do vetor de pesos $\mathbf{W}$, o que significa minimizar o módulo deste mesmo vetor de pesos.

O problema de otimização restrito que resolve este problema pode ser escrito da seguinte maneira: [36, 39]

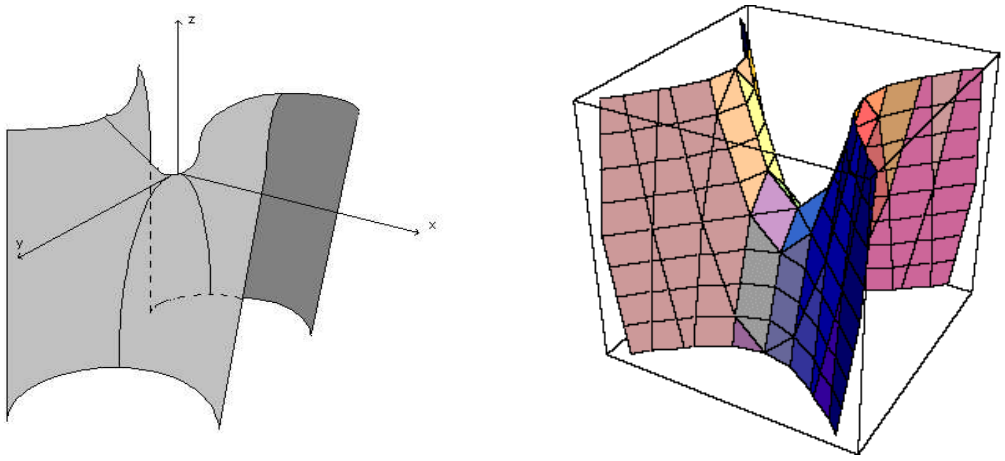
Sejam os dados de treinamento (que neste primeiro caso devem ser linearmente separáveis) $(\mathbf{x}_i,\ d_i),\ 1 \leq i \leq N,\ \mathbf{X}\ \epsilon\ \Re^m,\ d_i\ \epsilon\ \{+1, -1\}$, onde $\mathbf{X}$ são os dados de entrada e d_i corresponde a resposta desejada (classe do respectivo dado de treinamento). Deseja-se calcular o vetor de pesos $\mathbf{W}$ e o bias (ou intercepto) b que resolvam o seguinte problema:

$$\begin{aligned} &\text{Minimizar: } \Phi(\mathbf{W}) = \frac{1}{2}\mathbf{W}^T\mathbf{W} \\ &\text{sujeito a: } d_i(\mathbf{W}^T\mathbf{X}_i + b) \geq 1 \text{ para } i = 1, 2, ..., N \end{aligned} \tag{3.14}$$

onde o escalar 1/2 foi incluído por conveniência de representação.

Assim, aplicando o método dos multiplicadores de Lagrange será possível transformar este problema de otimização primal em seu correspondente dual [40].

O funcional para este problema de otimização é chamado de Lagrangeana e



(a) (b)

Figura 3.9: *Devido ao formato da superfície ser o de uma sela, o ponto* $(0,0)$ *é chamado de ponto de sela. Esta superfície é dada por* $ax^2 - by^2 = cz$.

pode ser escrito como:

$$L_P(\mathbf{W}, b, \alpha) = \frac{1}{2}\|\mathbf{W}\|^2 - \sum_{i=1}^{N} \alpha_i d_i(\mathbf{W}^T \mathbf{X_i}) + \sum_{i=1}^{N} \alpha_i \tag{3.15}$$

As variáveis α_i são chamadas de multiplicadores de Lagrange e só assumem valores não negativos. A resolução do problema de otimização por Lagrange permite a descrição dual que normalmente tem uma resolução mais fácil que a descrição primal, a qual apresenta restrições de desigualdade de difícil solução. Isto se tornou padrão na teoria de Máquinas de Vetor de Suporte, pois permite trabalhar em um espaço de alta dimensionalidade. Isto é possível porque o número de parâmetros a serem ajustados não depende da dimensão dos dados de entrada [36].

A Equação (3.15) é um problema de otimização restrito e sua solução pode ser determinada pelo ponto de sela da Lagrangeana (Figura 3.9a). Isto é realizado minimizando a função em relação a $\mathbf{W}$ e b e maximizando-a em relação a α. Isto pode ser melhor entendido pela Figura 3.9b [41].

Para realizar estes cálculos é necessário diferenciar a função $L_P(\mathbf{W}, b, \alpha)$ em relação a $\mathbf{W}$ e b e igualar os resultados a zero. [36]

$$\frac{\partial L}{\partial \mathbf{W}}(\mathbf{W}, b, \alpha) = \mathbf{W} - \sum_{i=1}^{N} \alpha_i d_i \mathbf{x}_i = 0 \tag{3.16}$$

$$\frac{\partial L}{\partial b}(\mathbf{W}, b, \alpha) = \sum_{i=1}^{N} d_i \alpha_i = 0 \tag{3.17}$$

Das equações (3.16) e (3.17) tem-se:

$$\mathbf{W} = \sum_{i=1}^{N} d_i \alpha_i \mathbf{X}_i \tag{3.18}$$

$$\sum_{i=1}^{N} d_i \alpha_i = 0 \tag{3.19}$$

Por definição o vetor de pesos $\mathbf{W}$ é originado a partir de uma expansão envolvendo todos os exemplos de treinamento. Apesar da solução ser única, devido à convexidade da Lagrangeana, isto não ocorre com os coeficientes de Lagrange. Pode-se mostrar que no ponto de sela o produto de um multiplicador de Lagrange pela sua respectiva restrição é nulo [36]:

$$\alpha_i[d_i(\mathbf{W}^T\mathbf{X}_i + b) - 1] = 0 \text{ para } i = 1, 2, ..., N \tag{3.20}$$

Logo só poderão assumir valores diferentes de zero os coeficientes de Lagrange que satisfaçam a Equação (3.20), propriedade esta resultante das condições de Kuhn-Tucker da teoria da otimização (condições KKT, anexo A.4). Isto quer dizer que a solução do problema dependerá somente dos vetores suporte.

Expandindo a função Lagrangeana (Equação (3.15)) obtém-se: [36]

$$L_P(\mathbf{W}, b, \alpha) = \frac{1}{2}\mathbf{W}^T\mathbf{W} - \sum_{i=1}^{N} \alpha_i d_i \mathbf{W}^T\mathbf{X}_i - b\sum_{i=1}^{N} \alpha_i d_i + \sum_{i=1}^{N} \alpha_i \tag{3.21}$$

De (3.18) obtém-se:

$$\mathbf{W}^T\mathbf{W} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{W}^T\mathbf{X}_i = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{X}_i^T\mathbf{X}_j \tag{3.22}$$

Devido à equação (3.19), o terceiro termo de (3.21) é igual a zero. Então, fazendo $L_P(\mathbf{W}, b, \alpha) = \mathbf{W}(\alpha)$, (3.21) pode ser reescrita como:

$$\mathbf{W}(\alpha) = \sum_{i=1}^{N} \alpha_i + \frac{1}{2}\mathbf{W}^T\mathbf{W} - \sum_{i=1}^{N} \alpha_i d_i \mathbf{W}^T\mathbf{X}_i \tag{3.23}$$

Substituindo (3.22) em (3.23):

$$\mathbf{W}(\alpha) = \sum_{i=1}^{N} \alpha_i + \frac{1}{2} \sum_{i=1}^{N} \alpha_i d_i \mathbf{W}^T \mathbf{X}_i - \sum_{i=1}^{N} \alpha_i d_i \mathbf{W}^T \mathbf{X}_i \tag{3.24}$$

$$\mathbf{W}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \alpha_i d_i \mathbf{W}^T \mathbf{X}_i \tag{3.25}$$

Substituindo (3.22) em (3.25), obtém-se:

$$\mathbf{W}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{X}_i^T \mathbf{X}_j \tag{3.26}$$

Pode-se agora formular o problema de otimização dual:

Dado um conjunto de treinamento constituído de duas classes linearmente separáveis $(\mathbf{X}_i, y_i) 1 \leq i \leq N, \mathbf{X} \in \Re^m, y \in \{+1, -1\}$, encontre os multiplicadores de Lagrange $(\alpha_i)\ 1 \leq i \leq N$ que resolvem o problema de otimização quadrático, ou seja, maximizar:

$$\mathbf{W}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j (\mathbf{X}_i^T \mathbf{X}_j) \tag{3.27}$$

sujeito às seguintes restrições:

$$\sum_{i=1}^{N} d_i \alpha_i = 0$$

$$\alpha_i \geq 0 \text{ para } i = 1, 2, ..., N$$

Somente dados de treinamento fazem parte da formulação deste problema de otimização dual. E somente de dados de entrada na forma de produtos escalares $(\mathbf{x}_i^T \mathbf{x}_j),\ 1 \leq i \leq N,\ 1 \leq j \leq N$, irá depender a função $\mathbf{W}(\alpha)$ a ser maximizada.

A solução deste problema de otimização é única, devido à Lagrangeana ser uma função convexa, e isto elimina a possibilidade de ocorrerem mínimos locais e faz com que o problema tenha como solução o mínimo global.

Após a determinação do vetor formado pelos multiplicadores de Lagrange ótimos, representados por $\alpha_{0,i}$, já é possível calcular o vetor de pesos ótimos $\mathbf{W}_0$, utilizando a equação (3.18),

$$\mathbf{W}_0 = \sum_{i=1}^{N} \alpha_{0,i} d_i \mathbf{X}_i \text{ onde } d \in \{+1, -1\}, \alpha_{0,i} \in R^+ \text{ e } \mathbf{X} \in \Re^m \tag{3.28}$$

Com a obtenção do vetor $\mathbf{W}_0$ pode-se determinar o hiperplano ótimo.

O *bias* ótimo pode ser calculado com a utilização de $\mathbf{W}_0$ e com uma amostra correspondente a um vetor suporte positivo X^{SV+} com $d = 1$, e assim escrever [36]:

$$b_0 = 1 - \mathbf{W}_0^T \mathbf{X}^{SV+} \text{ para } d^{SV+} = 1 \tag{3.29}$$

A função de decisão pode ser definida como

$$f(X) = \text{sign}(\mathbf{W}_0^T \mathbf{X} + b_0) = \text{sign}\left(\sum_{i=1}^{N} \alpha_{0,i} d_i (\mathbf{X}_i^T \mathbf{X}) + b_0\right) \tag{3.30}$$

onde o sinal de f(x) pode ser utilizado para classificar as amostras como sendo da classe -1 ou da classe $+1$ e $(\mathbf{X}_i^T \mathbf{X})$ é o *kernel* (que será visto em detalhes no próximo capítulo).

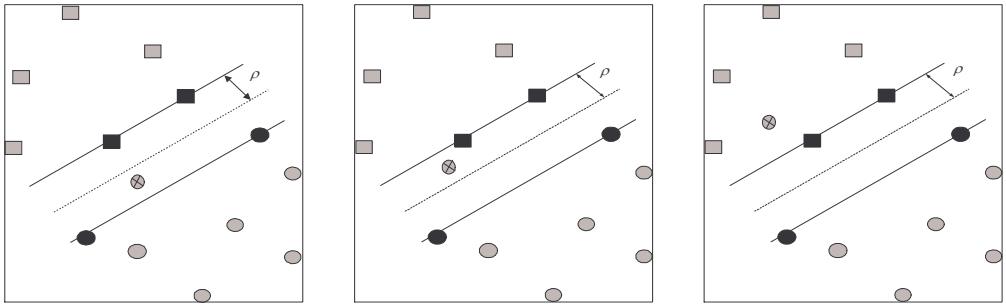
O classificador definido pela função de decisão (Equação (3.30)) é definido em termos de amostras de treinamento. Entretanto, somente as amostras com α diferente de zero (vetores suporte) definem o classificador. Na prática, a proporção de vetores suporte é muito pequena em relação ao número total de amostras. Logo, a solução é esparsa. Em outras palavras, as amostras é que definem o quão complexa a máquina necessita ser. Pode-se notar aqui um completo contraste com os classificadores que são tradicionalmente utilizados em reconhecimento de fala, como redes neurais (ANN) e modelos ocultos de Markov (HMM), onde a complexidade do sistema é normalmente predefinida ou escolhida por validação cruzada [6].

3.2 Hiperplano ótimo para padrões não linearmente separáveis

Esta formulação foi feita tendo como base [36] e [39].

Até aqui só foram analisados os padrões linearmente separáveis (que podem ser separados por um hiperplano). Porém, na maioria dos problemas reais é necessário separar dados não linearmente separáveis. Como extensão da formulação anterior, será discutido agora o caso de padrões não separáveis. O objetivo então passa a ser o de calcular um hiperplano ótimo que minimize a

possibilidade de ocorrências de erros de classificação. Neste caso, a margem de separação é dita como sendo flexível, pois existirão pontos (amostras de treinamento) que violarão as desigualdades da equação (3.13). Esta violação pode ocorrer de três formas diferentes, conforme Figura 3.10.



(a) (b) (c)

Figura 3.10: *Considerando que a linha pontilhada representa o hiperplano e que os vetores suporte estão indicados em preto: (a) O ponto marcado com um X está localizado dentro da margem de separação e do lado correto do hiperplano; (b) o ponto marcado com um X está localizado do lado incorreto do hiperplano e dentro da margem de separação; (c) o ponto marcado com um X está localizado do lado incorreto do hiperplano e fora da margem de separação.*

Nos casos apresentados na Figura, a classificação está correta no caso (a) e incorreta nos outros dois.

Para iniciar o estudo de classes não linearmente separáveis é necessário definir uma nova variável escalar e não negativa $\{\xi_i\}_{i=1}^{N}$, chamada de variável de folga, que será incluída na equação que define o hiperplano de separação, como mostrado na equação a seguir:

$$d_i[(\mathbf{W}^T\mathbf{X}_i) + b] \geq 1 - \xi_i,\ i = 1, 2, ..., N \tag{3.31}$$

As variáveis ξ_i medem o desvio de cada amostra $(x_i, d_i), 1 \leq i \leq N$ para o caso linearmente separável, ou seja, para a condição ideal de separação das classes (Figura 3.11). Para $\xi_i = 0$ a amostra está localizada em sua posição ideal. Para $0 < \xi_i \leq 1$, a amostra está localizada dentro da região de separação e do lado correto do hiperplano. Para $\xi_i > 1$ a amostra está do lado incorreto do hiperplano. Os vetores suporte são as amostras que satisfazem a igualdade presente na inequação (3.31), ou seja, são as amostras que mais perto estão do hiperplano [36].

É importante observar que se um vetor, que tem $\xi = 0$ e não é vetor suporte (i.e. Equação (3.31) maior que 1), for deixado fora do conjunto de treinamento, a superfície de decisão não muda (Vide Figura 3.11). Então, se um exemplo com

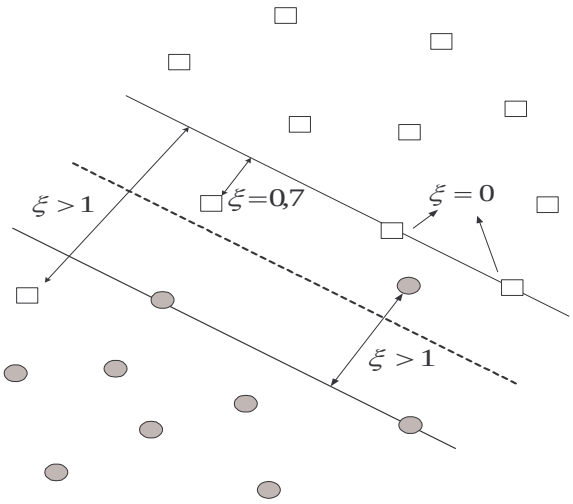


Figura 3.11: *Representação de diferentes valores para diferentes situações da variável de folga ξ.*

$\xi > 0$ for deixado de fora do conjunto de treinamento, a superfície de decisão tem grande chance de mudar.

Isto mostra que a definição de vetores de suporte é a mesma para o caso linearmente separável e para o não separável. E será visto mais a frente que a equação para os dois casos é muito semelhante.

Com a definição das variáveis de folga já realizada, o objetivo a ser alcançado é encontrar um hiperplano separador que minimize o erro de treinamento. Para isto será feita a minimização da seguinte função [36]:

$$\Phi(\xi) = \sum_{i=1}^{N} I(\xi_i - 1) \tag{3.32}$$

em relação ao vetor de pesos $\mathbf{W}$, a restrição da Equação (3.31) e a restrição sobre $||\mathbf{W}||^2$:

$$||\mathbf{W}||^2 \leq \frac{1}{\rho} \tag{3.33}$$

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

$$I(\xi - 1) = \begin{cases} 0 \text{ se } (\xi - 1) \leq 0 \\ 1 \text{ se } (\xi - 1) > 0 \end{cases} \tag{3.34}$$

Entretanto, a minimização de $\Phi(\xi)$ em relação a W é um problema de otimização não convexo que pertence a uma classe de problemas para os quais nenhum algo-

ritmo eficiente pode ser encontrado [10].

Para o problema ser tratável do ponto de vista matemático, $\Phi(\xi)$ será aproximada para:

$$\Phi(\xi) = \sum_{i=1}^{N} \xi_i \tag{3.35}$$

Ao simplificar o funcional que será minimizado com relação ao vetor peso $\mathbf{W}$, o cálculo computacional também será simplificado. Então tem-se:

$$\Phi(\mathbf{W}, \xi) = \frac{1}{2}\mathbf{W}^T\mathbf{W} + C\sum_{i=1}^{N} \xi_i \tag{3.36}$$

O primeiro termo do lado direito desta equação será minimizado quando $\mathbf{W}$ for minimizado. A minimização de $\mathbf{W}$ é conseguida pela minimização da dimensão VC, como visto na caso separável. O segundo termo refere-se à maior quantidade possível de erros de treinamento.

A variável C é o parâmetro de regularização (ou penalização) que faz o controle entre o número de erros de treinamento e a complexidade da máquina. Ele é escolhido pelo usuário e normalmente determinado experimentalmente, através do desempenho do algoritmo via dados de validação, ou de forma analítica estimando a dimensão VC, como foi mostrado no princípio de minimização do risco estrutural.

Para classes não linearmente separáveis, o problema de otimização primal que calcula o hiperplano ótimo, pode ser definido da seguinte maneira:

Dado $(\mathbf{X}_i, d_i)_{i=1}^{N}$, $\mathrm{x} \in \Re^m e\ \mathrm{d} \in \{+1, -1\}$, onde $\mathbf{X}$ representa as amostras de treinamento e d a classe à qual cada uma delas pertence, deseja-se calcular os valores de $\mathbf{W}$, b e ξ_i:

$$\begin{aligned} &\text{Minimizar: } V(\mathbf{W}, b, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \\ &\text{sujeito a:} \\ &d_i(\mathbf{W}^T\mathbf{x}_i + b) \geq 1 - \xi_i,\ i = 1, ..., N; \\ &\xi_i \geq 0,\ i = 1, ..., N. \end{aligned} \tag{3.37}$$

O caso de classes linearmente separáveis é um caso especial deste problema, onde todos os ξ são iguais a zero para todos os valores possíveis de i.

Agora, aplicando-se o método dos multiplicadores de Lagrange, pode-se transformar o problema de otimização primal em um problema dual.

Considerando-se a função Lagrangeana para o problema primal (Equação (3.37)):

$$L_P(\mathbf{W}, b, \xi, \alpha, r) = \frac{1}{2}\|\mathbf{W}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[d_i(\mathbf{W}^T\mathbf{X}_i + b) - 1 + \xi_i] - \sum_{i=1}^{N} r_i\xi_i \tag{3.38}$$

onde $\|\mathbf{W}\|^2 = \mathbf{W^T W}$ e os multiplicadores de lagrange α_i e r_i são não negativos.

Expandindo a função Lagrangeana (Equação (3.38)) :

$$L_P(\mathbf{W}, b, \xi, \alpha, r) = \frac{1}{2}\|\mathbf{W}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i d_i \mathbf{W}^T\mathbf{X}_i - b\sum_{i=1}^{N}\alpha_i d_i + \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N} r_i\xi_i \tag{3.39}$$

Da mesma forma que foi visto em classes separáveis, a Equação (3.38) é um problema de otimização restrito e sua solução pode ser determinada pelo ponto de sela, ou seja, minimizar a função em relação a $\mathbf{W}$, b, e C e maximizá-la em relação aos multiplicadores de Lagrange (aplicação das condições de Kuhn-Tucker). Será, então, necessário diferenciar a função $L_P(\mathbf{W}, b, \xi, \alpha, r)$ em relação a $\mathbf{W}$, b e ξ e igualar os resultados a zero [36].

Desta forma, tem-se:

$$\begin{aligned}
&\frac{\partial L}{\partial \mathbf{W}}(\mathbf{W}, b, \xi, \alpha, r) = \mathbf{w} - \sum_{i=1}^{N} d_i\alpha_i\mathbf{X}_i = 0;\\
&\frac{\partial L}{\partial b}(\mathbf{W}, b, \xi, \alpha, r) = \sum_{i=1}^{N} d_i\alpha_i = 0;\\
&\frac{\partial L}{\partial \xi}(\mathbf{W}, b, \xi, \alpha, r) = C - \alpha_i - r_i = 0;
\end{aligned} \tag{3.40}$$

Substituindo as equações obtidas na função Lagrangeana (Equação (3.39)):

$$L_P(\mathbf{W}, b, \xi, \alpha, r) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} d_i d_j \alpha_i\alpha_j(\mathbf{X}_i^T\mathbf{X}_j) \tag{3.41}$$

Esta equação é igual à que foi calculada para o caso das classes linearmente separáveis.

Agora já é possível escrever o problema de otimização dual. Neste caso o hiperplano é construído em um espaço característico de várias dimensões. Na prática, o hiperplano é um plano que corta o espaço característico de modo que as amostras pertencentes a uma classe sejam separadas das não pertencentes.

Este mapeamento no espaço característico é proveniente de um mapeamento não linear que é feito por um produto interno *Kernel* escolhido pelo usuário.

O problema de otimização dual pode ser descrito da seguinte maneira: tendo por base os dados de treinamento $(\mathbf{X}_i, y_i)$, $1 \leq i \leq N$, $\mathbf{X} \in \Re^m$ e $y \in \{+1, -1\}$ e fazendo uso do espaço característico definido de forma implícita pelo *Kernel* $K(X_i, X_j)$, calcule os multiplicadores de Lagrange (ótimos) $(\alpha_{i,0})$, $1 \leq i \leq N$ que irão resolver o problema de otimização quadrático [36, 39]:

$$\begin{aligned}
&\text{Maximizar:} \\
&\mathbf{W}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(\mathbf{X}_i, \mathbf{X}_j) \\
&\text{sujeito a :} \\
&\sum_{i=1}^{N} y_i \alpha_i = 0; \\
&\forall_{i=1}^{N} : 0 \leq \alpha_i \leq C
\end{aligned} \tag{3.42}$$

onde o parâmetro C deve ser especificado pelo usuário e ter valor positivo.

Examinando a formulação dual para classes separáveis e para classes não separáveis, verifica-se que a única diferença é que a restrição $\alpha_i \geq 0$, referente aos multiplicadores de Lagrange α_i's, é substituída por uma mais rigorosa, $0 \leq \alpha_i \leq C$. Pode-se notar também que as variáveis ξ_i não aparecem no problema dual. A função objetivo a ser maximizada é igual em ambos os casos.

Foi visto que a solução do problema dual é conseguida com a obtenção dos multiplicadores de Lagrange. Estes multiplicadores podem ser analisados da seguinte forma: quando um multiplicador de Lagrange for maior que zero e menor que 1, a amostra correspondente é um vetor suporte, se ele for igual a zero, tem-se uma amostra de treinamento que não participa da solução do problema e se for igual ao parâmetro de penalização C, esta amostra foi incorretamente classificada.

O vetor de pesos, o *bias* b e os vetores suporte são encontrados da mesma forma que foi vista para o caso de classes separáveis. Assim, a solução ótima para o vetor de pesos é dada por:

$$W_0 = \sum_{i=1}^{N_{sv}} \alpha_{0,i} d_i \mathbf{X}_i \tag{3.43}$$

onde N_{sv} é o número de vetores suporte.

Através das Equações (3.22) e (3.43) pode-se formular:

$$\|\mathbf{W}\|^2 = \mathbf{W^T W} = \sum_{i,j}^{N_{SV}} d_i d_j \alpha_i \alpha_j (\mathbf{X_i^T X_j}), \tag{3.44}$$

onde N_{SV} refere-se à quantidade de vetores suporte.

Da Equação (3.29) tem-se a seguinte expressão para o cálculo do *bias* b:

$$b_o = 1 - \mathbf{W}_o^T X^{SV+} \text{ para } d_i = +1 \tag{3.45}$$

onde X^{SV+} é um vetor suporte pertencente à classe positiva. Em [39] é dada outra forma para o cálculo do intercepto, via restrições primais.

Um detalhe importante na Equação (3.43) é que $\alpha_i = 0$ para toda amostra que não é vetor suporte. Os vetores suporte estão localizados sobre os hiperplanos canônicos (estes hiperplanos são paralelos ao hiperplano ótimo e estão a uma distância $\rho = 1/\|\mathbf{W_o}\|$ do mesmo). Estes pontos com $0 < \alpha_i < C$ são chamados de Vetores Suporte e além de dar nome à este tipo de máquina são os pontos que determinam todo o processo de treinamento e de classificação (ou predição).

Se todos os exemplos de treinamento, exceto os vetores suporte, forem removidos antes do treinamento, a superfície de decisão permanece a mesma.

Também pode-se notar que o problema de programação quadrática é convexo, isto é, não tem um mínimo local. Conseqüentemente, a solução sempre encontra o mínimo global. Isto é uma vantagem da SVM com relação a outras técnicas de otimização como, por exemplo, as redes neurais, onde o mínimo local existe.

A classificação final da amostra irá depender de qual lado do hiperplano separador (hiperplano ótimo) ela irá cair. A função de decisão para que isto seja realizado é dada por:

$$f(x) = \text{sign}(\mathbf{W}^T \mathbf{X} + b) = \text{sign}\left(\sum_{i=1}^{N_{sv}} y_i \alpha_{0,i} K(\mathbf{X}_i, \mathbf{X}) + b_0\right) \tag{3.46}$$

A variável N_{SV} no somatório indica que somente os vetores suporte serão relevantes para esta função de decisão. Isto ocorre porque que os α são não zero somente para um pequeno número de vetores (isto é válido para parâmetros ótimos dos *kernels*) chamados Vetores Suporte. Como somente os Vetores Suporte são considerados, a solução é dita ser esparsa.

3.3 SVM indutivo e SVM transdutivo

Neste trabalho a tarefa de classificação foi realizada utilizando os princípios da tradicional inferência indutiva. Entretanto, ela também pode ser realizada através da inferência transdutiva [33]. A técnica de aprendizado de máquina denominada

de Máquinas de Vetor de Suporte é utilizada como mecanismo de classificação.

A inferência indutiva é aquela normalmente utilizada em reconhecimento de fala onde as fases de treinamento e teste são realizadas em dois passos distintos.

Por outro lado, na inferência transdutiva, utiliza-se dois conjuntos de dados no treinamento do classificador: os dados tradicionais de treinamento e os 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. O objetivo será a classificação dos dados de predição. O classificador será treinado com estes dois conjuntos de dados e assim será possível classificar os dois conjuntos de dados em um único passo.

Mais detalhes sobre a inferência transdutiva podem ser encontrados em [33, 39]. Algumas aplicações de SVM modificada para atender os princípios da inferência podem ser encontradas em [39].

Capítulo 4

Kernels e metodologias para classificação com múltiplas classes

4.1 *Kernels*

Os *kernels* são necessários para permitir a separação de amostras (dados ou vetores de entrada) não linearmente separáveis por um hiperplano. Isto é possível a partir de um mapeamento em um espaço característico que possui número muito maior de dimensões, sem que este mapeamento precise ser explicitado ("mapeamento Φ"), como ilustrado na Figura 4.1.

Pode-se observar pela Figura 4.1 as duas formas de mapeamento, o implícito (função *kernel*) e o explícito (mapeamento Φ). O mapeamento Φ calcula as novas coordenadas dos dados de treinamento quando os mesmos são mapeados no espaço característico, onde eles podem ser mais facilmente separados pelo hiperplano ótimo. A função *kernel* permite esta separação, sem que o mapeamento precise ser explicitado.

Para entender a vantagem deste mapeamento, observe a Figura 4.2, considerando que ela represente duas classes de elementos: árvores e balões. Pode-se observar que deste ângulo (de cima e olhando através da posição dos balões) é difícil distinguir as classes.

Se os balões forem visualizados de um outro ângulo, como do nível do chão, por exemplo, será possível separar facilmente as classes com uma linha ou plano, ou seja, as classes se tornam linearmente separáveis, após um novo mapeamento (Figura 4.3). Isto foi possível, pois uma terceira dimensão foi considerada.

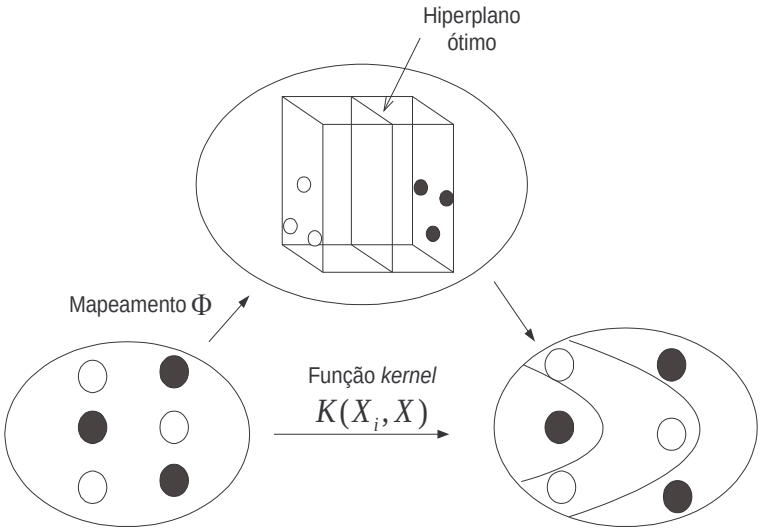


Figura 4.1: *As duas formas de mapeamento.*

A função do *kernel* é justamente esta: mapear os elementos que compõem as classes em um espaço de alta dimensão de modo que eles possam ser mais facilmente separáveis por um plano (hiperplano separador), minimizando assim o erro de classificação.

4.1.1 As funções *Kernel*

O produto interno $K(\mathbf{X_i}, \mathbf{X_j})$ pode ser visto como o elemento ij de uma matriz $\mathbf{M}$ simétrica N-por-N. Esta matriz é chamada de matriz *kernel* ou matriz Gram e é dada por:

$$M = \begin{bmatrix} k(X_1, X_1) & k(X_1, X_2) & \cdots & k(X_1, X_m) \\ k(X_2, X_1) & k(X_2, X_2) & \cdots & k(X_2, X_m) \\ \vdots & \vdots & \ddots & \vdots \\ k(X_m, X_1) & k(X_m, X_2) & \cdots & k(X_m, X_m) \end{bmatrix}$$

Seja o seguinte exemplo: Dado o vetor de entrada (ou amostra) $\mathbf{X_1} = (a_1, a_2) = (1, 2)$, o elemento $k(\mathbf{X_1}, \mathbf{X_1})$ é calculado pelo produto escalar $(a_1\ a_2) \cdot (a_1\ a_2)' = a_1^2 + a_1 a_2 + a_2^2 = (1)^2 + (1) * (2) + (2)^2 = 7$.

Logo, cada elemento $K(X_i, X_j)$ da matriz é um número escalar e são estes números que, quando substituídos na Equação (3.42), realizam o mapeamento dos dados para um espaço característico com muitas dimensões.

A matriz *kernel* é a estrutura central da SVM: ela contém todas as informações necessárias ao algoritmo de aprendizagem.

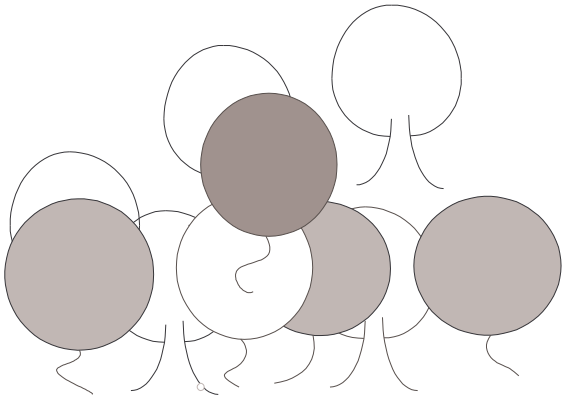


Figura 4.2: *Árvores e balões vistos por um ângulo que restringe a capacidade de classificação.*

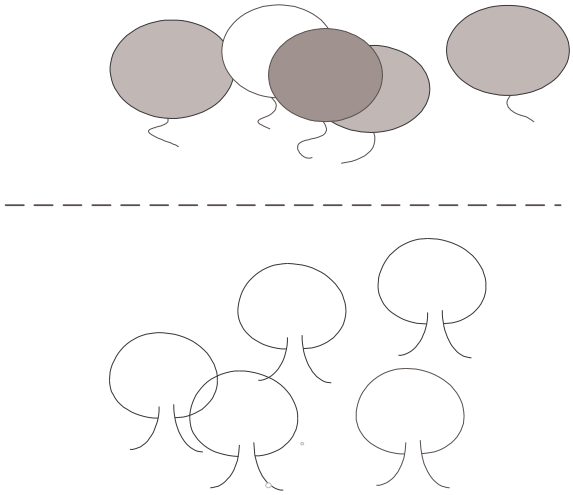


Figura 4.3: *As árvores e os balões são reposicionados e se tornam classes linearmente separáveis, pois agora a capacidade de classificação foi ampliada.*

A teoria que envolve *kernels*, produto interno e espaço característico é baseada no método RKHS (*Reproducing Kernel Hilbert Spaces*) [42, 43]. De acordo com esta teoria, um produto interno no espaço característico equivale a uma função *kernel* no espaço de entrada, ou seja:

$$K(\mathbf{X}, \mathbf{X^T}) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}^T) \rangle \tag{4.1}$$

onde $K(\mathbf{X}, \mathbf{X^T})$ é a função *kernel* e $\langle \phi(\mathbf{x})\phi(\mathbf{x}^T) \rangle$ é um produto interno no espaço característico (mapeamento explícito no espaço característico).

Para que uma função *kernel* exista, ela deve satisfazer às condições de Mercer [44]. Desta forma, se $K(\mathbf{X}, \mathbf{X^T})$ é uma função *kernel*,contínua, simétrica e definida

na região fechada $[a, b]*[a, b]$, ela deve satisfazer as seguintes condições de Mercer:

$$K(\mathbf{X}, \mathbf{X^T}) = \sum_{m}^{\infty} a_m \phi_m(x)\phi_m(x^T),\ a_m \geq 0, \tag{4.2}$$

$$\int_a^b \int_a^b K(\mathbf{X}, \mathbf{X^T})g(x)g(x^T)dxdx^T > 0,\ g \in L_2 \tag{4.3}$$

para todo $g(.)$ no qual

$$\int_a^b g^2(x)dx < \infty \tag{4.4}$$

O teorema de Mercer diz se uma função é de fato um produto interno *kernel* no espaço característico e portanto pode ser usada no treinamento de uma SVM.

Como foi visto, o *kernel* é útil em problemas reais, quando a SVM é utilizada para a classificação de classes não linearmente separáveis, isto é, quando não é possível separar satisfatoriamente os dados de treinamento por um hiperplano e uma função não linear é mais adequada. Neste caso, é necessário mapear cada amostra do conjunto de treinamento para o espaço característico.

Este espaço tem uma característica singular que é a escolha de uma função de mapeamento Φ apropriada. Este mapeamento pode fazer com que o conjunto de treinamento se torne um conjunto linearmente separável ou não. Caso ele se torne linearmente separável, passa a ser possível separar as amostras com um hiperplano [7, 45]). Pode ocorrer também que o conjunto não se torne linearmente separável após o mapeamento no espaço característico. O objetivo passa a ser a minimização do erro. Nos dois casos procura-se minimizar a possibilidade de erro de predição (erro na classificação de amostras futuras) de acordo com o princípio SRM. Se a taxa de erros de predição é baixa, diz-se que o sistema generaliza bem.

Percebe-se que com este procedimento, a única informação necessária sobre o mapeamento é a definição de como o produto interno $\Phi(X_i) * \Phi(X_j)$ pode ser calculado. Isto é obtido com os *kernels*, que após receber dois vetores (amostras) X_i e X_j do espaço de entrada, calculam o produto $\Phi(X_i) * \Phi(X_j)$ no espaço característico (proposto originalmente por Vapnik [7] este conceito está explicado também no tutorial de Burges [10] e no livro de Haykin [36]).

De maneira geral, é mais simples usar a função *kernel* do que fazer o mapeamento Φ (Figura 4.1). Por este motivo, é comum defini-la sem o conhecimento explícito deste mapeamento.

Por exemplo, sejam os vetores de entrada $\mathbf{X'} = (x_1, x_2)$ e $\mathbf{X}_i{}' = (x_{i1}, x_{i2})$ e utilizando um *kernel* polinomial, o mapeamento é dado por:

$$K(\mathbf{X}, \mathbf{X}_i) = (1 + X^T X_i)^2, \tag{4.5}$$

O produto interno pode ser expresso em termos de monômios de várias ordens da seguinte forma:

$$K(\mathbf{X}, \mathbf{X}_i) = 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2} \tag{4.6}$$

A imagem do vetor de entrada $\mathbf{X}$ induzida no espaço característico é dada por:

$$\Phi(\mathbf{X}) = [1, x_1^2, \sqrt{2}x_1 x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2] \tag{4.7}$$

De forma similar:

$$\Phi(\mathbf{X_i}) = [1, x_{i1}^2, \sqrt{2}x_{i1} x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}],\ i = 1, 2, 3, 4 \tag{4.8}$$

Por exemplo: Seja $X = (-1, -1)$ e $X_i = (-1, +1)$, então

$$K(\mathbf{X}, \mathbf{X}_i) = 1+(-1)^2(-1)^2+2(-1)(-1)(-1)1+(-1)^2 1^2+2(-1)(-1)+2(-1)1 = 1. \tag{4.9}$$

Como o mapeamento não precisa ser explicitado na prática, a "maldição da dimensionalidade" é minimizada. A "maldição da dimensionalidade" refere-se ao fato de que: quanto maior o número de dimensões do espaço em que estão as amostras, maior será o número de amostras necessário para caracterizar corretamente a distribuição das mesmas (também chamada função densidade de probabilidade). Mesmo com a minimização deste problema, o treinamento do classificador depende do número de amostras de treinamento para que um boa generalização seja possível [46].

Como visto anteriormente, a função do *kernel* é mapear os elementos que compõe as classes em um espaço de alta dimensão de modo que eles possam ser mais facilmente separáveis por um hiperplano.

Desta forma, o seguinte problema deve ser resolvido quando se utiliza *Kernels*: "Dado um conjunto de treinamento com vetores previamente classificados em duas classes distintas, desenvolver um procedimento que forneça um limite de decisão com boa generalização e que satisfaça as seguintes restrições:"

- O limite de decisão deve ser calculado via vetores de treinamento;
- O limite de decisão deve ser plano.

Para o desenvolvimento deste procedimento, é necessário calcular uma função de decisão que corte o espaço onde os dados foram mapeados de forma a minimizar os erros de classificação (Figura 4.4).

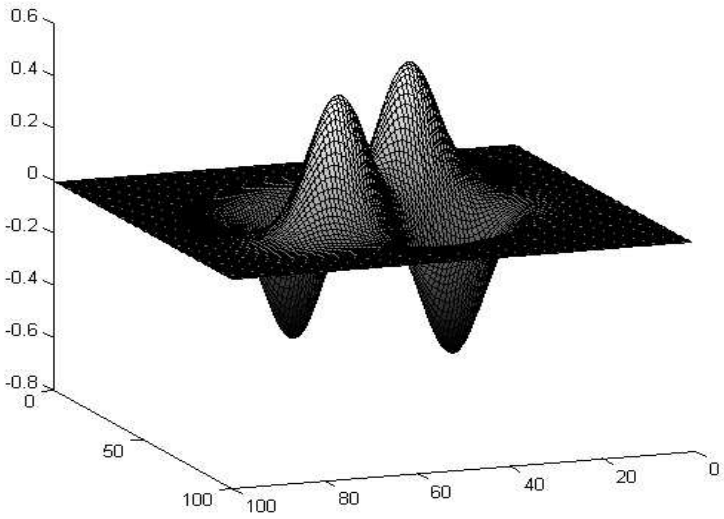


Figura 4.4: *Representação de um problema de classificação (problema do ou exclusivo) em um plano tridimensional (espaço característico), após a utilização do kernel RBF. A linha de decisão ou função discriminante (i.e.* $\{x \in X | g(x) = 0\}$ *é obtida pelo corte da hipersuperfície* $\sum_{i=1}^{m} y_i \alpha_i k(x, x_i)$ *na altitude 0 (esta altitude é dada pelo bias).*

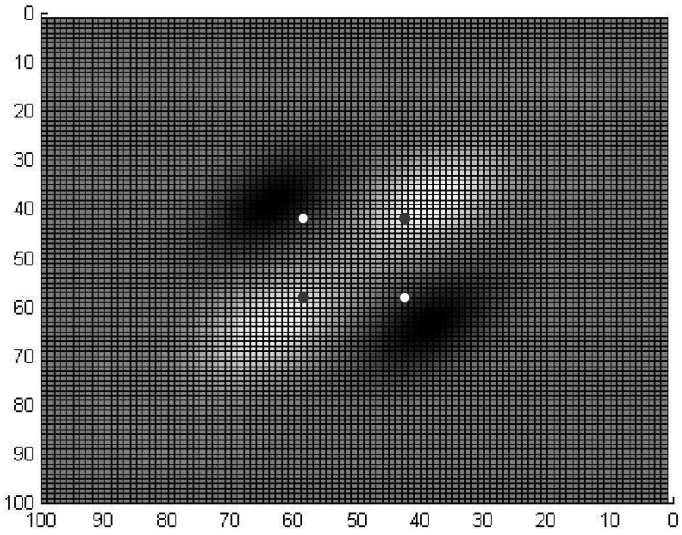


Figura 4.5: *Representação da discriminação de duas diferentes classes em um plano bidimensional (plano original dos dados) utilizado o kernel RBF no problema do "ou exclusivo".*

Para isto será escrita uma função de decisão em termos de *kernel* (mesma função de decisão que foi vista para o caso de classes não separáveis):

$$f(x) = \text{sign}(\mathbf{W}^T\mathbf{X} + b) = \text{sign}\left(\sum_{i=1}^{N_{sv}} d_i\alpha_{0,i}K(\mathbf{X}_i, \mathbf{X}) + b_0\right) \tag{4.10}$$

onde $\mathbf{X_i}$, $i = 1, 2, ..., m$, são os vetores suporte, $\mathbf{X}$ é a amostra a ser classificada, $K(\mathbf{X}, \mathbf{X_i})$ é o kernel, d é a classe do vetor, $\alpha_{0,i}$ são os multiplicadores de Lagrange ótimos e b é o bias.

4.1.2 Tipos de *kernel*

Os tipos de *kernels* mais usados são:

O kernel linear

$$K(\mathbf{X}_i, \mathbf{X}_j) = (X_i^T \cdot X_j) \tag{4.11}$$

onde n é o número de amostras de treinamento e i, j são iguais a $1, 2, ..., n$; $i, j \in \Re$; $i \neq j$.

O *kernel* polinomial

$$K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X_i}^T \cdot \mathbf{X}_j + 1)^d \tag{4.12}$$

onde d é o grau do polinômio.

O *kernel* "Função de base radial gaussiana"(RBF)

Este *kernel* é dado por:

$$K(\mathbf{X}_i, \mathbf{X}_j) = e^{\frac{-||\mathbf{X}_i - \mathbf{X}_j{}'||^2}{2\sigma^2}}. \tag{4.13}$$

onde σ^2 é a variância da gaussiana.

O método SVM, ao contrário dos métodos tradicionais de RBF, calcula todos os parâmetros necessários para construção da regra de decisão, dada pela equação (3.46):

$$f(x) = \text{sign}(\mathbf{W}^T\mathbf{X} + b) = \text{sign}\left(\sum_{i=1}^{N_{sv}} y_i\alpha_{0,i}K(\mathbf{X}_i, \mathbf{X}) + b_0\right) \tag{4.14}$$

onde:

- O número de centros N_{sv} é dado pelo número de vetores suporte;
- Os vetores que descrevem os centros são os vetores suporte $\mathbf{X_i}$;
- O valor do parâmetro σ na equação (4.13) é obtido pela seleção de parâmetros (via validação cruzada ou estimação da dimensão VC);
- As variáveis y_i são as classes de cada amostra de treinamento (só as amostras que são vetores suporte são utilizadas na função de decisão);
- As variáveis $\alpha_{0,i}$ são os multiplicadores de Lagrange.

Para exemplificar a utilização deste tipo de *kernels*, vide Figuras 4.4 e 4.5.

Polinômios completos

Este *kernel* é uma generalização para o *kernel* polinomial, sugerida por Vapnik [7] e é dado por:

$$K(\mathbf{X}, \mathbf{X_i}) = \left(\frac{(\mathbf{X^T} \cdot \mathbf{X_i})}{a} + b \right)^P, \tag{4.15}$$

onde a, b e P são definidos pelo usuário.

Função de base radial exponencial

$$K(\mathbf{X}, \mathbf{X_i}) = \exp\left(-\frac{\|\mathbf{X} - \mathbf{X}_i\|}{2\sigma^2} \right) \tag{4.16}$$

Este *kernel* produz uma solução linear por partes que pode ser útil em problemas em que descontinuidades são permitidas.

Splines lineares

As *splines* (interpolação polinomial segmentada) são provenientes do cálculo numérico. Elas vem recebendo uma atenção crescente em estatística devido à sua forte capacidade de adaptação na aproximação de funções.

As *splines* podem ser utilizadas na classificação de padrões. Para isto qualquer função pode ser traçada com *splines* em um determinado intervalo $[a, b]$ com o objetivo de separar amostras de classes diferentes. A idéia está em se dividir o intervalo original em intervalos menores $[x_o, x_1], ..., [x_k, x_k+1]$ e ajustar polinômios de grau p_i em cada intervalo $[x_i, x_i + 1]$, $i = 0, ..., k$. Este procedimento produz um polinômio por partes que pode assumir a forma de qualquer função.

Como o número de junções do *spline* não desempenha um papel importante, utiliza-se o *splines* com um número infinito de junções. Este *splines* é definido no intervalo $(0, a)$, $0 < c < \infty$, segundo a expansão [7]:

$$f(\mathbf{x}) = \sum_{i=0}^{p} a_i \mathbf{X}^i + \int_o^c s(t)(\mathbf{X} - t)_+^d dt, \tag{4.17}$$

onde os valores de a_i, $i = 0, ..., p$, serão determinados posteriormente. A expansão é determinada por uma função desconhecida $a(t)$ que pode ser vista como um produto interno. Agora a geração de *splines* de ordem p e com número infinito de junções já é possível.

No caso em que $p = 1$ tem-se:

$$K(\mathbf{X}, \mathbf{X_i}) = 1 + \mathbf{X}\mathbf{X_i} + \frac{1}{2}|\mathbf{X} - \mathbf{X_i}|\min(\mathbf{X}, \mathbf{X_i}) + \frac{(\min(\mathbf{X}, \mathbf{X_i}))^3}{3} \tag{4.18}$$

Este *kernel* é definido no intervalo [0,1). Ele também pode ser dado por:

$$K(\mathbf{X}, \mathbf{X_i}) = 1 + (\mathbf{X} \cdot \mathbf{X_i}) + \frac{1}{2}(\mathbf{X} \cdot \mathbf{X_i})\min(\mathbf{X} \cdot \mathbf{X_i}) - \frac{1}{6}\min(\mathbf{X} \cdot \mathbf{X_i})^3 \tag{4.19}$$

Bsplines

Uma outra formulação do *spline* é o *Bsplines*. Este *kernel* é definido no intervalo $[-1, 1]$, e tem a seguinte forma [7] (outras referências: [46], [47] e [48]):

$$K(\mathbf{x_i}, \mathbf{x_j}) = B_{2n+1}(\mathbf{x_i} - \mathbf{x_j}) \tag{4.20}$$

Para que as B_n-*splines* possam ser implementadas é necessário um equacionamento explícito que pode ser feito de duas formas: por um procedimento iterativo ou pela combinação linear de *splines* regulares [7].

Para que este equacionamento seja obtido, será considerada a seguinte função B_0-*splines*, ou seja *B-splines* de ordem 0:

$$B_0(u) = \begin{cases} 1, \text{ se } |u| \leq 0.5 \\ 0, \text{ se } |u| > 0.5 \end{cases} \tag{4.21}$$

O B_p-*spline* de ordem p pode ser definido como a convolução de duas funções: B_{p-1}-*splines* e B_0-*splines*:

$$B_p(u) = \int_{-\infty}^{\infty} B_{p-1}(u-t)B_0(t)dt. \tag{4.22}$$

Um $B_p(u)$-*splines* tem a seguinte construção:

$$B_p(u) = \sum_{r=0}^{p+1} \frac{(-1)^r}{r!} \left(u + \frac{p+1}{2} - r\right)_+^p \tag{4.23}$$

Vapnik [7] mostrou que ambas as definições descrevem a mesma coisa.

Usando B_p-*splines*, tem-se o seguinte mapeamento:

$$f(\mathbf{X}, \alpha) = \sum_{i=1}^{N} \alpha_i B_p(\mathbf{X} - t_i), \tag{4.24}$$

onde t_i, i= 1,..., N, definem as junções da expansão. Como a expansão tem a forma de um produto interno, a expansão B_p-*spline* é gerada pelo seguinte *kernel*:

$$K(\mathbf{X}, \mathbf{X_i}) = \sum_{K=1}^{N} B(\mathbf{X} - t_k)B(\mathbf{X_i} - t_k) \tag{4.25}$$

Perceptrons multicamadas

Este *kernel* é dado por uma função sigmoidal que pode ser escrita como:

$$K(\mathbf{X}, \mathbf{X_i}) = \frac{1}{1 + \exp\{v(\mathbf{X} \cdot \mathbf{X_i}) - C\}} \tag{4.26}$$

Este tipo de função (sigmoidal) é muito utilizado em redes neurais, logo, uma SVM com este tipo de *kernel* corresponderá à uma rede neural com uma camada escondida. Este *kernel* só satisfaz as condições KKT (Karush Kuhn Tucker) para alguns valores de C e v. (As condições KKT referem-se à condição necessária e suficiente para que o vetor de pesos $\mathbf{W}$ seja ótimo, conforme Equação (3.20) e anexo A.4).

Série de Fourier

A série de Fourier é muito utilizada no processamento de sinais para se descobrir as componentes de freqüência de um determinado sinal, ou seja, para passar um sinal do domínio do tempo para o domínio da freqüência.

A Série de Fourier pode ser considerada como uma expansão em um espaço característico de dimensão igual a $2N + 1$ [46].

Este *kernel* é definido na intervalo $[-\frac{1}{2}, \frac{1}{2}]$ e é dado por:

$$K(\mathbf{X}, \mathbf{X_i}) = \frac{\text{sen}(N + \frac{1}{2})(\mathbf{X} - \mathbf{X_i})}{\text{sen}(\frac{1}{2}(\mathbf{X} - \mathbf{X_i}))} \tag{4.27}$$

Este *kernel* não proporciona uma boa generalização [49, 46].

Ainda existem outros dois *kernels* que utilizam a Série de Fourier, o Fourier regularizado com modo de regularização fraca e o Fourier regularizado com o modo de regularização forte [50].

Produto tensorial

Para possibilitar a construção da SVM em um espaço n-dimensional, utiliza-se um produto de *kernels* unidimensionais, chamado de produto tensorial:

$$K(X, X_i) = \prod_{k=1}^{n} K(\mathbf{X_k}, \mathbf{X_{ik}}) \tag{4.28}$$

Este *kernel* é particularmente útil, quando utilizado em conjunto com *kernels splines* de várias dimensões.

kernels aditivos

Novos e mais sofisticados *kernels* podem ser obtidos com o somatório de *kernels* já existentes. Isto é possível, pois se as duas funções em separado satisfazem as condições KKT, a soma delas também irá satisfazer as mesmas condições. Esta soma pode ser expressa pela equação

$$K(\mathbf{X}, \mathbf{X_i}) = \sum_{k} K_k(\mathbf{X}, \mathbf{X_i}) \tag{4.29}$$

onde K_k, $k = 1, 2, ...$ representa cada uma das funções *kernels* que estão sendo somadas.

4.2 Bias implícito e Bias explícito

Um *kernel* pode utilizar ou não um bias explícito, como por exemplo, no caso do *kernel* polinomial:

$$\begin{cases} K(\mathbf{X}, \mathbf{X_i}) = (\mathbf{X} \cdot \mathbf{X_i})^P \\ K(\mathbf{X}, \mathbf{X_i}) = (\mathbf{X} \cdot \mathbf{X_i} + 1)^P \end{cases} \tag{4.30}$$

onde a equação em que aparece o termo $+1$ é a que tem *bias* explícito. A utilização

deste bias, além de tornar o resultado mais estável, também evita o problema de Nulidade da Hessiana (matriz que é utilizada nas técnicas de otimização que envolvem a função *kernel*). Esta matriz é a derivada segunda de uma função multivariável. Por exemplo, seja $f(x, y)$, então a matriz Hessiana será dada por:

$$H(x,y) = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial^2 y} \end{bmatrix}$$

onde $\frac{\partial^2 f}{\partial^2 x}$ é a derivada segunda da função f em relação a x e $\frac{\partial^2 f}{\partial^2 y}$ é a derivada segunda da função f em relação a y.

Apesar da utilização do *bias* explícito levar a um método de implementação mais eficiente, ambas as soluções generalizam bem, sendo que as soluções são diferentes em cada caso.

A título de exemplo, pode-se observar que, fazendo a fatoração dos polinômios, o mapeamento de uma amostra no espaço característico feito pelo *kernel* polinomial com *bias* explícito terá mais dimensões que com o *bias* implícito.

Continuando com o exemplo, seja uma amostra $\mathbf{X} = (x_1, x_2)$, o mapeamento pelo *kernel* polinomial com *bias* explícito será $(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$ e o mapeamento com o *bias* implícito será $(x_1^2, \sqrt{2}x_1x_2, x_2^2)$.

Pode-se concluir então que a máquina treinada com o *kernel* polinomial com *bias* explícito terá capacidade e complexidade maiores.

4.3 Escolhendo o melhor *kernel* e o parâmetro de penalização C

Como foi visto, cada *kernel* possui um parâmetro específico. No caso do *kernel* polinomial, por exemplo, este parâmetro é o grau do polinômio d e para o caso do *kernel* RBF ele será a variância da gaussiana σ^2.

Outro parâmetro importante é o parâmetro de penalização C. Ele é responsável pela definição da importância relativa entre a maximização da margem de separação (minimização do vetor de pesos) e a minimização do erro de treinamento.

O parâmetro específico do *kernel* é fortemente relacionado com o parâmetro de penalização C. Devido a isto, não é possível pesquisar o valor de cada parâmetro separadamente. Logo, é interessante procurar uma combinação ótima destes dois parâmetros.

Dentre as diferentes formas de se fazer isto serão citadas duas:

4.3.1 Validação cruzada

Inicialmente alguns pontos são plotados tendo como coordenadas o parâmetro C e o parâmetro do *kernel*. O desempenho em cada um destes pontos é verificado e encontrada uma região promissora, são feitos diversos testes por validação cruzada nas proximidades deste ponto. A forma como a validação cruzada é feita está detalhada no anexo A.5.

4.3.2 Limite superior da dimensão VC

A técnica mais formal para o cálculo do *kernel* ótimo é pelo limite superior com relação à dimensão VC [10]. Entretanto, apesar da dimensão VC conter as informações de complexidade e generalização do *kernel*, não há prova de que o *kernel* é o ótimo. A escolha do *kernel* só será validada por inúmeros testes independentes.

Existem técnicas para o cálculo da dimensão VC média (em uma faixa de valores indicada pelo usuário) para um determinado conjunto de treinamento. Este calculo é feito através da seguinte equação [51]:

$$h \leq R^2 \|\mathbf{W}\|^2 \tag{4.31}$$

onde h é a dimensão VC, $\mathbf{W}$ o vetor de pesos ótimo e R é o raio da menor hiperesfera que circunda todos os dados de treinamento. Este raio pode ser calculado via otimização quadrática. Com este método pode-se procurar a máquina de menor dimensão VC, ou seja, pode-se escolher o melhor *kernel* e seus parâmetros.

Pode-se usar também a seguinte formulação:

$$h \leq min(R^2 \|\mathbf{W}\|^2, n) + 1, \tag{4.32}$$

onde n é a dimensão dos vetores de entrada (amostras ou dados).

A Equação (4.32) calcula o limite superior para a dimensão VC.

Uma forma alternativa para o cálculo da dimensão VC, é deixar de fora da hiperesfera uma ou mais das amostras. Desta forma é possível conseguir um valor de dimensão VC ainda menor. Com isto a capacidade (ou complexidade) da máquina necessária para classificar determinado conjunto de dados também diminui. Esta diminuição dependerá da distribuição dos dados no interior da hiperesfera. A Figura 4.6 ilustra esta idéia.

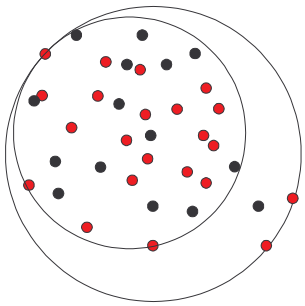


Figura 4.6: *Em um plano bidimensional, a hiperesfera é representada por um círculo. Pode-se observar que o menor círculo necessário para envolver todo o conjunto de treinamento torna-se bem menor quando três das amostras mais externas são retiradas.*

4.3.3 Diferentes valores de C para cada classe

Com relação à escolha do parâmetro de penalização C, nas aplicações em que o número de amostras para cada classe é diferente, pode ser interessante que C assuma um valor para cada classe de amostras. Isto foi sugerido no contexto de uma aplicação médica para dados desbalanceados (cada classe de dados tem um tamanho diferente) [52] .

4.4 Decomposição do conjunto de treinamento

Em certas aplicações, como por exemplo, no reconhecimento de fala, o tamanho do conjunto de treinamento necessário é da ordem de milhares ou até milhões de amostras. Neste casos, a quantidade de memória requerida se torna computacionalmente intratável, pois muitos algoritmos de otimização quadrática precisam armazenar a matriz *kernel* integralmente. Este problema foi resolvido através do algoritmo de decomposição proposto por Osuna e Girosi [53]. Ele decompõem o problema de otimização em vários problemas menores. Assim este problema será dividido em uma parte inativa e uma ativa, chamada de "conjunto de trabalho". Os α's referentes à parte ativa (conjunto de trabalho) serão atualizadas a cada interação do algoritmo. Desta forma, os α's referentes a cada parte do conjunto de treinamento vão sendo atualizados por partes, até que a solução ótima seja encontrada.

Alguns algoritmos que utilizam a estratégia proposta acima podem ser encontrados em [54, 55].

4.5 Metodologias para classificação com múltiplas classes

Como a técnica de SVM é originalmente para classificação binária, serão necessários métodos para estendê-la para múltiplas classes. Existem duas formas de se fazer isto: a primeira forma é a combinação de vários classificadores binários ("um contra um", "um contra todos","um contra todos único", DAG, árvore binária, ECOC, MOC e Podh, proposto neste trabalho) e a segunda forma é a que considera todas as classes na formulação do problema de otimização (métodos que consideram todos os dados). De forma geral a primeira forma é a mais utilizada, pois é computacionalmente menos custoso solucionar vários problemas de classificação binária que solucionar um problema com várias classes.

Segundo Hsu & Lin [56], os métodos mais convenientes para o uso prático são o "um contra um" e o DAG.

4.5.1 "Um contra um"

Este método foi introduzido por Knerr [57] sendo que Fridman [58] sugeriu a estratégia "Max Wins", Krebel [59] o aplicou com excelentes resultados e ele foi comparado com outros métodos por Hsu e Lin [47].

Sendo n o número de classes, este método irá treinar um classificador binário para cada uma das possíveis combinações de duas classes, totalizando $n(n-1)/2$ classificadores. Por exemplo, se temos as classes 1, 2 e 3, serão treinados classificadores relativos aos pares 1 e 2, 1 e 3 e também 2 e 3. Cada classificador binário resolverá o seguinte problema de classificação binária (ou problema de otimização para o caso de dados não separáveis):

Dado $(\mathbf{X}_i, d_i)_{i=1}^{N}$, $\mathbf{X} \in \Re^m e\, d \in \{+1, -1\}$, onde $\mathbf{X}$ representa as amostras de treinamento e d a classe a qual cada uma delas pertence, deseja-se calcular os valores de $\mathbf{W}$, b e ξ_i:

$$\begin{aligned} &\text{Minimizar: } V(\mathbf{W}, b, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &(\mathbf{W}^T\mathbf{X}_i + b) \geq 1 - \xi_i, \text{ se } d_i = +1,\ i = 1, ..., N; \\ &(\mathbf{W}^T\mathbf{X}_i + b) \leq 1 - \xi_i, \text{ se } d_i = -1,\ i = 1, ..., N; \\ &\xi_i \geq 0,\ i = 1, ..., N. \end{aligned} \tag{4.33}$$

Este problema de otimização primal é o mesmo que foi resolvido para classes

não linearmente separáveis. Após o cálculo de seu correspondente dual, ele resulta na seguinte função de decisão (que é a mesma que foi vista para dados não separáveis):

$$f(x) = \text{sign}(\mathbf{W}^T\mathbf{X} + b) = \text{sign}\left(\sum_{i=1}^{N_{sv}} y_i \alpha_{0,i} K(\mathbf{X}_i, \mathbf{X}) + b_0\right) \tag{4.34}$$

A classificação da amostra irá depender do sinal de $f(\mathbf{X})$, ou seja, de que lado do hiperplano ótimo ela está. A classe que vencer em um classificador binário receberá um voto.

Para a fase de testes, existem diversas formas de combinar os classificadores binários para que uma única classe seja atribuída à uma amostra de teste $\mathbf{X}$. A mais comum é aquela em que, na fase de testes, cada amostra de entrada é classificada por cada um dos classificadores e a classe que receber mais votos ganha. Ou seja, para cada classificador binário, a equação $\text{sign}(\mathbf{W}^T\mathbf{X}_i + b)$ indicará a qual classe pertence a amostra ou dado $\mathbf{X}$, então um voto é dado à esta classe. Esta forma de votação é chamada de Estratégia *Max Wins* [58].

No caso de duas classes receberem o mesmo número de votos, uma estratégia de desempate deve ser adotada. Uma forma de desempate foi sugerida por Hsu & Lin [47], no qual a classe que tem o menor índice é a vencedora. Contudo, esta estratégia não é adequada, pois não tem justificativa teórica e depende da implementação das classes.

As vantagens do método "um contra um" é que ele é de fácil entendimento e implementação além de ter um bom desempenho, Como desvantagem pode-se citar que o número de classificadores binários se torna rapidamente muito grande com o aumento do número de classes, isto faz com que seja necessário uma grande quantidade de memória para armazenamento das variáveis. Além disso, na fase de teste utilizando a estratégia *Max Wins*, como cada amostra tem que ser classificada por todos os classificadores, o custo computacional será elevado.

Com o objetivo de diminuir o custo computacional e/ou de melhorar o desempenho do classificador, surgiram formas alternativas de se combinar os classificadores após o treinamento com o método "um contra um". Como exemplo, pode-se citar as estratégias DAG, árvore binária e Podh que é uma contribuição original desta tese.

4.5.2 "Um contra todos"

Este foi provavelmente o primeiro método a ser utilizado na classificação de múltiplas classes [60].

Se n é o número de classes, este método treinará n classificadores binários.

Cada classe é então comparada com o conjunto formado por todas as outras. Na fase de testes, a amostra de teste é então classificada por cada um dos classificadores. O classificador binário pode apresentar saída binária (presença do operador sign na equação 4.34) ou analógica (equação sem o sign). Este operador faz com que todos as saídas positivas assumam o valor $+1$ e que todas as negativas assumam o valor -1.

Assim com a presença ou não do operador sign um dos classificadores deverá apresentar valor positivo na saída, indicando a classe vencedora.

Como isto na prática nem sempre acontece, uma alteração válida é fazer com que a amostra de teste seja classificada como pertencente à classe que obtiver o valor mais alto (ou mais baixo, conforme implementação), a partir da função de decisão de cada classificador binário [60], após a retirada do operador sign. A principal vantagem desta alteração é o grande aumento do desempenho (maior que todos os métodos testados), como será mostrado no capítulo com os resultados das simulações.

Neste método existem n classificadores e, portanto, n funções de decisão:

$$\begin{array}{c} (\mathbf{W}_1^T\mathbf{X} + b_1) \\ \vdots \\ (\mathbf{W}_n^T\mathbf{X} + b_n). \end{array} \tag{4.35}$$

Logo, a classe da amostra será dada por:

$$\text{classe de X} \equiv \underset{\text{i=1,..., n}}{\text{argmax}}(\mathbf{W}_i^T\mathbf{X} + b_i), \tag{4.36}$$

Uma vantagem do método "um contra todos" é que o número de classificadores é pequeno. A classificação (predição) de uma amostra de teste com este método será mais rápida do que com o método "um contra um". Como desvantagem pode-se citar que a quantidade de memória necessária durante o processamento é maior que a de outros métodos (na fase de treinamento e na fase de testes). Isto se deve ao fato de que cada classificador precisa ser treinado com todos os dados de treinamento.

Assim como no método "um contra um ", surgiram formas alternativas de se combinar os classificadores após o treinamento com o método "um contra todos". Como exemplo pode-se citar as estratégias ECOC e "um contra todos único".

4.5.3 ECOC

O método ECOC (em inglês *Error-correcting output codes*), proposto por Dietterich e Bakiri em 1991 [61], cada classe é relacionada com uma palavra código binária pertencente a uma matriz código. Um exemplo de matriz código é a matriz formada pelas possíveis combinações de saída do método "um contra todos", onde tem-se 1 para as respostas positivas e zero para as negativas. Por exemplo, para 6 classes tem-se a seguinte matriz código:

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

onde cada linha corresponde à uma determinada classe.

Na estratégia ECOC, a combinação da saída dos classificadores dará origem à uma palavra binária (palavra do código). Se n é o número de bits de um código, a quantidade de classes ou palavras código possíveis é igual 2^n.

A palavra binária obtida após a classificação de uma amostra de teste é comparada com cada linha da matriz código e a que tiver a menor distância de Hamming vence.

O número de *bits* que muda de uma palavra binária para outra é chamado de distância de Hamming [62]. Se d é a menor distância de Hamming em uma matriz código, pode-se corrigir até $(d-1)/2$ erros com um código corretor de erros.

Na matriz de código dada, a distância de Hamming é 1 e, desta forma, não será possível corrigir nenhum erro.

Como outros exemplos de matriz código pode-se citar a matriz densa e a matriz formada pelo código BCH (Bose Chaudhuri Hocquenhem), utilizadas por Allwein [63] e Ghani [64], respectivamente.

No caso de cinqüenta classes, por exemplo, seria necessário utilizar um código com comprimento mínimo de 6 *bits*. Um código que pode ser utilizado é o código BCH (15, 50), que precisará de apenas 15 SVMs para classificar 50 classes.

As 3 três primeiras linhas deste código, em um um total de 50 linhas, são:

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Um método praticamente idêntico ao ECOC é o MOC (em inglês, *Minimal Output Coding*). O MOC, da mesma forma como o ECOC, associa uma palavra binária com cada classe de amostras. A diferença é que neste método um mínimo número de bits n_b é usado para codificar n_c classes:

$$n_b = \log_2 n_c \tag{4.37}$$

A vantagem deste método é a utilização de um número menor de SVMs e a desvantagem é que o código corretor de erros não pode ser utilizado. Desta forma a memória utilizada pelo MOC é menor que a utilizada pelo ECOC mas o desempenho do MOC tende a ser menor.

4.5.4 "Um contra todos único"

O treinamento deste método é semelhante a do "um contra todos" em conjunto com a do "um contra um". Como visto na fase de testes do método "um contra todos", nem sempre um só classificador produz valor positivo e então tem-se mais de uma classe vencedora (i.e. um empate). Neste caso uma nova etapa é incluída. Nesta etapa, é aplicado o método "um contra um". Ou seja, se n é o número de classes, foram treinados mais $n(n-1)/2$ classificadores e se q é o número de classificações positivas, então os $q(q-1)/2$ classificadores referentes às classes que empataram serão utilizados para classificar a amostra $\mathbf{X}$ e a classe que conseguir mais votos ganha. Observa-se que neste caso o problema do empate foi resolvido.

Apesar do possível ganho de desempenho, as desvantagens deste método são muitas pois são somadas as do método "um contra um" e a do método "um contra todos".

A desvantagem com relação ao método "um contra todos" é que a quantidade de memória necessária durante o processamento é maior que a de outros métodos.

E com relação ao método "um contra um" (caso de empate), além dos K classificadores normalmente treinados no método "um contra todos", são treinados mais $n(n-k)/2$ classificadores. Ou seja, o número de classificadores binários se torna rapidamente muito grande com o aumento do número de classes.

Assim, o custo computacional será maior que a dos outros dois métodos.

4.5.5 DAG ou DAGSVM

Platt e co-autores [65] sugeriram um método chamado de grafo direcionado acíclico (do inglês *Directed Acyclic Graph Support Vector Machine*).

Neste método, o treinamento é o mesmo do "um contra um" (se n é o número de classes, são treinados $n(n-1)/2$ classificadores). Entretanto, na fase de testes, o DAG utiliza um grafo direcionado acíclico com $n(n-1)/2$ nós e n níveis.

Pela Figura 4.7 pode-se observar que todas as classes estão presentes no nó inicial, por exemplo 1, 2, 3 e 4, e a amostra de teste será classificada pelo SVM (classificador binário), correspondente ao par 1 e 4. Supondo que a SVM classifique a amostra como não pertencente à classe 1, o algoritmo irá para o nó (2, 3 e 4). Continuando do nó (2, 3 e 4), com a locução sendo classificada pelo SVM, correspondente ao par 2 e 4, se a amostra for classificada como não pertencente à classe 4, então o algoritmo irá para o nó (2, 3). E, finalizando o exemplo, a amostra será classificada como não pertencente à classe 3 e a classe 2 é vencedora.

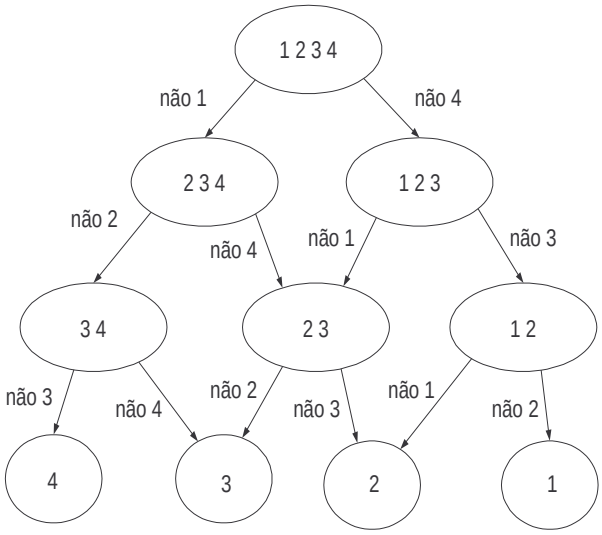


Figura 4.7: *Ilustração do método DAGSVM.*

Uma das vantagens deste método é que seu tempo de processamento na fase de teste é menor que nos anteriores.

Hsu and Lin [47] apresentaram resultados de teste mostrando que o DAG e o "um contra um" são mais práticos que outros métodos. Além disso, Platt [65] apresentou um resultado, sugerindo que o desempenho do DAG é igual ou um pouco maior que o dos métodos "um contra todos" e "um contra um".

4.5.6 Árvore binária

Este método, proposto por Salomon em [66], foi inspirado em uma estrutura em árvore que é muito comum em máquinas de aprendizado. Sua estrutura é muito semelhante à da DAG. O que diferem os dois métodos é que a árvore binária não aproveita o treinamento do "um contra um". Isto faz com que este método precise ser treinado de forma diferente. Sua estrutura é mostrada na Figura 4.8. Pela Figura pode-se observar que no nó inicial estão presentes todas as classes. A diferença do método anterior é que cada classificador binário é treinado com um grupos de classes e não com apenas um par. Por exemplo, no nó inicial temos uma classe representando o grupo de classes 1, 2, 3 e 4 e outra classe representando o grupo 5, 6 e 7. Assim, cada classificador verifica a qual grupo de classes pertence a amostra, até restar apenas uma classe possível.

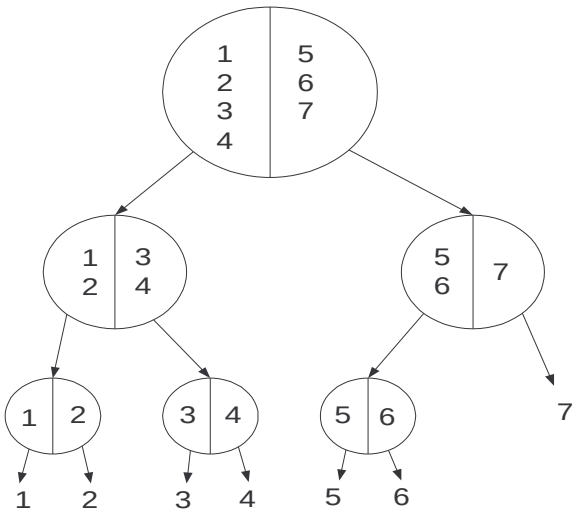


Figura 4.8: *Ilustração do método árvore binária.*

Apesar de necessitar no treinamento de uma pequena quantidade de SVMs e de ter um pequeno tempo de processamento na fase de testes, o desempenho deste método em [66] se mostrou abaixo do apresentado pelos outros métodos.

4.5.7 Métodos que consideram todos os dados

Cada um destes métodos segue o mesmo princípio que o "um contra todos", ou seja, separa cada classe do conjunto formado por todas as outras. A diferença é que as funções discriminantes são calculadas por um único problema de otimização. Métodos baseados nesta estratégia foram propostos por Vapnik [33] e por Weston & Watkins [12].

Seja j como sendo a classe que será separada das demais e k o número de classes, o problema de otimização pode ser escrito da seguinte forma [60]:

$$\begin{aligned}
&\text{minimizar } V(\mathbf{W}, B, \xi) = \frac{1}{2}\sum_{j=1}^{k}(\mathbf{W_j})^T\mathbf{W_j} + C\sum_{i=1}^{N}\sum_{j\neq y_i}^{K}\xi_i^j \\
&\text{sujeito às seguintes restrições:} \\
&\mathbf{W^{y_i}}^T\mathbf{X_i} + b^{y_i} \geq \mathbf{W^j}^T\mathbf{X_i} + b^j + 2 - \xi_i^j \\
&\xi_i^j \geq 0 \\
&i = 1, ..., N \\
&j \in \{1, ..., k\} \setminus y_i.
\end{aligned} \tag{4.38}$$

onde j representa a j-ésima classe e y_i todas as outras. Logo, a função de decisão é

$$\underset{i=1,...,k}{\text{argmax}}(\mathbf{W_j}^T\mathbf{X} + b_j) \tag{4.39}$$

Isto significa que a classe vencedora será indicada pela função de decisão que obtiver o maior resultado.

Um método similar foi proposto por Hsu & Lin [47] com a diferença de que ao invés de utilizar a formulação anterior (Equação (4.38)), o termo $\sum_{j=1}^{k}(b^j)^2$ foi adicionado à função objetivo. Então tem-se [60]:

$$\begin{aligned}
&\text{minimizar } V(\mathbf{W}, B, \xi) = \frac{1}{2}\sum_{j=1}^{k}[(\mathbf{W_j})^T\ b_j][\mathbf{W_j}\ b_j]^T + C\sum_{i=1}^{N}\sum_{j\neq y_i}^{k}\xi_{ij} \\
&\text{sujeito às restrições} \\
&[(\mathbf{W}_y i)^T\ b_{y_i}][\mathbf{X_i}\ 1]^T \geq [(W_j)^T\ b_j][\mathbf{X_i}\ 1] + 2 - \xi_{ij} \\
&\xi_{ij} \geq 0,\ i = 1, ..., N,\ j \in \{1, ..., K\} \setminus y_i.
\end{aligned} \tag{4.40}$$

A função de decisão é a mesma do método "um contra todos":

$$\underset{\text{i=1,..., n}}{\text{argmax}}((\mathbf{W}_i^T\mathbf{X} + b_i), \tag{4.41}$$

A formulação que utiliza um só problema de otimização teve ainda outras formulações equivalentes. Elas foram propostas por Guermeur [67] e por Brednsteiner

& Bennett [68]. Estas formulações apresentam a desvantagem de apresentar limitações computacionais [60].

Outra formulação deste mesmo princípio foi proposta por Crammer e Singer [69] e tem como diferença a utilização de apenas N variáveis de folga ξ.

4.5.8 Método da poda por histograma - PODH

Este método é uma contribuição desta tese. Ele teve origem a partir de uma análise do método "um contra um" (onde a amostra pertencerá à classe que receber mais votos). Nesta análise, verificou-se que, no método "um contra um", se n é o número de classes, então $n-1$ é o máximo número de votos que receberá a classe vencedora.

Por exemplo, se o problema tem 10 classes, o máximo número de votos que receberá a classe vencedora é 9.

Para um problema de classificação de 50 palavras, verificou-se que uma amostra era classificada como pertencente a uma determinada classe com 49 votos em 97,59 % dos casos e com 48 votos em 2,16 % dos casos (tabela 4.1).

Tabela 4.1: *Número de votos recebido pela classe vencedora.*

Resultados com	49 votos	48 votos	47 votos	46 votos
Percentual	97.5882 %	2.1569 %	0.2157 %	0.0392 %

A idéia então é fazer com que uma classe seja eliminada após perder a disputa em um ou dois classificadores.

A fase de treinamento deste método é idêntica à do método "um contra um" onde são treinados $n(n-1)/2$ classificadores binários.

A fase de testes é composta de uma ou mais sessões de votação e uma sessão de poda.

Na sessão de votação cada locução será classificada com classificadores binários da seguinte forma (Figura 4.9): considerando as classes 1, 2, 3, 4, 5 e 6 tem-se os pares: 1 e 2, 3 e 4, 5 e 6. Cada par correspondendo a um classificador. Após se classificar a amostra de teste em cada um desses classificadores, é obtida uma lista com todas as classes e a quantidade de votos recebida por cada classe. Em seguida serão eliminadas as classes que obtiverem votação igual a zero (logo na sessão de votação elimina-se metade das possíveis classes). Em cada iteração, a amostra de teste será passada através do classificador referente aos pares formados pelas classes sobreviventes e as classes que não obtiverem nenhum voto são eliminadas. Isto ocorre sucessivamente até só restar uma classe que é a classe vencedora.

Para o caso de duas sessões de votação e considerando as classes 1, 2, 3, 4, 5 e 6 tem-se os pares 1 e 6, 2 e 4 , 3 e 5, relativos à primeira sessão de votação e

os pares 1 e 2, 3 e 4, 5 e 6 relativos a segunda sessão de votação(Figura 4.10). Pode-se observar pela figura que, após duas sessões de votação, as classes que receberam o menor número de votos são eliminadas. A sessão de poda é igual ao caso anterior, conforme ilustrado na figura. Ou seja, em cada iteração, a amostra de teste será passada através do classificador referente aos pares formados pelas classes sobreviventes e as classes que não obtiverem nenhum voto são eliminadas.

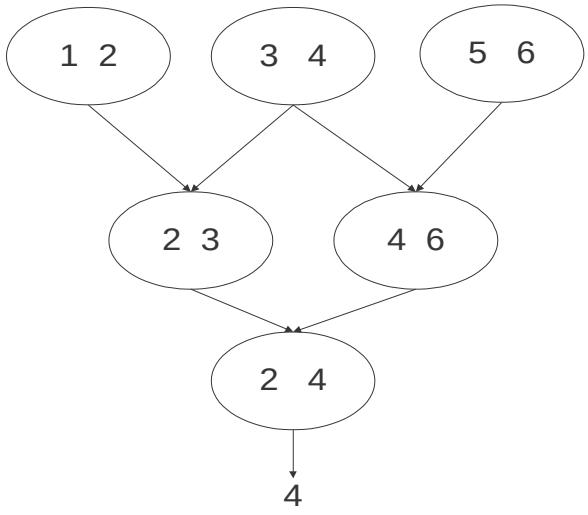


Figura 4.9: *O método da poda por histograma com uma sessão de votação plena, seguida de um sessão de poda. Na sessão de poda, após duas classes competirem entre si, a que perdeu é eliminada permanentemente da disputa. No caso de número ímpar de classes, uma das classes terá que competir duas vezes.*

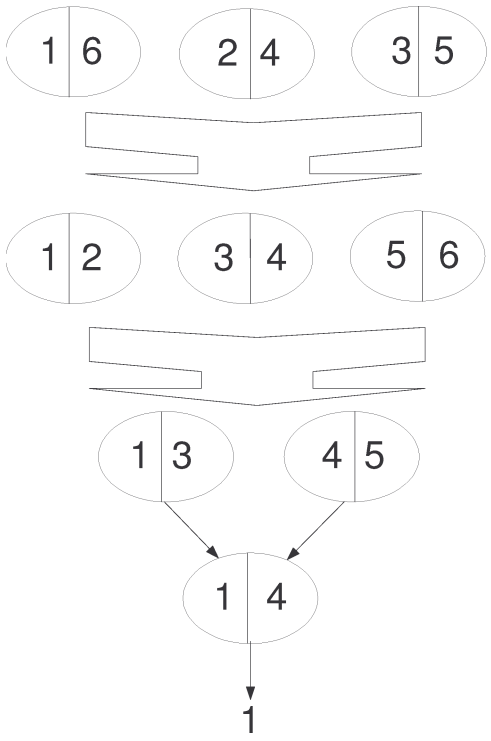
O método Podh com uma sessão de votação tem como vantagens um tempo de processamento bem menor que os outros métodos e um desempenho igual ao "um contra um".

No entanto, quando utilizado com duas sessões de votação, este método passa a ter um desempenho um pouco maior que o conseguido pelos métodos "um contra um" e DAG, mas seu tempo de processamento aumenta muito e se iguala ao DAG.

Os resultados dos testes serão apresentados no próximo capítulo.

4.5.9 Análise da complexidade computacional dos métodos

Uma das preocupações com relação à eficiência de um determinado algoritmo é quando se utiliza um número considerável de amostras, fato que ocorre com freqüência em aplicações na área de reconhecimento de fala.



classe	votos	ativa ?
1	1	sim
2	0	sim
3	1	sim
4	1	sim
5	0	sim
6	0	sim

classe	votos	ativa ?
1	2	sim
2	0	não
3	2	sim
4	1	sim
5	1	sim
6	0	não

Figura 4.10: *O método da poda por histograma com duas sessões de votação plena.*

Uma forma de exprimir uma aproximação da relação entre a quantidade de trabalho necessário e o número de elementos considerados é utilizar a notação matemática conhecida por notação 'O' (lê-se 'Oh' ou, algumas vezes, 'Big Oh') [70].

Será calculada, agora, a notação ´O' para as funções que calculam o número de SVMs necessários para algumas das metodologias existentes.

A tabela 4.2 contém as funções que calculam o número de SVMs e a tabela 4.3 contém as respectivas notações 'O'.

Os "métodos que consideram todos os dados" precisam de somente um classificador SVM com complexidade computacional equivalente a n^n .

Tabela 4.2: *Número de SVMs necessários para cada método. O método ECOC dependerá também do código utilizado. A variável n refere-se ao número de classes.*

Método	Treinamento	Teste
"Um contra um"	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
DAG	$\frac{n(n-1)}{2}$	n
Podh(1 sessão de votação)	$\frac{n(n-1)}{2}$	n
Podh(2 sessões de votação)	$\frac{n(n-1)}{2}$	$n + \frac{n}{2}$
"Um contra todos"	n	n
Árvore binária	$n-2$ à $n-1$	De $\frac{n-1}{2}$ à $\frac{n-3}{2}$
ECOC	$\geq \log_2 n$	$\geq \log_2 n$
MOC	$\log_2 n$	$\log_2 n$

Tabela 4.3: *Comparação da complexidade de cada método segundo a notação 'O' referente a tabela 4.2. O método ECOC utiliza um número de SVMs igual ao número de bits do código utilizado. Assim, se b é o número de bits, o número de classes possíveis com um código de comprimento b é 2^b. A variável n refere-se ao número de classes.*

Método	Treinamento	Teste
"Um contra um"	n^2	n^2
DAG	n^2	n
Podh(1 sessão de votação)	n^2	n
Podh(2 sessões de votação)	n^2	n
"Um contra todos"	n	n
Árvore binária	n	n
ECOC	$\geq \log_2 n$	$\geq \log_2 n$
MOC	$\log_2 n$	$\log_2 n$

Capítulo 5

Resultados experimentais

5.1 Introdução

Os experimentos foram realizados com o objetivo de apontar e estudar algumas possíveis alternativas para o reconhecimento de palavras isoladas com a técnica de SVM.

Inicialmente foi feita a normalização dos dados e o desempenho foi comparado ao obtido sem normalização. Foram utilizados os coeficientes mel, Δ mel e Δ Δ mel pois são os mais utilizados na literatura [71, 72]. O *kernel* RBF também foi o escolhido inicialmente devido ao mesmo motivo e de ser apontado por Hsu [73] como uma boa escolha inicial. O objetivo da normalização é diminuir a ocorrência de problemas numéricos.

Em seguida, algumas metodologias para a classificação de múltiplas classes foram comparadas. Foi proposta, então, uma nova metodologia denominada Podh (poda por histograma), a qual foi escolhida para a continuidade dos testes, pois, dentre as que utilizam o treinamento do "um contra um", foi a que obteve o menor tempo de processamento sem perda de desempenho. O objetivo foi levantar dados para poder escolher qual a metodologia mais apropriada quando se quer otimizar o tempo de processamento, o desempenho do sistema ou a combinação de ambos.

Outro experimento realizado foi a comparação do método "um contra um" e do método "um contra todos", com o objetivo de comparar o custo computacional e o desempenho dos mesmos.

Continuando com os experimentos, foi feita a redução da dimensão das amostras (vetores) com a utilização do método PCA (análise de componente principal) e tendo por objetivo diminuir o custo computacional do sistema.

Nos testes realizados foi feita a seleção dos parâmetros ótimos do *kernel*, através de um conjunto de dados de validação.

Por fim, foram feitas conclusões baseadas nos experimentos realizados e na

comparação entre o SVM e algumas das técnicas mais utilizadas na literatura sobre reconhecimento de fala.

5.2 Base de dados

Foi utilizada uma base de dados [74] composta por 50 palavras (vide anexo B), sendo que algumas delas com sons muito parecidos, o que dificulta a classificação. Na gravação foi utilizada uma freqüência de amostragem de 8 KHz e 16 *bits*. Nesta base, os conjuntos de treinamento, teste e validação são compostos por 5250, 4500 e 600 locuções, respectivamente. O conjunto de treinamento equivale a 50,72 % das locuções, o de teste a 43,48 % e o de validação a 5,80 %. Com relação à quantidade de locutores masculinos e femininos, tem-se o conjunto de treinamento, teste e validação dado pela tabela 5.1.

Tabela 5.1: *Divisão dos conjuntos de teste e treinamento para locutores masculinos e femininos.*

fase	masculino	feminino	totais
treinamento	22	13	35
teste	18	12	30
validação	03	01	04

5.3 Parâmetros acústicos

O método de extração de características padrão é a utilização de extratores melcepstrais [75]. Isto é conseguido, convertendo a representação da forma de onda de uma palavra (ou locução) em diversos vetores de 12 parâmetros melcepstrais (mel), sua derivada de tempo(Δ mel) e sua segunda derivada($\Delta\ \Delta$ mel).

Cada locução é dividida em diversos quadros. Normalmente, como o tamanho das locuções varia, a quantidade de quadros também é variável. O comprimento da menor locução é 0,160 s e o da maior locução 1,780 s. Cada vetor de 12 parâmetros mel representa uma janela de 20 ms da locução, estas janelas abrangem toda a extensão da locução e são sobrepostas de modo que nenhuma informação seja perdida na extração de parâmetros devido à ação dos filtros [75]. Estes vetores são organizados de modo a formar um único vetor para cada locução, o qual também tem tamanho variável.

A primeira questão que se apresenta é que a entrada do SVM tem tamanho fixo. Logo, para que este processo de extração de parâmetros possa ser adequado

ao SVM, ou seja, para que o conjunto de vetores de parâmetros que representa cada locução (i. e. um vetor) possa se adequar à entrada, ele precisa ser de tamanho fixo.

Para que isto fosse possível, o tamanho da janela máxima, da janela mínima, do deslocamento máximo e do deslocamento mínimo foram variados, conseguindo assim manter o número de quadros fixo. O menor número de quadros conseguido foi de 49. A razão de utilizar o menor número de quadros possível se justifica pela diminuição do custo computacional e da dimensão VC.

A Figura 5.1 ilustra o procedimento. Em um primeiro momento, cada locução é dividida em 49 quadros. Na figura as janelas estão sendo representadas pelos retângulos abaixo dos quadros. Para cada janela serão extraídos 12 coeficientes melcepstrais. No sistema implementado, o comprimento de cada janela variou de 15 a 45 ms e a sobreposição mínima foi de 25 % da janela. Os vetores mel, de 12 coeficientes cada, serão então dispostos em seqüência, compondo assim um único vetor de comprimento igual a 588.

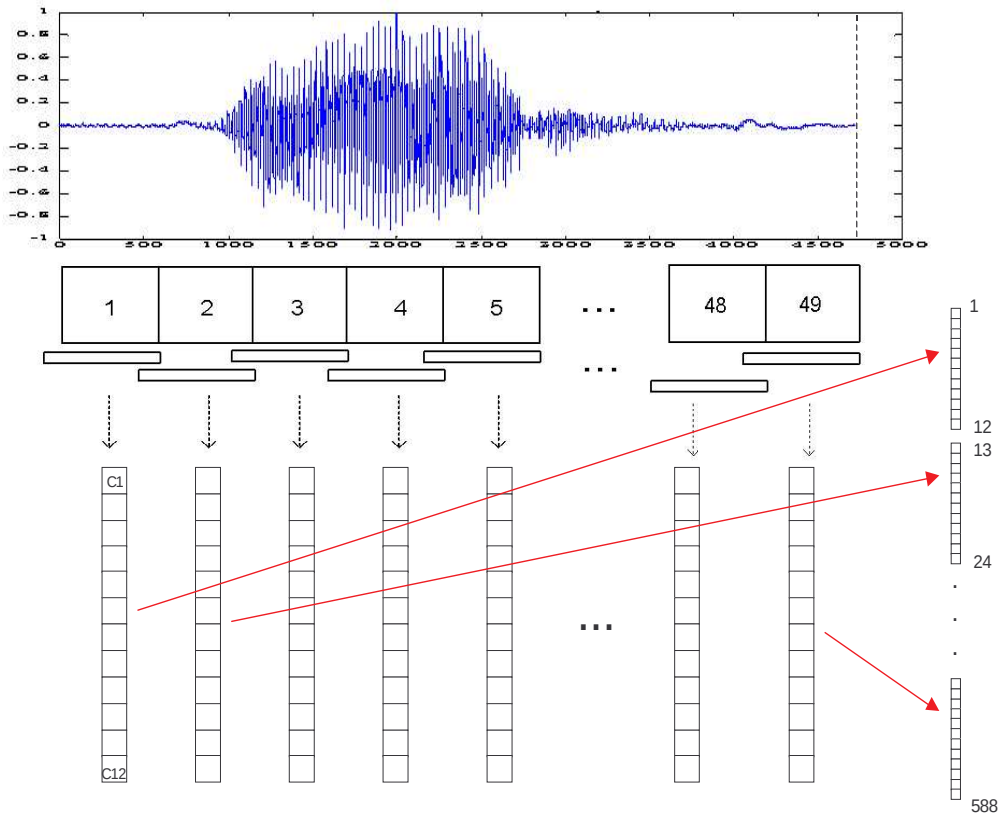


Figura 5.1: *Conversão de um sinal de fala em um vetor de entrada composto de coeficientes melcepstrais e com número fixo de quadros.*

5.4 Normalização dos dados

A necessidade da normalização pode ser explicada pela análise da seguintes equações:

$$K(\mathbf{X}_i, \mathbf{X}_j) = e^{\frac{-||\mathbf{x_i} - \mathbf{x_j}^T||^2}{2\sigma^2}} \tag{5.1}$$

e

$$K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j^T + 1)^d. \tag{5.2}$$

A equação (5.1) representa o kernel RBF. Observe que se o valor da diferença entre $\mathbf{X}_i$ e $\mathbf{X}_j^T$ na equação for aumentando, o valor da exponencial tenderá rapidamente para $-\infty$, dificultando a distinção entre as amostras.

Na equação (5.2) está representado o *kernel* polinomial. O resultado desta equação tenderá a ∞ com o aumento de $\mathbf{X_i}$ e $\mathbf{X}_j^T$ e com a aumento do grau do polinômio. A normalização dos dados diminui os valores de $\mathbf{X_i}$ e $\mathbf{X_j}^T$, minimizando este problema numérico.

Apesar disto, a normalização pode não ser indicada para todos os tipos de *kernel* ou de aplicação, sendo necessário fazer testes prévios para cada caso.

A normalização faz com que os valores assumidos pelas amostras fiquem dentro de uma determinada faixa. Como por exemplo entre -1 e 1. Com isto, os problemas citados anteriormente são minimizados.

Os histogramas dos dados (coeficientes mel) antes e depois da normalização estão nas figuras (5.2) e (5.3). Os valores máximos e mínimos antes da normalização são 87,96 e -76,29 respectivamente e depois da normalização 13,16 e -13,11. Entretanto, estes valores não são visíveis na figura devido aos valores de escala do eixo vertical serem muito elevados.

A normalização dos dados será explicada a seguir.

Seja a seguinte equação:

$$\mathbf{X_{norm}} = \frac{\mathbf{X} - \mathbf{E(X)}}{\sigma} \tag{5.3}$$

onde $\mathbf{X}$ é a amostra a ser normalizada, o termo $\mathbf{E(X)}$ refere-se a média de $\mathbf{X}$, σ é o desvio padrão dado por $\sqrt{Var(\mathbf{X})}$.

Considere agora que o conjunto de treinamento é composto de 5250 amostras (vetores) com 1764 dimensões cada uma e que cada amostra é composta por três partes com 588 dimensões cada uma. A primeira parte corresponde aos coeficientes mel, a segunda aos coeficientes delta mel e a terceira aos coeficientes delta delta mel. A normalização de cada uma destas partes será feita em separado. Isto será repetido para todos os vetores.

O algoritmo irá decompor a primeira parte do vetor em 49 vetores de 12

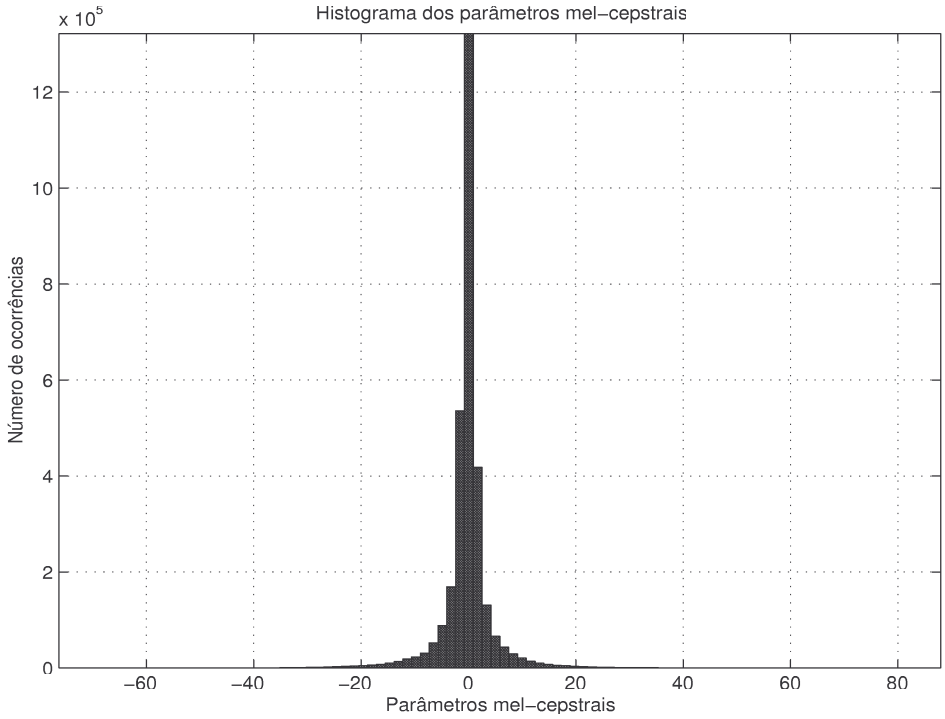


Figura 5.2: *Histograma dos dados sem normalização.*

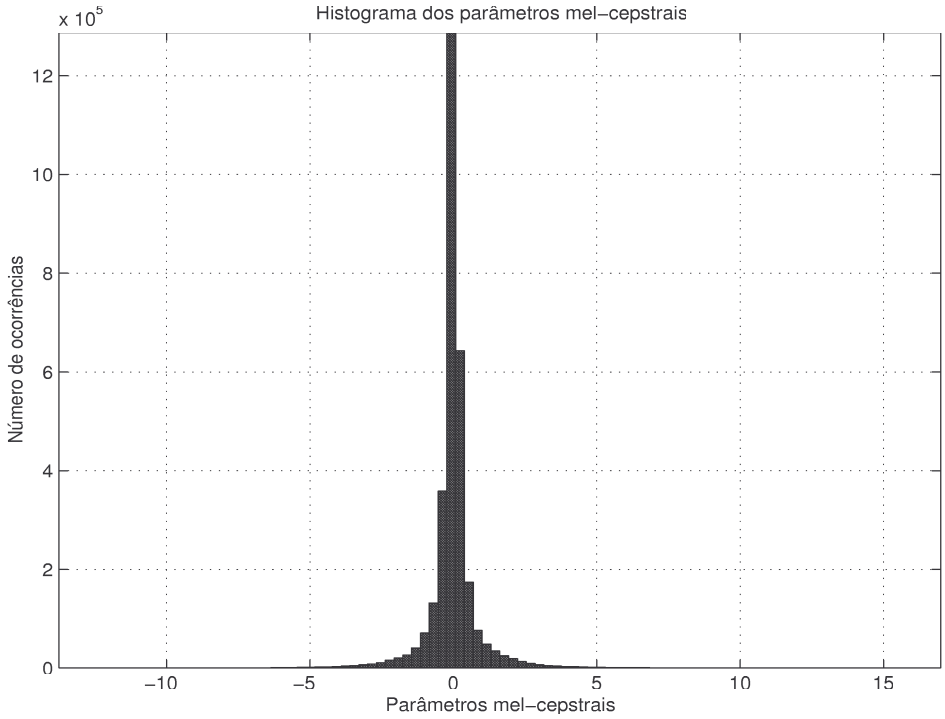


Figura 5.3: *Histograma dos dados normalizados.*

dimensões cada uma. Estes vetores serão colocados, lado a lado, formando uma matriz de 12 por 49. A média de cada uma das 12 linhas será calculada, assim será obtido um vetor de 12 dimensões. Em seguida, este vetor média será subtraído de cada um dos 49 vetores (cada uma das colunas da matriz). Um novo vetor de 588 dimensões irá então ser construído colocando cada coluna da matriz em seqüência, este vetor será representado por **xm**.

Depois disto, o algoritmo calcula a variância. Para isto, primeiro se calcula a média geral de todos os elementos de cada vetor $E(\mathbf{X})$ (coeficientes mel). E em seguida a média geral de todos este elementos ao quadrado $E(\mathbf{X^2})$. A variância e o desvio padrão serão então dados por:

$$Var(\mathbf{X}) = E(\mathbf{X}^2) - E(\mathbf{X})^2 \tag{5.4}$$

e

$$\sigma = \sqrt{Var(\mathbf{X})} \tag{5.5}$$

Finalmente, o vetor normalizado é calculado por **:**

$$\mathbf{X_{norm}} = \frac{\mathbf{xm}}{\sigma} = \frac{\mathbf{X} - E(\mathbf{X})}{\sigma} \tag{5.6}$$

Este procedimento é então repetido para cada parte do vetor, onde uma variância diferente para cada tipo de coeficiente (mel, delta mel e delta delta mel) será calculada, totalizando 3 valores de variância.

Os resultados obtidos com e sem normalização estão na tabela 5.2.

Para a obtenção dos parâmetros ótimos foi feita uma busca exaustiva com dados de validação.

Tabela 5.2: *Resultados comparando os desempenhos com e sem normalização. Foi utilizado o* kernel *RBF.*

	TX validação	TX teste%
Sem normalização, $C = 105$ e $\sigma = 46,2$	93,2	90,4
Com normalização, $C = 90$ e $\sigma = 0,5$	93,0	89,6

O resultado esperado com a normalização é a diminuição dos problemas numéricos e uma possível melhora no desempenho. Entretanto, o resultado dos experimentos realizados (tabela 5.2) indica que a normalização prejudicou o desempenho do sistema para o caso do *kernel* RBF. Um outro fato relevante é que a ocorrência de problemas numéricos aumentou significativamente com relação aos dados sem normalização.

Com isto pode-se concluir que a utilização de normalização não é indicada no caso específico em que se utiliza o *kernel* RBF para o reconhecimento de palavras isoladas.

Para fundamentar as conclusões aqui expostas, foi consultada a literatura existente e assim verificou-se que a normalização é mais utilizada para alguns tipos específicos de *kernel*. Como exemplo, pode-se citar o polinomial e o perceptron multicamadas.

5.5 Variando a metodologia utilizada para a classificação de múltiplas classes

Foram comparadas diferentes metodologias para a classificação de múltiplas classes: "um contra um", " DAG" e Podh (ou poda por histograma) que é uma contribuição deste trabalho.

A tabela 5.3 contém uma comparação de resultados obtidos com os métodos citados. O número de SVMs refere-se ao número de classificadores binários, necessários na fase de teste, pois o número de SVMs na fase de treinamento é igual para todos estes métodos (1225 SVMs). Os termos 1 ses. e 2 ses. referem-se à utilização de uma ou de duas sessões de votação utilizando o método Podh.

Tabela 5.3: *Comparando os resultados conseguidos com cada método de classificação. Neste caso foi utilizado o* kernel *RBF gaussiana com* $\sigma = 6.4$ *e C=1. O termo "Tempo por palavra" refere-se ao tempo necessário para o reconhecimento de uma palavra. Foi utilizado um pentium 4 e o software Matlab.*

	um contra um	DAG	podh (1 ses.)	podh (2 ses.)
TX acertos em %	90,529	90,294	90,529	90,686
Tempo total	120 hs	10 hs	5,2 hs	9,9 hs
Tempo por palavra	84,7 s	7 s	3,67 s	6,9 s
Número de SVMs	1225	50	50	75

Os resultados obtidos com o método Podh estão na tabela 5.4. Pode-se observar que o método Podh com 1 sessão de votação apresentou o mesmo desempenho do método "um contra um" e o menor tempo de processamento na fase de testes. Pode ser observado também que, com a inclusão de mais uma sessão de votação no método Podh (2 sessões de votação), o desempenho do sistema aumenta ligeiramente, mas o tempo de processamento aumenta muito. Quando 3 sessões de votação são utilizadas, o desempenho praticamente não se altera, entretanto o tempo de processamento aumenta muito.

Pode-se concluir então que o melhor Podh a ser utilizado quando o fator tempo é predominante, é aquele com uma sessão de votação. Devido a isto ele será utilizado nos próximos experimentos.

Tabela 5.4: *Resultados dos testes com a variação do número de sessões de votação no método Podh.*

Resultados com	1 sessão	2 sessões	3 sessões
Taxa de acertos em %	90,53	90,69	90,61
Tempo total	5,2 hs	9,9 hs	14,2 hs
Tempo por palavra	3,68 s	6,99	10,02

5.5.1 Experimentos utilizando o método "um contra todos"

O método "um contra todos" utiliza a função de decisão vista para o caso de classes não separáveis, reescrita abaixo por conveniência:

$$f(x) = \text{sign}(\mathbf{W}^T\mathbf{X} + b) = \text{sign}\left(\sum_{i=1}^{N_{sv}} y_i \alpha_{0,i} K(\mathbf{X}_i, \mathbf{X}) + b_0\right) \tag{5.7}$$

Como visto anteriormente, o método "um contra todos" utiliza todos os dados para o treinamento de cada SVM.

Os experimentos com este método obtiveram o melhor desempenho de todos os métodos testados em termos de taxa de acertos. Entretanto, isto só foi conseguido após a retirada do operador sign da função de decisão [47] (equação (5.7)) .

Os melhores desempenhos obtidos com e sem o operador sign estão na tabela 5.5.

Tabela 5.5: *Resultados dos testes com o método "um contra todos" com sign,* kernel *RBF e dados não normalizados. Os parâmetros ótimos forma escolhidos via validação cruzada.*

	Taxa de acertos em %
Com sign, C =300 e $\sigma = 30$	83,94
Sem sign $C = 300$ e $\sigma = 40$	92,18

O aumento do desempenho com a retirada do operador sign é explicado pelo fato de que houve uma mudança na regra que escolhe a classe vencedora. Isto acontece pois, ao invés da classe vencedora ser aquela com sinal diferente dos demais, ela passa a ser a que tem o maior valor (ou menor valor, conforme implementação).

A tabela 5.6 compara o método Podh com o método "um contra todos". Por esta tabela pode-se observar que, apesar do método "um contra todos" apresentar um desempenho maior, seu tempo de processamento é muito grande. Isto limita a busca dos parâmetros ótimos com a utilização da validação cruzada à faixas de valores menores.

Tabela 5.6: *Comparação entre o método "um contra um" e o método "um contra todos".*

	"um contra um"	"um contra todos" %
Maior desempenho obtido	90,90 % (com PCA)	92,18 %
Menor tempo de treinamento	1,12 hs	24 hs
Número de SVMs	1225	50
Número de amostras por SVM	210	5250
Tempo por palavra (fase de teste)	3,67 s	4,8 s

5.6 Seleção de características dos dados

A seleção de características tem como objetivos diminuir o custo computacional do sistema, através da redução da dimensão das amostras e analisar relações entre as características.

Entre os métodos que podem ser utilizados tem-se a análise de componente principal (PCA- *Principal Component Analysis*). A definição teórica de PCA está no anexo A.1.

5.6.1 Experimentos com PCA

Para os testes foi utilizada a técnica de SVM com a metodologia "podh" para múltiplas classes.

Para calcular os autovalores e autovetores foi utilizado o algoritmo QL com deslocamentos implícitos em conjunto com o procedimento de tridiagonalização Householder [76].

A Tabela 5.7 foi gerada pelo algoritmo, o qual também foi utilizado para diminuir a dimensão dos dados. Nesta tabela pode-se observar a quantidade de informação (explicabilidade) referente à uma determinada quantidade de componentes principais. Pode-se observar, por exemplo, que a quantidade de informação restante após a redução do número de coeficientes de 36 para 25 é de 99,72 %.

As taxas de desempenho obtidas estão na tabela 5.8.

Com relação aos resultados obtidos na tabela 5.8, houve uma pequena melhora no desempenho (0,2 à 0,4 %). A explicação para isto se deve ao fato de que a técnica de PCA faz com que os dados sejam representados em eixos ortogonais, ou seja que fiquem descorrelacionados, facilitando sua classificação.

Ao mesmo tempo, com a redução da dimensionalidade dos dados, ocorre a redução da dimensão VC. Assim, a capacidade de generalização do sistema irá aumentar.(O limite superior da dimensão VC é dado por $h \leq \min(\|\mathbf{W}\|^2 R^2, n)+1$, onde W é o vetor de pesos, R o raio da menor hiperesfera que contém os dados, após o mapeamento no espaço característico e n é o numero de dimensões

Tabela 5.7: *Numero de componentes e a percentagem da informação correspondente.*

Número de componentes	Quantidade da informação em %
36	100
35	99,99
34	99,98
33	99,96
32	99,95
31	99,93
30	99,91
29	99,88
28	99,85
27	99,81
26	99,77
25	99,72
24	99,66
23	99,59
22	99,52
21	99,44
20	99,35
19	99,25
18	99,14
17	99,01
16	98,85
15	98,65
14	98,40
13	98,05
12	97,51
11	96,95
10	96,25
09	95,40
08	94,37
07	93,01
06	91,54
05	89,67
04	87,32
03	83,29
02	74,76
01	58,83

do espaço característico).

Como feito anteriormente, a literatura existente foi consultada com o objetivo de fundamentar e/ou validar as conclusões aqui expostas.

Nesta literatura são apresentadas aplicações na área de reconhecimento de

Tabela 5.8: *Desempenho obtido com a diminuição da dimensão dos dados de entrada. Os parâmetros ótimos foram calculados via dados de validação.*

	TX validação	TX teste	tempo treinamento e teste
Sem PCA	93,2 %	90,4 %	12182 s
Com PCA 36 para 36	93,2 %	90,6 %	12361 s
Com PCA 36 para 30	93,2%	90,9 %	4793 s
Com PCA 36 para 25	92,7 %	90,6 %	3099 s

fala, usando o algoritmo PCA padrão (linear) e o *kernel* PCA [77]. A diferença entre os dois é que o PCA padrão analisa os dados no espaço de entrada e o *kernel* PCA analisa os dados no espaço característico.

Também na literatura foi verificado que o resultado esperado com a utilização da técnica de PCA (linear ou não) é a diminuição do custo computacional sem grande perda de desempenho [77].

5.6.2 Implementações

Os algoritmos utilizados nos experimentos com SVM foram implementados tendo como base a *toolbox* de Anton Schwaighofer [55] onde estão contidas as funções básicas para implementação de um sistema com SVM. Pode-se dizer também que esta *toolbox* é uma implementação em Matlab do programa SVM light de Joachims [54], o qual foi implemetado em linguagem C.

As funções desta *toolbox* se constituem basicamente do treinamento e teste de um classificador binário. Par a realização deste trabalho, foram implementados os métodos utilizados para estendê-lo para várias classes, assim como toda a manipulação dos dados de voz.

5.6.3 Uso de memória

A quantidade de memória (tanto disco rígido quanto memória RAM) utilizada pelo sistema é proporcional ao número de amostras e a capacidade de generalização do sistema.

Pode-se exemplificar isto da seguinte forma:

Um classificador binário com os parâmetros ótimos (parâmetros do *kernel* e de penalização) terá uma maior generalização e, assim, utilizará menos memória que um classificador com parâmetros não ótimos.

Da mesma forma um classificador treinado com o método um contra um utilizará menos memória que um classificador treinado com o método “um contra todos”.

5.6.4 Conclusões

Foi observado nos experimentos, que utilizaram o treinamento do "um contra um", que o número de vetores suporte se mantém na faixa de aproximadamente 10 à 20 % do total de amostras de treinamento para os parâmetros ótimos ou nas suas proximidades e em aproximadamente 2 % quando o treinamento do "um contra todos" é utilizado. Isto proporciona uma diminuição do tempo de processamento e de espaço em memória, com relação ao treinamento e teste com parâmetros não ótimos.

O maior desempenho conseguido com o treinamento do "um contra todos", indica que quanto mais amostras são utilizadas no treinamento de cada classificador binário, maior será a sua capacidade de generalização.

Com o método PODH, obteve-se uma grande redução do custo computacional sem que houvesse perda de desempenho: na fase de testes, este método se mostrou quase duas vezes mais rápido que o DAG (190%), 23 vezes mais rápido que o método "um contra um"(2308%), e 1,3 vezes mais rápido que o método um contra todos (130%). Com relação ao tempo de treinamento, este método se mostrou 21,4 vezes mais rápido que o método "um contra todos"(2143%). Logo esta metodologia é muito útil quando o fator tempo é preponderante.

A comparação entre os melhores resultados de diferentes sistemas de reconhecimento de fala utilizando o mesmo banco de dados é dada na tabela (5.9). Os sistemas utilizados foram: SVM, HMM semi-contínuo [74], ANN [74] e o híbrido HMM + ANN. Os resultados referem-se aos testes feitos com 3 repetições para cada locução.

Tabela 5.9: *Comparando os melhores resultados de diferentes sistemas de reconhecimento de fala.*

SVM	HMM semi-contínuo	ANNs (Redes Neurais)	HMM + ANN
92,18 %	99,47 %	93,82 %	99,00 %

Pela tabela 5.9 pode-se observar que o melhor resultado obtido com o HMM ou com sistemas híbridos HMM/ANN foi maior que os melhores resultados com SVM ou redes neurais (ANNs).

Isto ocorre na tarefa de reconhecimento de palavras isoladas, porque redes neurais e SVM apresentam dificuldades para trabalhar com variações temporais do sinal de fala.

Apesar deste resultados, o SVM mostra ter o mesmo desempenho que o HMM em tarefas mais complexas como por exemplo o reconhecimento de fala contínua [6].

Capítulo 6

Conclusões e extensões deste trabalho

6.1 Conclusões

Este trabalho estudou o comportamento do SVM na classificação de palavras isoladas. Para isto foi utilizado o *Kernel* RBF. As características de voz que foram analisadas são os coeficientes melcepstrais e suas primeira e segunda derivadas.

Um algoritmo diferente para o teste da SVM foi implementado (Podh) e seu desempenho e velocidade foram muito satisfatórios.

A técnica que obteve o maior desempenho com relação à taxa de acertos foi a variação do método "um contra todos" com a retirada do sign, que obteve um desempenho de 92,18%. Entretanto, este método apresentou um tempo de processamento muito grande.

O método que apresentou o menor tempo de processamento, aliado a um bom desempenho foi o método Podh.

O *kernel* que obteve o maior desempenho neste trabalho e que é o mais utilizado na literatura de reconhecimento de fala é o RBF.

O PCA mostrou ser útil para a redução do custo computacional, sem perda de desempenho quando utilizada em conjunto com a técnica de SVM.

Como foi visto na tabela 5.9, o desempenho conseguido pelo SVM no reconhecimento de palavras isoladas foi muito próximo ao que foi conseguido na literatura com a utilização de redes neurais. Pode-se verificar também que o desempenho dos sistemas que utilizam HMM ou Híbridos HMM + ANN para o reconhecimento de fala foram superiores às técnicas ANN e SVM. Pode-se explicar isto pelo fato que redes neurais e SVMs apresentam dificuldades para trabalhar com variações temporais do sinal de fala.

Por outro lado, pela literatura disponível, pode-se comprovar que a utilização da técnica de SVMs mostrou-se compatível com o HMM em muitas aplicações envolvendo classificação de fala. Como exemplo pode-se citar o reconhecimento de fala contínua [6, 78] e a segmentação automática [30].

A técnica de SVM também tem as suas desvantagens, uma delas é a ausência de uma metodologia sistemática e de baixo custo computacional para a escolha do melhor *kernel*, dos seus parâmetros e do parâmetro de penalização C [48].

6.2 Extensões deste trabalho

1. Estudar métodos alternativos para a escolha do melhor *kernel*, do(s) seu(s) parâmetro(s) e do parâmetro de penalização C;

2. Comparar o desempenho de diferentes *Kernels*;

3. Implementar e comparar outras metodologias para classificação de múltiplas classes (ecoc);

4. Utilizar o SVM para classificação fonética.

5. Implementar um híbrido HMM/SVM [27] [72] e comparar com um híbrido HMM/ANN [79] em tarefas de reconhecimento de fala.

6. Utilizar a combinação de diferentes SVMs para classificar diferentes características, isto é, SVMs com diferentes *kernels*, escolhidos via dimensão VC [51] e pela validação cruzada, utilizados para classificar separadamente características mel, delta mel e delta delta mel.

7. Testar a utilização da escolha do *kernel* pela dimensão VC ($h \leq \min(\|\mathbf{W}\|^2 R^2, n)+1$). Comparar os resultados ao artigo de Chapelle [51]. Uma outra proposta é uma variação da escolha pela dimensão VC. A diferença é que uma ou mais amostras são deixadas de fora para o cálculo da hiperesfera. Utilizar diversos *kernels* para comparação.

8. Utilizar o SVM (indutivo ou transdutivo) para criar uma máquina que, após a classificação de uma determinada quantidade de amostras de teste, utilize as mesmas para treinar novamente a máquina. Esta será, então, uma máquina adaptativa. (E, se necessário, haverá nova escolha do *kernel* e seus parâmetros).

9. Estudo de informações suplementares para auxiliar no reconhecimento de fala com SVM. Como por exemplo: gestos, imagens do rosto e da boca, reconhecimento do nível de *stress* na fala, etc.

10. Estudar o reconhecimento de palavra chave com SVM (*Word Spotting*).

11. Implementar a detecção de palavra fora do vocabulário com SVM para aplicações em reconhecimento de fala.

12. Utilizar diferentes valores do parâmetro C [52] em tarefas de classificação fonética ou outras que tenham dados desbalanceados.

13. Estudo de diferentes formas de obter probabilidade posterior na saída do SVM. Assim, estudo de diferentes formas de codificação e decodificação. Por exemplo LS-SVMs e ECOC [80].

Anexo A

Glossário de conceitos referenciados na tese

A.1 Análise de componente principal

A análise de componente principal (*Principal component analysis* - PCA) está relacionada à explicação da estrutura de variância-covariância, através de algumas poucas combinações lineares das variáveis originais. Seus objetivos gerais são a redução e a interpretação dos dados [81].

A Análise de Componente Principal procura disponibilizar um conjunto de eixos ortogonais ao longo dos quais os dados possam ser projetados, esperando assim representar a maior parte da informação com apenas os principais eixos (componentes) deste novo espaço.

Para reproduzir toda a variabilidade de um sistema são necessárias p componentes. Entretanto, se a maior parte desta variabilidade puder ser avaliada por um número menor k de componentes principais, irá existir quase a mesma quantidade de informação nas k componentes quanto nas p variáveis originais. Desta forma, as k componentes principais poderiam substituir as p variáveis originais, e o conjunto de dados inicial, consistindo de n medidas em p variáveis, é reduzido para outro, consistindo de n medidas em k componentes principais.

Além disso, geralmente revela relações que não eram previamente imaginadas e portanto permite novas interpretações sobre os fenômenos estudados.

Algebricamente, componentes principais são combinações lineares particulares das p variáveis aleatórias $\mathbf{X_1}$, $\mathbf{X_2}$, ..., $\mathbf{X_p}$. Geometricamente, estas combinações lineares representam a seleção de um novo sistema de coordenadas, obtido pela rotação do sistema original com $\mathbf{X_1}$, $\mathbf{X_2}$, ..., $\mathbf{X_p}$ como eixos coordenados. Os novos eixos representam as direções com máxima variabilidade e fornecem uma

descrição mais simples e parcimoniosa da estrutura de covariância. A teoria de análise de componente principal pode ser resumida nos dois próximos resultados.

Resultado 1: seja $\mathbf{\Sigma}$ a matriz de covariância associada ao vetor aleatório $\mathbf{X'} = [x_1, x_2, \ldots, x_p]$. Se os autovalores e autovetores de Σ são dados por $(\lambda_1, \mathbf{e}_1)$, $(\lambda_2, \mathbf{e}_2)$, ..., $(\lambda_p, \mathbf{e}_p)$, onde $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$, então a i-ésima componente principal é dada por

$$Y_i = \mathbf{e}_i{}'\mathbf{X} = e_{i1}X_1 + e_{i2}X_2 + \ldots + e_{ip}X_p,\ i = 1, 2, \ldots, p \tag{A.1}$$

Com estas escolhas,

$$\text{var}(Y_i) = \mathbf{e}_i{}'\Sigma\mathbf{e}_i = \lambda_i,\ i = 1, 2, \ldots, p \tag{A.2}$$

$$\text{cov}(Y_i, Y_k) = \mathbf{e}_i{}'\Sigma\mathbf{e}_k = 0,\ i \neq k \tag{A.3}$$

Este resultado mostra que as componentes principais são descorrelacionadas e têm variâncias iguais aos autovalores de Σ.

Resultado 2: seja o vetor $\mathbf{X'} = [x_1, x_2, \ldots, x_p]$ com matriz de covariância Σ, com autovalores e autovetores dados por $(\lambda_1, \mathbf{e}_1)$, $(\lambda_2, \mathbf{e}_2)$, ..., $(\lambda_p, \mathbf{e}_p)$, onde $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$. Seja a i-ésima componente principal dada por $Y_i = \mathbf{e}_i{}'\mathbf{X} = e_{i1}X_1 + e_{i2}X_2 + \ldots + e_{ip}X_p$, $i = 1, 2, \ldots, p$. Então

$$\sigma_{11}^2 + \sigma_{22}^2 + \ldots + \sigma_{pp}^2 = \sum_{i=1}^{p} \text{var}(X_i) = \lambda_1 + \lambda_2 + \ldots + \lambda_p = \sum_{i=1}^{p} \text{var}(Y_i) \tag{A.4}$$

Este resultado mostra que a variância total é dada por

$$\sigma_{11}^2 + \sigma_{22}^2 + \ldots + \sigma_{pp}^2 = \lambda_1 + \lambda_2 + \ldots + \lambda_p \tag{A.5}$$

e conseqüentemente, a proporção da variância total devida (explicada) à k-ésima componente principal é dada por

$$\frac{\lambda_k}{\lambda_1 + \lambda_2 + \ldots + \lambda_p}, textk = 1, 2, \ldots, p \tag{A.6}$$

Se a maior parte (por exemplo, 80 a 90%) da variância populacional total puder ser atribuída às primeiras, uma, duas ou três componentes, estas poderiam "substituir" as p componentes originais sem muita perda de informação.

É importante notar que embora os resultados acima sejam calculados a partir da matriz de covariância σ, os mesmos resultados seriam conseguidos usando a

matriz de correlação R.

A.2 Dualidade Lagrangeana

Dado o problema primal:

$$\min f(w),\ w \in X \subseteq R^n \tag{A.7}$$

$$\text{sujeito a } \begin{cases} g_i(w) \leq 0,\ i = 1, ..., k \\ h_i(w) = 0,\ i = 1, ..., m \end{cases} \tag{A.8}$$

existe outro problema associado a ele, chamado problema dual.

Existem diversas formulações para o problema dual, correspondente ao problema primal, dentre elas a dualidade Lagrangeana, que resolve problemas lineares e não-lineares convexos ou não. Está formulação também pode ser aplicada a problemas de otimização discreta, onde algumas ou todas as variáveis são inteiras.

A formulação para o problema dual Lagrangeano é a seguinte:

$$\max \Theta(\alpha, \beta) \tag{A.9}$$

$$\text{sujeito a } \begin{cases} \alpha \geq 0 \end{cases} \tag{A.10}$$

$$\text{onde } \Theta(\alpha, \beta) = \inf\{f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{j=1}^{m} \beta_j h_j(w)\} \tag{A.11}$$

e inf é o mínimo global.

Neste problema as restrições $g_i(w) \leq 0$ e $h_i(w) = 0$ são incorporadas à função objetivo, utilizando os multiplicadores de lagrange α_i e β_i. O multiplicador α_i associado à restrição $g_i(x) \leq 0$ deve ser maior ou igual a zero. Já para o multiplicador β_i, associado a $h_i = 0$, não existe restrição de sinal. Em alguns casos, $\Theta(\alpha, \beta)$ pode assumir o valor $-\infty$ para algum vetor (α, β). Desde que o dual consista em maximizar o ínfimo (maior limite inferior simbolizado por inf) da função $f(w)+\sum_{i=1}^{k} \alpha_i g_i(w)+\sum_{j=1}^{m} \beta_j h_j(w)$, ele é chamado de max-min problema dual.

O problema dual pode ser escrito como $sup\{\Theta(u, v) : u \geq 0\}$ desde que o máximo para o problema não exista.

O problema primal e o dual podem ser escritos também na forma vetorial, onde $f : \Re^n \to \Re, g : \Re^n \to R^m$ é uma função vetorial com componentes g_i e $h : \Re^n \to \Re^L$, com componentes h_i,donde

O problema primal

$$\min f(x) \text{ sujeito a } \begin{cases} g(x) \leq 0 \\ h(x) = 0 \\ x \in X \end{cases} \tag{A.12}$$

O problema dual Lagrangeano

$$\max \Theta(u, v) \text{sujeito a } \begin{cases} u \geq 0 \end{cases} \tag{A.13}$$

onde $\Theta(u, v) = \inf\{f(x) + u^T g(x) + v^T h(x) : x \in X\}$.

A.3 Problema de otimização quadrático e problema de otimização convexo

Com base em [39] temos que o problema de otimização quadrático é um problema em que a função-objetivo é quadrática, enquanto que as restrições são todas lineares. Já o problema convexo tem sua função objetivo e todas as restrições convexas. Como no problema de treinamento para SVMs, as restrições são lineares e a função-objetivo é convexa e quadrática, o problema de otimização também será convexo e quadrático.

Para resolver este tipo de problema temos de utilizar 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 Tucher estenderam o método e permitiram restrições de desigualdade. Este novo método conduz às condições de Kuhn-Tucker.

A.4 Condições de Kuhn-Tucker

Dado o problema A.7 com o domínio convexo , $f \in C^1$ convexa, g_i, i=1,...,k e h_j, j=1,...,m sendo funções afins, a condição necessária e suficiente para o ponto w^* ser ótimo, é a existência de α^* , β^* satisfazendo:

$$\begin{cases} \frac{\partial L}{\partial w}(w^*, \alpha^*, \beta^*) = 0; \\ \frac{\partial L}{\partial \beta}(w^*, \alpha^*, \beta^*) = 0; \\ \alpha_i^* g_i(w^*) = 0, i = 1, ..., k; \\ g_i(w^*) \leq 0, i = 1, ..., k; \\ \alpha_i^* \geq 0, i = 1, ..., k. \end{cases} \tag{A.14}$$

A.5 Validação cruzada

Na validação cruzada com 10 campos (em inglês *10 fold cross validation*) as amostras são randomicamente divididas em 10 grupos de tamanhos iguais . Um grupo é separado para ser utilizado como conjunto de teste e os outro 9 grupos restantes como dados de treinamento. Após o treinamento do SVM utilizando os 9 grupos, o grupo de teste é apresentado ao sistema . A média do erro absoluto é então calculada. Este processo é repetido para os outros 9 grupos. Depois são calculados a média e o desvio padrão do erro absoluto para todos os grupos.

1. Dividir o conjunto de treinamento em 10 blocos do mesmo tamanho;
2. Escolher um intervalo para os parâmetros do *kernel* e para o parâmetros C;
3. Dado um parâmetro do *kernel* e um parâmetro C definido no intervalo acima
4. Repitir 10 vezes:
   4.1 Retirar um bloco
   4.2 Treinar o classificador sobre os outros nove blocos
   4.3 Testar o sistema com o bloco restante
   4.4 Calcular a média e o desvio padrão do erro absoluto
5. Calcular a média e o desvio padrão do erro absoluto para todas as 10 repetições.
6. Guardar o valor médio e o desvio padrão do erro absoluto.
7. voltar ao passo 3
8. escolher o valor mínimo da média e do desvio padrão será encontrado o parâmetro do *kernel* e o parâmetro C.

Anexo B

Locuções utilizadas na tese

Número da palavra	Palavra	Número da palavra	Palavra
1	zero	26	nordeste
2	um	27	sul
3	dois	28	sudeste
4	três	29	centro-oeste
5	quatro	30	esportes
6	cinco	31	departamento
7	seis	32	divisão
8	sete	33	seção
9	oito	34	coordenação
10	nove	35	imagem
11	meia	36	voz
12	sim	37	áries
13	não	38	touro
14	terminar	39	câncer
15	repetir	40	leão
16	continuar	41	gêmeos
17	voltar	42	virgem
18	avançar	43	libra
19	certo	44	escorpião
20	errado	45	capricornio
21	opções	46	sagitário
22	dólar	47	aquário
23	real	48	peixes
24	tempo	49	horóscopo
25	norte	50	ajuda

Tabela B.1: *As 50 locuções utilizadas na tese.*

Referências Bibliográficas

[1] Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue, "Survey of the state of the art in human language technology.", *http://cslu.cse.ogi.edu/publications/index.htm*, 1998.

[2] MIT, "Automatic speech recognition course", http://ocw.mit.edu-/OcwWeb/Electrical-Engineering-and-Computer-Science/6-345Automatic-Speech-RecognitionSpring2003/CourseHome/.

[3] L. R .Rabiner and B.H.Juang, *Fundamentals Of Speech Recognition*, Prentice Hall, Inc, Englewood Cliffs, NJ, USA, 1993.

[4] J. Tebelskis, *Speech Recognition Using Neural Networks*, PhD thesis, Carnegie Mellon University, 1995.

[5] Douglas A. Reynolds, "Speaker identification and verification using gaussian mixture speaker models", *Speech Commun.*, vol. 17, no. 1-2, pp. 91–108, 1995.

[6] Hamaker Ganapathiraju and Picone, "Continuous speech recognition using support vector machines", *Computers, Speech and Language*, 2001.

[7] Vladimir Vapnik, *The Nature of Stastistical learning Theory*, Springer, New York, 1995.

[8] C. Cortes and V. Vapnik, "Support vector networks", *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[9] N. Cristiannini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

[10] Christopher J. C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[11] Vladmir N. Vapnik, "An overview of statistical learning theory", *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–1000, September 1999.

[12] J. Weston and C. Watkins, "Multi class support vector machines", *in Proceedings of ESANN99*, pp. 219–224, 1999.

[13] J. Shawe-Taylor J. C. Platt, N. Cristianini, "Large margin dags for multiclass classification", *to appear in Neural Information Processing Systems*, , no. 12, 2000.

[14] Vapnik V., "The nature of stastistical learning theory", *Spring Velarg*, 1999.

[15] Alex Smola and Bernhard Schölkopf, "Bibliography of svm papers", http://www.kernel-machines.org/publications.html.

[16] A.J.Smola, "Regression estimation with support vector learning machines", Master's thesis, Technische Universität München, 1996.

[17] G.Rässch B.Schölkopf J.Kohlmorgen K.R.Müller, A.Smola and V. Vapnik, "Predicting time series with support vector machines", *In B.Schölkopf, C.J.C Burges, and A.J.Smola, editors, Advances in kernel Methods- Support Vector Learning*, pp. 243–254, 1999.

[18] D.tax and R.Duin, "Data domain description by support vector", *In ESANN99*, pp. 251–256, 1999.

[19] A. Smola J. Shawe-Taylor B. Schölkopf, R. Williamson and J. Platt, "Support vector method for novelty detection", http://citeseer.ist.psu.edu/sch00support.html.

[20] Colin Campbell and Kristin P. Bennett, "A linear programming approach to novelty detection", in *NIPS*, 2000, pp. 395–401.

[21] E.Osuna, R.Freund, and F. Girosi, "Training support vector machines: An application to face detection", *In Proceedingings of CVPR'97*, 1997.

[22] T. Joachims, "Text categorization with support vector machines:learning with many relevant features", *In Proc. 10th European Conference on Machine Learning EMCL-98*, pp. 137–142, 1998.

[23] R.Cooley, "Classification of news stories using support vector machines", *In Proc. 16th International Joint Conference on Artificial Intelligence Text Mining Workshop*, 1999.

[24] C. Burges B.Schölkopf and V. Vapnik, "Extracting support data for a given task", *Proceedings Fisrt InternationalConference on Knowledge Dicovery and Data Mining Menlo Park*, 1995.

[25] Isabelle Guyon, "Svm application list", http://www.clopinet.com/isabelle/-Projects/SVM/applist.html.

[26] C.-H. Lee ans L. R. Rabiner, "On the use of support vector machines for phonetic classification", *Speech and Signal Processing*, vol. II, pp. 585–588, 2000.

[27] A. Ganapathiraju, J. Hamaker, and J. Picone, "Hybrid svm/hmm architectures for speech recognition", 2000.

[28] Yassine Benayed at al, "Confidence measures for keyword spotting using support vector machine", *ICASSP*, 2003.

[29] "Speechdat projects", http://www.speechdat.org/.

[30] Amit Juneja and Carol Espy-Wilson, "Speech segmentation using probabilistic phonetic feature hierarchy and support vector machines".

[31] Mihaela Gordan Constantine, "Application of support vector machines classifiers to visual speech", http://citeseer.ist.psu.edu/667892.html, 2002.

[32] J. R. Movellan, "Visual speech recognition with stochastic networks", *in Advances in Neural Information Processing Systems*, vol. 7, 1995.

[33] V. Vapnik, *Stastistical learning theory*, John Wiley, New York, USA, 1998.

[34] Andrew Moore, "Vc - dimension for characterizing classifiers", http://www-2.cs.cmu.edu/ awm/tutorials/vcdim08.pdf, 2001.

[35] R. Duda and P. Hart, "Pattern classification and scene analysis.", *John Wiley and Sons.*, 1978.

[36] Simon Haykin, *Redes neurais: princípios e prática*, Bookman, 2001.

[37] Murray W. Gill, P.E. and M.H. Wright, "Pratical otimization", *Academic Press*, 1981.

[38] D. Bertsekas, "Nonlinear programming", *Nashua,NH: Athena Scientific*, 1995.

[39] Robinson Semolini, "Support vector machines, inferência transdutiva e o problema da classificação", Master's thesis, Unicamp, 2002.

[40] SCHNABEL R. DENNIS, J. E., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Pratice Hall, 1983.

[41] UFMG, "Cálculo integral e diferencial ii - tabela de quádricas", http://www.mat.ufmg.br/ syok/cursos/mat039/quadricas/quadricas.htm.

[42] Stephan Bergman, *The kernel function and conformal mapping.*, AMS, Providence, 1970.

[43] Nachman Aronszajn, "Theory of reproducing kernels.", *Trans. Amer. Math. Soc.*, vol. 69(3), pp. 337–404, 1950.

[44] J. Mercer, "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, 1909.

[45] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[46] Gunn S., "Support vector machines for classification and regression", *Technical Report ISIS-1-98, Image Speech & Intelligent Systems Group*, 1998.

[47] C.-J.Lin C.-W. Hsu, "A comparison on methods for multi-class support vector machines", *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.

[48] C. A. M. Lima, *Comitê de Máquinas:Uma Abordagem Unificada Empregando Máquinas de Vetores-Suporte*, PhD thesis, UNICAMP, 2004.

[49] A. J. Smola and B. Schölkopf, "On a kernel based method for pattern recognition,regression, approximation and operator inversion.", April 1997.

[50] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function aproximation, regression estimation, and signal processing.", *In M. Mozer, M. Jordan, and T. Petsche (Eds): Neural Information Processing Systems*, vol. 9, 1997.

[51] O. Chapelle and V. Vapnik, "Model selection for support vector machines", 2000, to appear in Advances in Neural Information Processing Systems 12, ed. S.A. Solla, T.K. Leen and K.-R. Muller, MIT Press, 2000.

[52] N. Cristianini K. Veropoulos and C. Campbell, "The application of support vector machines to medical decision support: A case study", *Department of Engineering Mathematics, Bristol University, Bristol BS8 1TR, United Kingdom*, 1999.

[53] F. Girosi E.Osuna, R.Freund, "Training support vector machines: An application to face detection.", *In Proceedingings of CVPR'97*, 1997.

[54] Thorsten Joachims, "Svm ligth", http://svmlight.joachims.org/, 1999.

[55] Anton Schwaighofer, "Svm toolbox for matlab", http://www.igi.tugraz.at/-aschwaig/software.html, 2002.

[56] Lin-C.-J. Hsu, C.-W., "A comparison on methods for multi-class support vector machines. ieee transactions on neural networks", *IEEE Transactions on Neural Networks 13(2):415-425*, March 2002.

[57] S. Knerr, L. Personnaz, and G. Dreyfus ., "Single-layer learning revisited: A stepwise procedure for building and training a neural network.", *In F. Fogelman-Souli´e and J. H´erault, editors, Neurocomputing: Algorithms, Architectures and Applications*, pp. 41–50, 1990.

[58] J.H.Friedman, "Another approach to polychotomous classification", 1996.

[59] U. Krebel, "Pairwise classification and support vector machines", *In B. Sch¨olkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods Support Vector Learning*, pp. 255–268, 1999.

[60] C. Hsu and C. Lin, "A comparison of methods for multi-class support vector machines", 2001.

[61] Tom G. Dietterich and Ghulum Bakiri, "Error-correcting output codes: A general method for improving multiclass inductive learning programs.", *In Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 572–577, 1991.

[62] Bernard Sklar, *Channel coding. In: Digital communications: fundamentals and applications.*, Prentice Hall, Upper Saddle River, New Jersey, 2001.

[63] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers.", *In Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 9–16, 2000.

[64] Rayid Ghani, "Using error-correcting codes for text classification.", *In Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 303–310, 2000.

[65] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification.", *In S. A. Solla, T. K. Leen, and K.-R. Muller, editors, Advances in Neural Information Processing Systems.*, 2000.

[66] Jesper Salomon, "Support vector machines for phoneme classification", Master's thesis, University of Edinburgh, 2001.

[67] Y. Guermeur, "Combining discriminant models with new multi-class svms.", *Technical Report NeuroCOLT2, 2000-086*, 2000.

[68] E. Bredensteiner and K. P. Bennet., "Multicategory classification by support vector machines.", *Computational Optimizations and Applications*, pp. 53–79, 1999.

[69] K. Crammer and Y. Singer., "On the algorithmic implementation of multiclass kernel-based vector machines.", *Journal of Machine Learning Research*, 2001.

[70] Celso Roberto Moraes, *Estruturas de dados e algoritmos*, Futura, São Paulo, 2003.

[71] Joseph W. Picone, "Signal modeling techniques in speech recognition", *Proceedingings of the IEEE*, vol. 81, no. 9, pp. 1215–1247, 1993.

[72] J. Hamaker A. Ganapathiraju and J. Picone, "Advances in hybrid svm/hmm speech recognition", *in GSPx / Interna-tional Signal Processing Conference*, 2003.

[73] Chih-Chung Chang Chih Wei Hsu and Chih-Jen Lin, "A pratical guide to support vector classification", 2003.

[74] José Antônio Martins, *Avaliação de Diferentes Técnicas para o Reconhecimento de Fala*, PhD thesis, UNICAMP, 1997.

[75] Joseph W. Picone, "Signal modeling techniques in speech recognition", *Proceedings of The IEEE*, vol. 81, no. 09, pp. 1215–1247, September 1993.

[76] Flanery-B. P. Teukolsky S. A. Press, W. H. and W. T. Veterlling, *Numerical recipies - the art of cientific computing.*, Cambridge University Press, Cambridge, 1987.

[77] A. Smola B. Schölkopf and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem", *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[78] J. Hamaker A. Ganapathiraju and J. Picone, "Applications of support vector machines to speech recognition", *IEEE Transactions on Signal Processing*, vol. 52, no. 08, pp. 2348–2355, August 2004.

[79] H. Bourlard and N. Morgan, "Connectionist speech recognition - a hybrid approach", *Kluwer Academic Publishers*, 1994.

[80] T. Van Gestal, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle., "Bayesian framework for leastsquares support vector machine classifiers, gaussian processesand kernel fisher discriminant analysis.", *Neural Computation*, vol. 14, pp. 1115–1147, 2002.

[81] Richard Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall, New Jersey, 1998.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo