Teoria e Aplicação de *Support Vector Machines* à Aprendizagem e Reconhecimento de Objetos Baseado na Aparência

Eulanda Miranda dos Santos

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba, Campus II, como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Modelos Computacionais e Cognitivos

Herman Martins Gomes

Orientador

Campina Grande, Paraíba, Brasil

S237T

SANTOS, Eulanda Miranda dos

Teoria e Aplicação de Support Vector Machines à Aprendizagem e Reconhecimento de Objetos Baseado na Aparência.

Dissertação de Mestrado, Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande – PB, Junho de 2002.

111 p. Il.

Orientador: Herman Martins Gomes

Palavras Chave:

1.Inteligência Artificial
2.Visão Computacional
3.Support Vector Machines
4.Aprendizagem de Máquina

CDU – 007:52

TEORIA E APLICAÇÃO DE *SUPPORT VECTOR MACHINES* À APRENDIZAGEM E RECONHECIMENTO DE OBJETOS BASEADO NA APARÊNCIA"

EULANDA MIRANDA DOS SANTOS

DISSERTAÇÃO APROVADA EM 20.06.2002

PROF. HERMAN MARTINS GOMES, Ph.D
Orientador

PROF. MARCELO ALVES DE BARROS , Dr.
Examinador

PROF. JOÃO MARQUES DE CARVALHO, Ph.D
Examinador

CAMPINA GRANDE – PB

Agradecimentos

A Deus, autor e consumador da minha fé, que deu as condições necessárias para que eu pudesse chegar até aqui. Certamente, a alegria do Senhor é a nossa força e Ele foi a pessoa chave para que esse trabalho fosse realizado, porque nele, por Ele e para Ele são todas as coisas. A Ele pois, a glória eternamente. Amém!

Ao professor Herman Martins Gomes que pacientemente acompanhou e orientou esse trabalho de forma participativa e flexível. Soube direcionar as atividades pelo melhor caminho, demonstrando confiança e motivação.

À minha grande família, que apesar de distante, em nenhum momento permitiu que eu me sentisse sozinha. Acompanharam-me, especialmente com suas orações, carinho e palavras de incentivo.

A todos que fazem a COPIN. Aos professores que procuraram da melhor forma, atender-me em minhas necessidades. À Aninha e Vera por toda a atenção dispensada a mim. À Zeneide e demais funcionários da miniblio e à Eleonora da miniblio da Copele.

Aos amigos que consquistei durante esse período e que certamente contribuíram para este trabalho, seja através de sugestões, de apoio, incentivo e, até mesmo, através de momentos de descontração, sempre necessários ao ser humano.

À Chris, que sabe na prática todo o significado do texto de Provérbios "Há amigos mais chegados que um irmão".

Aos membros da banca examinadora pela atenção e participação.

A todos que direta ou indiretamente contribuíram de alguma forma, para o desfecho de mais essa etapa da minha vida.

Resumo

Support Vector Machines (SVM) é uma técnica de aprendizagem de máquina derivada de duas fundamentações sólidas: Teoria da Aprendizagem Estatísta e Otimização Matemática. SVM têm sido recentemente aplicado com sucesso a uma variedade de problemas que vão desde o reconhecimento de caracteres ao reconhecimento de objetos baseado na aparência.

Alguns dos motivos para esse sucesso estão relacionados ao fato dessa técnica exibir bom desempenho de generalização em muitas bases de dados reais, é bem fundamentada teoricamente, o processo de treinamento elimina a possibilidade de mínimos locais, existem poucos parâmetros livres para ajustar e a arquitetura não precisa ser encontrada por experimentação. Entretanto, por tratar-se de uma abordagem relativamente nova, livros-texto e artigos estão geralmente disponíveis em uma linguagem que não é facilmente acessível para Cientistas da Computação. Portanto, um dos objetivos desta dissertação é prover uma introdução sobre SVM que apresente os conceitos e teoria essenciais à técnica e que seja mais didática.

Estratégias de reconhecimento de objetos com base na aparência se aplicam a problemas em que há dificuldades na obtenção de modelos geométricos dos objetos, desde que as imagens utilizadas não apresentem oclusões. Algumas técnicas de aprendizagem de máquina têm sido aplicadas a este problema, tais como: PCA (*Principal Component Analysis*), PAC (*Probably Approximately Correct*) e Redes Neurais, mas nenhuma mostrou-se tão promissora quanto SVM.

Dentro desse contexto, esta dissertação objetiva investigar a aplicação de SVM ao reconhecimento de objetos baseado na aparência. Apresenta resultados práticos de classificação utilizando inicialmente uma pequena base de dados e, em seguida, explorando todo o poder da técnica em uma base de dados relativamente grande. Esta dissertação também descreve resultados experimentais usando diferentes variações da técnica e compara o desempenho de reconhecimento de SVM com o desempenho de Redes Neurais do tipo *Multilayer Perceptron Backpropagation*.

Abstract

Support Vector Machines (SVM) is a machine learning technique derived from two solid backgrounds: Statistical Learning Theory and Mathematical Optimisation. SVM has recently been applied with success to a variety of problems, ranging from character recognition to appearance based object recognition.

Some of the reasons for this success are related to the fact this technique exhibits good generalisation performance on many real-life data sets, is well-founded theoreticaly, the training process eliminates the possibility of local minima, there are few free paramets to adjust and the architecture does not have to be found by experimentation. However, since this is a relatively new approach, text books and papers are usually in a language that is not easily acessible to Computer Scientists. Therefore one of the objectives of this dissertation is to provide an introduction to SVM that presents the essential concepts and theory behind the technique and that is more didatic.

Appearance-based object recognition strategies appear to be well-suited for the solution of recognition problems in which geometric models of the viewed objects can be difficult to obtain, although they are not naturally tolerant to occlusions. Some machine learning techniques have been applied in this problem like, Principal Component Analysis (PCA), Probably Approximately Correct (PAC) and Neural Networks, but none posed as promissing as SVM.

Within this context, this dissertation aims to investigate the application of SVM to appearance-based object recognition. It presents practical results of classification initialy using a small dataset and then exploring the full power of the technique on a relatively large dataset. It also presents experimental results using different variations of the technique and compares the recognition performance of SVM with the performance of Multilayer Perceptron Backpropagation Neural Networks.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Na maioria dos problemas de Inteligência Artificial (IA) que envolvem aprendizagem, é importante prover as máquinas de percepção, a fim de que possam adquirir conhecimento do ambiente. A percepção envolve os cinco sentidos, como no ser humano, porém, o reconhecimento de voz e a visão são os sentidos mais pesquisados [Russel and Norvig, 1995]. Esta dissertação investiga a aprendizagem de máquina em um problema de visão. Nesse contexto muitas tarefas têm sido investigadas, como por exemplo: detecção de imagens, reconstrução, reconhecimento etc.

Esta dissertação trata do reconhecimento de objetos baseado na aparência utilizando a técnica estatística de aprendizagem de máquina *Support Vector Machines* (SVM). Reconhecimento baseado na aparência não envolve a produção de modelos geométricos e sim, a percepção de visões do objeto (o mesmo objeto visto sob ângulos diferentes). Esse tipo de tarefa pode ser útil para a navegação de robôs, inspeção visual, sistemas de segurança, etc. Entretanto, antes de descrever o problema e a técnica investigados, é importante esclarecer o conceito e principais características da aprendizagem. A próxima seção tratará da aprendizagem de máquina com o objetivo de delimitar a abrangência desta dissertação.

1.1 Aprendizagem

Entidades inteligentes destacam-se pela capacidade de adequar-se a novos ambientes e de resolver novos problemas. Um computador pode ser orientado à interpretar as informações recebidas de uma forma que melhore gradualmente seu desempenho [Rich and Knight, 1991].

Há muitas discussões dentro da IA quanto à possibilidade de máquinas aprenderem de fato, entretanto, essa questão depende necessariamente da definição do termo aprendizagem.

Existem inúmeras atividades associadas com a noção de aprendizagem, dificultando a definição exata do termo e tornando-o questão de ponto de vista e de contexto. Aprendizagem para a Psicologia é diferente da aprendizagem que ocorre em sala de aula, entre outros. Em [Rich and Knight, 1991], página 447, encontra-se a seguinte definição de aprendizagem:

> "...mudanças adaptáveis no sistema, que permitem ao sistema fazer, da próxima vez, a mesma tarefa ou tarefas tiradas do mesmo grupo com mais eficiência e eficácia".

Independente da definição exata de aprendizagem, é fato que sistemas de aprendizagem confiáveis são de importância estratégica, pois há muitas tarefas que não podem ser solucionadas por técnicas de programação clássicas. Os algoritmos de aprendizagem de máquina, então, podem ser a chave para solucioná-los. A Figura 1.1 mostra um modelo simples de aprendizagem de máquina. O ambiente fornece informação para um elemento de aprendizagem que usa essa informação para fazer melhoramentos em uma base de conhecimento e então, um elemento de desempenho usa essa base para executar sua tarefa.

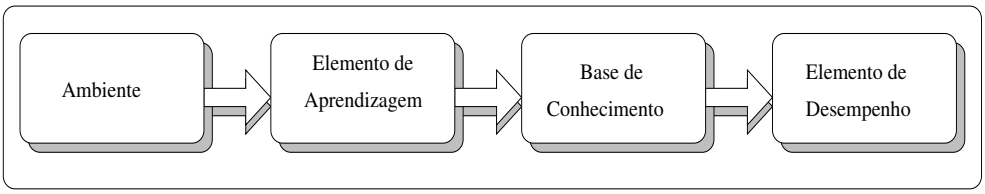

Figura 1.1: Modelo simples de aprendizagem de máquina.

A aprendizagem de máquina pode ser dividida, a nível genérico, em dois paradigmas fundamentais: aprendizagem com professor e aprendizagem sem professor [Haykin, 1999]. No primeiro, normalmente chamado *aprendizagem supervisionada*, o sistema precisa conhecer o ambiente. Esse conhecimento é representado por um conjunto de exemplos de pares de entrada-saída que são transmitidos em uma sequência de instruções que o computador seguirá para alcançar o efeito desejado. Na aprendizagem sem professor, não há exemplos rotulados da função a ser aprendida. Nesse paradigma são identificadas duas subdivisões:

- *Aprendizagem de reforço*: aprendizagem a partir de um mapeamento de entrada-saída alcançada através de interação continuada com o ambiente para minimizar um índice escalar de desempenho [Haykin, 1999].

- *Aprendizagem não-supervisionada*: não há valores de saída desejada. A tarefa de aprendizagem é ganhar alguma compreensão do processo que gerou o dado e desenvolver habilidades para formar representações internas capazes de codificar características da entrada e, portanto, criar novas classes automaticamente [Haykin, 1999]. Esse tipo de aprendizagem inclui estimação de densidade, formação de agrupamentos, dentre outros. A Figura 1.2 ilustra um diagrama de bloco de aprendizagem não supervisionada. O ambiente fornece informações sobre o estado do mundo ao sistema de aprendizagem, que faz os ajustes a partir dessas informações.

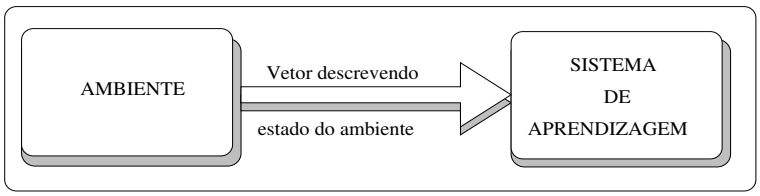

Figura 1.2: Diagrama de bloco de aprendizagem não supervisionada.

A aprendizagem de máquina pode também ser dividida, de uma forma mais específica, em relação à forma de desenvolver o conhecimento. Segundo [Ginsberg, 1993], essa divisão pode ser feita da seguinte forma: Aprendizagem por Descoberta e Aprendizagem por Generalização. Um outro tipo também importante é Aprendizagem por Analogia. Abaixo é enumerada esta divisão de aprendizagem proposta por [Ginsberg, 1993].

1. **Aprendizagem por Descoberta**: é uma forma restrita de aprendizagem que tenta descobrir novos fatos, conceitos, fenômenos etc, sem conhecimentos a priori e sem o auxílio de alguém que possua conhecimentos. Alguns exemplos de programas de IA que empregaram esta forma de aprendizagem são: Automatic Mathematician (AM), Eurisko, Bacon, Glauber e Boole.

2. **Aprendizagem por Generalização**: Objetiva gerar conhecimentos genéricos e abrangentes a partir de informações existentes. Pode ser subdividido em:

- Aprendizagem Dedutiva: tenta aprender novas informações que são consequências válidas de algo já conhecido, como em aprendizagem baseada em explicação.
- Aprendizagem Indutiva: baseia-se na inferência indutiva de fatos providos por um professor ou ambiente [Rich and Knight, 1991]. O sistema tenta induzir uma regra geral a partir de um conjunto de instâncias observadas. Alguns exemplos de técnicas indutivas são: Programa de Aprendizagem de Winston, PAC (*Probably Approximately Correct*), Espaços de Versão e Árvores de Decisão. Um outro método indutivo que está recebendo grande interesse atualmente é o princípio de Minimização do Risco Estrutural, implementado na técnica SVM. Esse método será descrito com detalhes no Capítulo 2 desta dissertação.

3. **Aprendizagem por Analogia**: A analogia é uma poderosa ferramenta de inferência. Permite que as semelhanças entre objetos ou fatos sejam declaradas de forma eficaz. Um sistema que aprende por analogia pode ser usado para converter um programa existente em um outro que realize tarefa semelhante ao original [Rich and Knight, 1991]. Há dois principais métodos para a solução de problemas por analogia:

   - Analogia por Transformação: tem como objetivo transformar a solução de um problema antigo na solução para o problema atual.
   - Analogia por Derivação: derivação nesse contexto, é a história detalhada do episódio da solução de um problema. A analogia por Derivação é a argumentação analógica que considera essas histórias. É um componente necessário na transferência de habilidades em domínios complexos.

Independente do tipo de aprendizagem, normalmente o conjunto de todas as regras definidas para a solução de um problema de aprendizagem é chamado algoritmo de aprendizagem. Há uma variedade de algoritmos que oferecem vantagens e desvantagens, dependendo do problema em que são aplicados. Esses algoritmos podem ter organização *top-down* ou *bottom-up*, ambos utilizados com muita frequência em IA. Uma organização *top-down* é construída a partir de procedimentos computacionais bem definidos e que podem incluir algum pré-armazenamento de conhecimento. Em uma organização *bottom-up* porém, não há regras claramente definidas. O sistema aprende a partir de sua experiência, necessitando portanto, que as regras sejam sujeitas à contínuas modificações [Penrose, 1995].

Cada organização pode oferecer vantagens, dependendo da classe de problemas em que seja aplicada. A *top-down* apresenta melhores resultados em certos problemas matemáticos (como por exemplo diferenciação ou integração simbólicas) e em problemas que envolvam diagnóstico médico, jogos de xadrez utilizando regras ou estratégias pré-programadas etc. Já a *bottom-up*, pode ser aplicada ao reconhecimento de faces ou sons, jogos de xadrez em que o computador aprende a partir de suas experiências, entre outros [Penrose, 1995]. Dentre os sistemas do tipo *bottom-up* mais conhecidos estão as Redes Neurais Artificiais, que compõem a parte da IA Conexionista ou Sub-Simbólica. A faixa *top-down* é característica da outra parte da IA, a Simbólica. A forma geral dos mecanismos simbólicos envolve um programa consistindo essencialmente de regras de alguma classe, armazenadas na memória com estruturas de dados apropriadas [Franklin, 1995]. Um outra forma utilizada é a construção de modelos conexionistas estruturados ou sistemas híbridos que integrem as duas abordagens.

Os sistemas de Redes Neurais Artificiais (RNAs) são muito utilizados principalmente devido à sua estrutura paralela e distribuída e à sua habilidade para aprender, e portanto, generalizar o conhecimento adquirido [Haykin, 1999]. Essa técnica também destaca-se pelas seguintes propriedades: aprendizagem de conjuntos linearmente inseparáveis, mapeamento de entrada-saída, adaptação, tolerância à falhas e uniformidade de análise e projeto. É importante destacar também que as RNAs são baseadas na minimizaçaõ do risco empírico, ou seja, minimização do erro de treinamento. Entretanto, essa técnica apresenta problemas como: algoritmos de aprendizagem envolvendo geralmente um grande número de exemplos de treinamento, longos períodos de treinamento, ocorrência de mínimos locais e arquitetura definida a partir de experimentos.

Alguns desses problemas não ocorrem em SVM, técnica de aprendizagem que tem uma forte fundamentação estatística e que tem recebido um crescente interesse nos últimos anos. Esse crescimento nas pesquisas pode ser observado através da proliferação de *papers* e seções especiais de SVM em conferências e simpósios de RNAs e aprendizagem de máquina [Burges, 1998]. O treinamento de SVM encontra uma solução global, enquanto que em RNAs há geralmente mínimos locais. Em SVM há a minimização do risco empírico, como em RNAs, juntamente com a Minização do Risco Estrutural, equivalente à minimização do erro de generalização, que envolve a imposição de um limite para diminuir ao máximo a possibilidade de classificações erradas na etapa de teste. Essa técnica também é atrativa

por sua habilidade de condensar as informações contidas no conjunto de treinamento. Essas características motivaram a escolha de SVM cpara o desenvolvimento desta dissertação, o qual trata do estudo e da aplicação de SVM a um problema de Visão de Máquina. A próxima seção apresenta sucintamente um histórico de SVM.

1.2 Breve Histórico de SVM

Os fundamentos de SVM são provenientes da Teoria de Aprendizagem Estatística desenvolvida inicialmente pelo pesquisador russo Vladmir Vapnik e colaboradores[Vapnik, 1999]. Vapnik idealizou o princípio indutivo de Minimização do Risco Estrutural. Este princípio busca minimizar o erro do conjunto de treinamento (risco empírico), juntamente com o erro do conjunto de teste. No Capítulo 2 esses conceitos são descritos matematicamente.

Segundo [Vapnik, 1999], a motivação para o desenvolvimento do princípio indutivo de Minimização do Risco Estrutural surgiu da necessidade de desenvolver limites teóricos para a capacidade de generalização dos sistemas de aprendizagem. Uma maior generalização normalmente implica um maior número de acertos na fase de teste. O princípio de Minimização do Risco Empírico, utilizado em técnicas como Redes Neurais, era considerado pelos seus defensores como sendo de fácil intuição e que, por apresentar bons resultados práticos, não necessitava de provas teóricas. Para esses defensores, uma boa generalização era obtida unicamente através da escolha de pesos para os neurônios de modo a prover um erro de treinamento mínimo.

Entretanto, para a análise teórica do processo de aprendizagem, esse conceito era visto de forma diferente. Em [Vapnik, 1999], página 7, encontra-se a seguinte afirmação:

> "O princípio de minimização do número de erros de treinamento não é evidente e precisa ser justificado. É possível que haja outro princípio indutivo capaz de prover um melhor nível de habilidade de generalização. O objetivo principal da análise teórica do processo de aprendizagem é encontrar um princípio indutivo com nível mais alto de habilidade de generalização e construir algoritmos que implementam esse princípio".

As respostas para essas pesquisas foram o princípio indutivo de Minimização do Risco

Estrutural e as Máquinas de Vetores de Suporte (*Support Vector Machines*) que realizam esse princípio.

Embora as pesquisas desenvolvidas com SVM sejam recentes, as raízes do método foram formuladas na década de 70. Houve uma paralização nas pesquisas por mais de três décadas. Em [Cristianini and Shawe-Taylor, 2000] os autores destacam dois problemas encontrados no período inicial, do ponto de vista algoritmico, que podem justificar essa parada nas pesquisas: 1 - o algorítmo original (Vapnik e Chervonenkis, 1974) era limitado à problemas de aprendizagem linearmente separáveis; 2 - não apresentava uma forma de tratar com dados isolados ou muito distantes do padrão original, os *"outliers"*.

Os dois problemas citados anteriormente foram solucionados durante a década de 90. O primeiro com o uso de funções Kernel, que fazem o mapeamento dos dados do espaço de entrada para um espaço de características com uma dimensão muito superior, possibilitando que os dados não separáveis linearmente, tornem-se separáveis no espaço de características. A idéia de utilização de funções kernel foi empregada pela primeira vez em 1964, no mesmo período em que Vapnik e Chervonenkis desenvolveram a idéia do hiperplano de separação ótima. Este hiperplano separa as diferentes classes, maximizando a margem entre os pontos extremos de cada classe. O uso combinado desses dois conceitos na formulação de SVM foi apresentado por Vapnik e colaboradores em 1992. O segundo problema foi solucionado através da introdução de variáveis de folga (Capítulo 2, Seção 2.2.2) para dar mais flexibilidade às restrições e permitir a classificação correta dos *"outliers"*.

Entretano, Vapnik afirma que as pesquisas não paralizaram. Em 1974 foram obtidos os limites para a introdução de um possível novo princípio indutivo (Minimização do Risco Estrutural), completando o desenvolvimento da teoria de aprendizagem para Reconhecimento de Padrões. Em [Vapnik, 1999], página 8, há a seguinte afirmação:

> "...construído durante 30 anos de análise do processo de aprendizagem, nos anos 90 começou a síntese das novas máquinas de aprendizagem capazes de controlar a habilidade de generalização".

Nos últimos anos têm havido desenvolvimentos significativos na compreensão teórica de SVM, em estratégias algorítmicas para implementá-lo e aplicações da técnica a diversos problemas práticos. Como já foi citado na seção anterior, esta dissertação investiga a aplicação

de SVM a problemas de Visão de Máquina. A próxima seção apresenta uma breve descrição da área de Visão de Máquina, com o intuito de delimitar o escopo da dissertação.

1.3 Visão de Máquina

Um dos sentidos que têm sido mais utilizados por máquinas para perceberem o ambiente em que estão inseridas é a visão [Russel and Norvig, 1995], sentido este que pode ser considerado uma tarefa de representação e processamento de informações, sendo portanto, passível de tratamento computacional [Marr, 1982]. Entretanto, a visão não é um problema de fácil tratamento computacional e, apesar dos vários anos de pesquisa na área, ainda não alcançou-se um conhecimento exato de como é realizada a nível fisiológico e psíquico, assim como muitos dos processos humanos mentais e perceptivos, que apresentam alguns aspectos desconhecidos para a ciência.

Diante disso, há dificuldades na definição de um conceito para visão. Dependendo do enfoque das pesquisas, o conceito pode variar. Há um certo consenso, entretanto, em torno da teoria de visão produzida por [Marr, 1982], onde a seguinte definição é apresentada:

> "...visão é o processo de descobrimento através de imagens do que está presente no mundo e onde está."

O foco principal é portanto responder às questões clássicas "*what*" e "*where*", ou seja, qual a identidade dos objetos e sua localização. Essa e outras definições evidenciam que a tentativa de simular o processo de visão em sistemas computacionais não é uma tarefa trivial e exige um certo grau de abstração.

A Visão de Máquina trabalha geralmente "coletando" luz refletida de objetos na cena e criando uma imagem em duas dimensões (2D). Em seguida, essa imagem é utilizada para obter informações sobre a cena. Essas informações podem ser usadas em muitas áreas. Há um grande interesse em Visão de Máquina devido sua aplicabilidade em uma grande classe de problemas, tais como:

- Manipulação - capacitar máquinas e rôbos para executarem tarefas que são penosas, repetitivas, que ofereçam risco de vida ao homem, ou mesmo que possam ser feitas

com mais eficiência por máquinas. Essa área envolve trabalhos em linhas de produção de indústrias, remoção de lixo tóxico ou radioativo etc.

- Navegação - dotar robôs de visão artificial para capacitá-los a encontrar caminhos livres, evitar obstáculos e calcular orientação e velocidade adequadas. Isso dá condições ao robô de participar de missões espaciais, ter acesso a minas subterrâneas, plataformas submarinas etc.

- Reconhecimento de Objetos - habilidade usada para diferenciar objetos e relações entre objetos. Essa área abrange tarefas de inspeção visual, sistemas de segurança com reconhecimento ou detecção de faces, íris e impressões digitais, controle de qualidade em indústrias, reconhecimento de dígitos manuscritos, etc.

O processo de visão, do ponto de vista computacional, é normalmente integrado e pode ser dividido em várias etapas: aquisição, pré-processamento, segmentação, descrição, reconhecimento e interpretação. As seções seguintes explicam sucintamente cada uma destas etapas

1.3.1 Aquisição

Etapa em que as informações visuais do meio são convertidas para sinais elétricos através de dispositivos ou sensores ópticos. A qualidade das imagens obtidas é importante para o sucesso das etapas seguintes. Um sensor é algo que pode mudar o estado computacional do sistema em reposta à mudanças ocorridas no estado do ambiente à sua volta. Pode ser simples, como os arranjos de CCD existentes nas camêras contemporâneas, ou complexo, como os sensores log-polar [Spie89] que imitam a estrutura da retina humana.

1.3.2 Pré-processamento

Nessa etapa, as imagens obtidas na fase anterior são preparadas para facilitar os processos seguintes. Caso as imagens apresentem ruídos ou pouca definição de detalhes, são aplicadas técnicas de Processamento Digital de Imagens (PDI) para eliminar esses ruídos, suavizar efeitos indesejados decorrentes de fatores externos, além de realçar detalhes importantes nos

objetos. Essas tarefas são feitas tendo-se em mente o objetivo final do processo e, dependendo desse objetivo, algumas técnicas de PDI são aplicadas e outras não.

1.3.3 Segmentação

É a etapa em que a imagem é dividida em regiões que constituem os diversos objetos que nela são representados. A identificação de um objeto se baseia na detecção de descontinuidades ou similaridades na imagem, gerando uma representação abstrata de seu contorno ou da região que ocupa. Os objetos podem ainda compor-se de outros objetos menores e nessa situação há a necessidade de um processo de modelagem para a identificação dos relacionamentos entre o objeto e seus objetos-componentes. Essa modelagem pode ser de grande ajuda no processo de reconhecimento de objetos.

Na segmentação, a identificação de objetos pode ser orientada por regiões semelhantes ou por contornos que evidenciam descontinuidades. Essas duas estratégias proporcionam formas distintas de representação.

- Representação por Regiões - É a enumeração dos pontos da imagem que constituem o objeto. O mapa de identificação dos objetos é uma matriz com as mesmas dimensões da imagem original, onde a cada ponto é atribuído um rótulo relativo ao objeto a que pertence. As técnicas mais utilizadas para a segmentação por região são o Limiar e o Crescimento por Regiões. Através delas, busca-se atribuir a cada objeto os pontos da imagem que possuam as mesmas propriedades de cor, intensidade e textura, ou seja, que apresentem similaridades segundo um critério pré-estabelecido.

- Representação por Contornos - Um contorno pode ser conceituado como o subconjunto de pontos de um objeto que o separa do restante da imagem. Como representação, constitui uma curva fechada de pontos conectados, a partir da qual se consegue reconstruir a silhueta de todo o objeto. As técnicas de segmentação por contorno buscam detectar descontinuidades que evidenciem limites entre objetos. Métodos de detecção de bordas, busca em Grafos, Programação Dinâmica, Transformada de Hough e acompanhamento de contorno podem ser utilizadas por essa abordagem.

1.3.4 Descrição

Cada objeto identificado na etapa anterior é analisado para a extração de algumas de suas características mais importantes. Esse conjunto de caracteríticas é chamado de padrão e representará o objeto nas próximas etapas. Um padrão pode ser considerado uma abstração definida por uma coleção de possíveis instâncias. Sob um ponto de vista alternativo, um padrão é considerado algo observável, gerado por uma entidade física, como por exemplo instâncias de visões 2D de um objeto 3D [Schölkopf, 1997].

A escolha da forma para representar os objetos é condicionada a fatores relacionados à natureza das imagens tratadas e da tarefa a ser realizada. Em alguns casos, essa etapa pode inclusive ser eliminada. Isso ocorre quando a própria imagem (original ou processada) é utilizada como padrão representativo do objeto. Esse tipo de padrão é chamado de matriz ou retina de pontos e o tipo de reconhecimento pode ser chamado de icônico [Grove and Fisher, 1996]. Se porém, forem extraídas algumas características descritivas do objeto, e este passar a ser representado por esse conjunto, o tipo de padrão é chamado vetor de características, ou apenas padrão. Por apresentar bons resultados com as mais variadas técnicas de aprendizagem de máquina, essa última abordagem é a mais utilizada.

Entretanto, essa estratégia apresenta desvantagens em certos domínios de problemas, onde a visão precisa tratar com o brilho das imagens, fator que não é função apenas da forma, mas também de outras propriedades intrínsecas à cena, como a reflectância e a fatores externos, como iluminação [Nayar et al., 1996]. Um outro problema é a dificuldade de desenvolvimento de sistemas de reconhecimentos em tempo real, pois técnicas para a aquisição automática de formas de objetos ainda estão sendo pesquisadas. Esses fatores motivaram o desenvolvimento do método que não se baseia na representação da forma e sim, da aparência. Esse método utiliza as próprias imagens como padrão, visões do objeto capturadas sob diversos ângulos, para codificarem variações de brilho geradas pela forma 3D, propriedades de reflectância da superfície, parâmetros de sensor e condições de iluminação.

Os modelos conexionistas foram os primeiros a apresentarem resultados satisfatórios no reconhecimento icônico. Recentemente, a técnica SVM também têm sido aplicada a este problema com ótimos resultados [Pontil and Verri, 1998]. As características da representação baseada na aparência (destacadas acima) e as características mais importantes de SVM (descritas na Seção 1.1) motivaram o desenvolvimento desta dissertação de mestrado, cujos

principais objetivos são estudar a técnica SVM e investigar sua aplicação à aprendizagem de objetos visuais, com representação baseada na aparência.

1.3.5 Reconhecimento e Interpretação

Nesta etapa, o padrão de cada objeto identificado é comparado às classes de padrões já conhecidas com o intuito de decidir à qual grupo ele pertence. A formalização do processo decisório é indispensável para sistemas de visão artificial. Essa tarefa pode ser feita através de funções que dividem o espaço de características em regiões, de acordo com o número de classes envolvidas no problema. A forma mais simples de particionar um espaço Euclidiano de n dimensões é através de *hiperplanos*. SVM baseia-se também nessa estratégia, porém, utiliza um tipo especial, o hiperplano de separação ótima. Trata-se de um hiperplano que divide as classes maximizando a margem de separação entre elas. A Figura 1.3 mostra um hiperplano de margem máxima. A linha mais espessa representa o hiperplano e as pontilhadas, as margens.

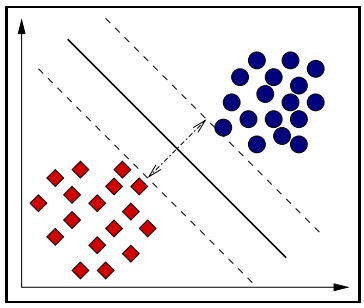

Figura 1.3: Hiperplano de separação ótima para duas classes.

Os diferentes tipos de problemas de Reconhecimento de Padrões fazem suposições sobre a forma de geração dos padrões [Schölkopf, 1997]. Para o reconhecimento baseado na aparência, alguns dos termos relevantes são:

- Visões ou poses - variações no ângulo de percepção dos objetos que são capturadas, ou seja, são imagens do objeto sob pontos de vista diferentes.
- Classes - são conjuntos de objetos apresentando propriedades em comum.

- Objetos - são instâncias concretas de uma classe, com uma estrutura rica, contendo normalmente todas as visões que correspondem a transformações 3-D rígidas de uma entidade física específica.

Um sistema basedo em visões, tal qual o utilizado para o desenvolvimento desta dissertação, considera esses três conceitos. Entretanto, em função das bases de imagens utilizadas nesta dissertação, o conceito de classe pode ser confundido com o de objeto, uma vez que nessas bases uma classe é constituida por várias visões do objeto.

Conforme a natureza do problema tratado, pode ser necessária uma análise de cena, onde os objetos são relacionados entre si, buscando-se uma descrição do ambiente em que se encontram. Um exemplo é a subárea da interpretação e compreensão visual que combina técnicas da IA e sistemas baseados em conhecimento com técnicas de Visão de Máquina para fornecer o conteúdo da cena. No problema de formação de agrupamentos, em aprendizagem não-supervisionada pode haver a necessidade de interpretação da cena para a definição dos grupos, a partir de relacionamentos entre objetos e entre o ambiente e os objetos. Nesta dissertação, em especial, essa etapa não é necessária, uma vez que o objetivo se resume a apenas aprender e reconhecer objetos.

Os métodos utilizados para realizar o Reconhecimento de Padrões podem ser divididos nos seguintes grupos:

- **Métodos lingüísticos ou sintáticos** - Envolvem a construção de uma linguagem formal que representa o conjunto de objetos pertencentes a uma determinada classe, a partir da análise de suas propriedades sintáticas. Esses métodos normalmente aparecem associados a outras metodologias e são indicados para problemas onde a análise da estrutura do objeto é fundamental no processo de reconhecimento.

- **Métodos heurísticos** - Nesses métodos o reconhecimento é realizado através de regras baseadas no conhecimento e experiência prévia de especialistas. Compõem essa categoria: sistemas cognitivos, Árvores de Decisão e Diagramas de Influência.

- **Métodos matemáticos** - As funções de decisão são determinadas matematicamente através de uma abordagem determinística, como em Redes Neurais, ou estatística, como o Classificador de Bayes e SVM.

Nesta dissertação SVM é aplicado ao reconhecimento de objetos a partir de suas matrizes de pixels. Segundo a literatura da área, para uma metodologia de Reconhecimento de Padrões ser eficaz, deve apresentar pelo menos duas características: boa capacidade de generalização, permitindo que padrões próximos sejam classificados da mesma forma, e boa capacidade de discriminação, que assegura a correta separação entre as classes no espaço de características. SVM realiza essas duas tarefas. A primeira através da Minimização do Risco Estrutural e a segunda através da maximização da margem que separa as classes entre si. As próximas seções apresentam mais detalhes sobre o desenvolvimento desta dissertação.

1.4 Objetivos

O objetivo geral desta dissertação é realizar um estudo teórico de SVM e investigar sua aplicação a problemas de Visão de Máquina no contexto de Reconhecimento de Padrões a partir de uma representação dos objetos baseada na aparência. SVM já foi aplicado a uma série de problemas de Visão, como reconhecimento de dígitos manuscritos [Vapnik, 1999], modelagem de percepção através de atenção visual [Eghbalnia and Assadi, 2001], detecção e autenticação de faces [Osuna et al., 1997b] [Jonsson et al., 1999], aprendizagem de modelos de faces a partir de multi-visualizadores [Kwong and Gong, 1999] e também em reconhecimento de objetos basedo na aparência [Pontil and Verri, 1998]. Os resultados mostraram a aplicabilidade da técnica a estes problemas com desempenho equivalente ou até superior à outras abordagens mais clássicas.

O objetivo específico desta dissertação é treinar um classificador SVM para reconhecer um número limitado de classes de objetos, vistos sob vários ângulos diferentes. Trata-se portanto, de aprendizagem de objetos 3-D a partir de imagens 2D, com múltiplas classes, constituídas por múltiplas visões dos objetos e considerando-se variações de orientação e escala das imagens. Os experimentos envolvem a utilização da estratégia de validação cruzada para treinar exaustivamente o sistema e a utilização de visões nos conjuntos de treinamento e de teste, selecionadas de forma aleatória, para tornar o problema o mais próximo da situação real.

A investigação é feita utilizando-se, em um primeiro momento, SVM com kernel polinomial quadrático, com o objetivo de verificar o comportamento da técnica nesse tipo de

problema. Posteriormente investiga-se SVM com três tipos diferentes de kernel polinomial (linear, quadrático e cúbico) para um estudo comparativo entre esses tipos e avaliação do kernel mais indicado ao problema. Finalmente, utiliza-se uma Rede Neural *Multilayer Perceptron* para o desenvolvimento de um estudo comparativo considerando o desempenho de classificação.

Um outro objetivo geral desta dissertação é aumentar o entendimento da técnica, por tratar-se de uma técnica relativamente nova, ainda não existem livros-texto com uma apresentação que seja naturalmente acessível à comunidade de Ciência da Computação. Consequentemente, esta dissertação objetiva dar uma contribuição no sentido de proporcionar uma visão mais didática do assunto, contribuindo para uma maior compreensão da técnica SVM e gerar novos resultados que ratifiquem a aplicabilidade da técnica, em problemas de Visão de Máquina.

1.5 Contribuições e Relevância

Esta dissertação está contribuindo com a compreensão do real grau de abrangência da técnica de SVM em problemas práticos. É importante destacar que as pesquisas com SVM são recentes, como já foi citado anteriormente e estão crescendo muito, porém ainda em pequena escala no Brasil. Portanto, uma outra contribuição desta dissertação é do ponto de vista didático, ao sistematizar a compreensão dessa técnica que envolve conceitos matemáticos e estatísticos de difícil assimilação, tais como: Otimização Matemática, Teoria de Aprendizagem e funções kernel e seus diversos tipos (como polinomial, Funções de Base Radial, etc).

Além disso, os resultados servirão para ratificar e principalmente expandir os resultados obtidos em [Pontil and Verri, 1998], [Roobaert, 1999], [Roobaert et al., 2000] e [Roth et al., 2000]. Também vale ressaltar que a estratégia de treinamento utilizada nesta dissertação executa um treinamento exaustivo, sem intervenção humana no sentido de induzir o sistema à taxas de acerto mais realísticas. A mesma estratégia também é aplicada juntamente com Redes Neurais para um estudo comparativo e para uma avaliação mais completa da técnica.

Esta dissertação também se propõe a ser um estímulo para o desenvolvimento de novos trabalhos envolvendo SVM e reconhecimento de objetos baseado na aparência no âmbito

do programa de Pós-Graduação em Informática desta Universidade. Em um levantamento bibliográfico realizado nas bibliotecas das Coordenações de pós-graduação em Informática (COPIN) e Engenharia Elétrica (COPELE), a maioria dos trabalhos realizados tratam de Reconhecimento de Padrões a partir da extração/identificação de características dos objetos. Abaixo são descritos alguns desses trabalhos.

Em [da Silva, 1999] que, embora tenha sido aplicado em imagens com múltiplas visões, é proposto o desenvolvimento de um método para a identificação de vértices tridimensionais das imagens, com o objetivo de definir um modelo para o processo de reconhecimento de formas.

Em [de Fátima Santos Farias, 1997] é apresentado um método de reconhecimento de objetos em cenas tri-dimensionais baseado em técnica multi-vista, porém a identificação do objeto ocorre através de casamento de atributos geométricos extraídos de objetos bi-dimensionais da cena. Estes atributos são casados com os encontrados em modelos tri-dimensionais dos objetos, previamente construídos e armazenados.

Em [Cortez, 1996] foi desenvolvida uma nova metodologia que utiliza modelos poligonais e seus atributos geométricos para realizar a tarefa de identificar objetos. Essa metodologia é capaz de tratar com objetos que estejam isolados, em contato ou mesmo parcialmente visíveis (parcialmente oclusos), diferentemente do método de reconhecimento basedo na aparência.

Há trabalhos que tratam de segmentação e técnicas de Processamento de Imagens [de Alcântara Morais, 1998] e [de Arruda, 1997], reconhecimento de caracteres [Correia, 2000], reconhecimento de dígitos manuscritos [Veloso, 1998] e detecção e reconhecimento de objetos, como em [Ribeiro, 2001] que trata de objetos que são peças em um tabuleiro de xadrez. Este último, utiliza Redes Neurais para a detecção e reconhecimento das peças. Esta dissertação contribui com a área de reconhecimento baseado na forma dos objetos e aprendizagem de máquina com o método estatístico SVM.

1.6 Organização da Dissertação

Esta dissertação está dividida em cinco capítulos. O Capítulo 2 contém um estudo teórico sobre SVM. Este capítulo representa a contribuição desta dissertação a um melhor entendi-

mento da área. São descritos os principais conceitos e fundamentos de SVM. O primeiro conceito é a Teoria de Aprendizagem Estatística, a qual é apresentada juntamente com os seus pontos importantes, como Minimização do Risco Estrutural e demais termos que determinam a boa capacidade de generalização da técnica. Em seguinda, são descritos os conceitos da Teoria de Otimização Matemática, estratégia usada para realizar o treinamento de SVM. Finalmente, todos os teoremas e problemas relacionados ao treinamento, teste, funções kernel e demais conceitos que envolvem SVM, são apresentados.

O Capítulo 3 apresenta uma revisão bibliográfica. O objetivo é descrever os principais trabalhos desenvolvidos tanto com SVM quanto com reconhecimento de objetos. Os trabalhos são discutidos com o intuito de mostrar em que problemas SVM já foi aplicado, destacando os resultados obtidos e que outras técnicas também foram usadas em problemas de reconhecimento de objetos com múltiplas visões. As seções abrangem os trabalhos teóricos com SVM, as estratégias de implementação da técnica, as extensões propostas, e uma seção final descreve alguns problemas em que SVM foi aplicado, dando ênfase ao problema de reconhecimento de objetos 3D a partir de visões 2D e técnicas utilizadas em trabalhos anteriores.

O Capítulo 4 contém uma descrição de todas as informações obtidas dos experimentos desenvolvidos durante o trabalho prático. É apresentada inicialmente uma avaliação de algumas bibliotecas disponíveis no domínio público que foram testadas para a escolha da mais adequada ao problema. Os experimentos estão divididos em três partes. Na primeira, há a descrição do treinamento e teste usando uma base de imagens pequena e uma *toolbox* para o simulador Matlab. O segundo experimento, compreende a aplicação de SVM, utilizando-se uma biblioteca em C++, numa base de imagens de maior dimensão, para uma avaliação de três tipos diferentes de kernel. O terceiro experimento compreende um estudo comparativo entre SVM, com os kernels que apresentaram melhores resultados na etapa anterior, e Redes Neurais. Todos os resultados obtidos também são apresentados. Finalmente o Capítulo 5 sintetiza os principais resultados e conclusões da dissertação e perspectivas de trabalhos.

Capítulo 2

Support Vector Machines

Support Vector Machines (SVM) são sistemas de aprendizagem de máquina treinados com um algoritmo de Otimização Matemática e que implementam um limite derivado da Teoria de Aprendizagem Estatística. Essa estratégia de aprendizagem desenvolvida por Vladmir Vapnik [Vapnik, 1999] e colaboradores, em poucos anos desde a sua introdução, vem superando outros sistemas em uma variedade de aplicações [Cristianini and Shawe-Taylor, 2000].

SVM destaca-se por pelo menos duas características: possui sólida fundamentação teórica e pode alcançar alto desempenho em aplicações práticas. A teoria de aprendizagem pode identificar precisamente os fatores que devem ser considerados para a aprendizagem ser bem sucedida e construir modelos que são bastante complexos.

O treinamento de SVM envolve a otimização de uma função quadrática convexa, que é um problema de Otimização Matemática. SVM envolve poucos parâmetros livres que precisam ser ajustados pelo usuário e não há uma dependência, pelos menos de uma forma explícita, na dimensão do espaço de entrada do problema, o que sugere que SVM pode ser útil em problemas com um grande número de entradas. Outro destaque é que a arquitetura de SVM não é encontrada por experimentação [Kwok, 1998].

Essa técnica pode ser aplicada ao Reconhecimento de Padrões (estimar funções indicadores), Regressão (estimar funções de valores reais) e Extração de Características. Como o problema de reconhecimento de objetos basedo na aparência, investigado nesta dissertação, encaixa-se na área de Reconhecimento de Padrões, este capítulo descreve a teoria que fundamenta SVM dando ênfase a esta área.

Conforme mencionado no Capítulo 1, o processo decisório em problemas de Reconheci-

mento de Padrões pode ser realizado através de funções que dividem o espaço de características em regiões. A forma mais simples de fazer isso é através de hiperplanos. SVM baseia-se nessa estratégia ao construir um tipo especial de hiperplano, o Hiperplano de Margem Máxima [Schölkopf et al., 1999a], que será descrito na Seção 2.3.1. A Teoria de Aprendizagem Estatística orienta como controlar a capacidade e prevenir super especialização, através do controle das medidas de margem do hiperplano. A Teoria de Otimização provê as técnicas matemáticas necessárias para encontrar o hiperplano otimizando essas medidas.

Portanto, é importante conhecer essas duas teorias: Teoria de Aprendizagem Estatística e Teoria de Otimização Matemática, que são fundamentais ao entendimento de SVM. Essas teorias são descritas nas próximas duas seções. Na Seção 2.3, a técnica SVM é apresentada à luz dessas duas teorias que lhe dão suporte.

2.1 Teoria de Aprendizagem

O objetivo desta seção é descrever o princípio do modelo conceitual para os processos de aprendizagem que são usados na construção de algoritmos ou máquinas de aprendizagem, em especial SVM. Para construir qualquer teoria é necessário o uso de alguns conceitos em torno dos quais a teoria é desenvolvida. É importante usar conceitos que descrevam condições necessárias e suficientes para a consistência do método de aprendizagem. Em linhas gerais, consistência significa a capacidade de um método de aprendizagem convergir para uma solução ótima.

A Teoria de Aprendizagem Estatística no contexto de SVM, objetiva controlar, em termos matemáticos, a habilidade de generalização (capacidade de classificação correta de padrões não treinados) da técnica. Para esse fim, há a necessidade de uma teoria que permita descrever precisamente os fatores que devem ser controlados para garantir bom desempenho de generalização. Existem diversas teorias de aprendizagem que podem ser aplicadas. A técnica SVM é derivada dos princípios básicos descritos na teoria de Vapnik e Chervonenkis (VC). A teoria VC é a mais apropriada para descrever SVM, mas é possível também dar uma interpretação Bayesiana, entre outras [Cristianini and Shawe-Taylor, 2000].

Para uma tarefa de aprendizagem, com uma quantidade de dados de treinamento finita, o melhor desempenho de generalização ocorre quando é atingido um equilíbrio entre a preci-

são alcançada em um conjunto de treinamento particular e a capacidade do sistema, ou seja, a habilidade do sistema aprender qualquer conjunto de treinamento sem erro [Burges, 1998]. Segundo Bugues [Burges, 1998], um sistema com uma capacidade muito alta é comparado a um botânico que, ao ver uma árvore, conclui que não é árvore porque tem quantidade de folhas diferente das árvores já conhecidas por ele. Por outro lado, um sistema com capacidade muito baixa é comparado a um botânico que conclui que se um objeto é verde, é uma árvore.

Há uma grande família de limites que governam a relação entre a capacidade de um algoritmo de aprendizagem e seu desempenho. Esta dissertação considera os limites definidos na teoria de Vapnik [Vapnik, 1999], que são os limites usados em SVM. Em [Haykin, 1999] também pode-se encontrar esses mesmos limites da teoria de aprendizagem, só que aplicados a Redes Neurais. Por tratarem-se de conceitos não triviais, a próxima seção apresenta um modelo de estimação de função, para tentar facilitar a compreensão desses limites.

2.1.1 Modelo de Estimação de Função

No contexto estatístico, o foco de interesse não é a evolução de um vetor peso w, como ocorre em Redes Neurais, e sim, a diferença entre uma função alvo $f(x)$ e a saída real do sistema. A Figura 2.1 mostra um modelo geral de aprendizagem supervisionada a partir de exemplos, possuindo três componentes principais:

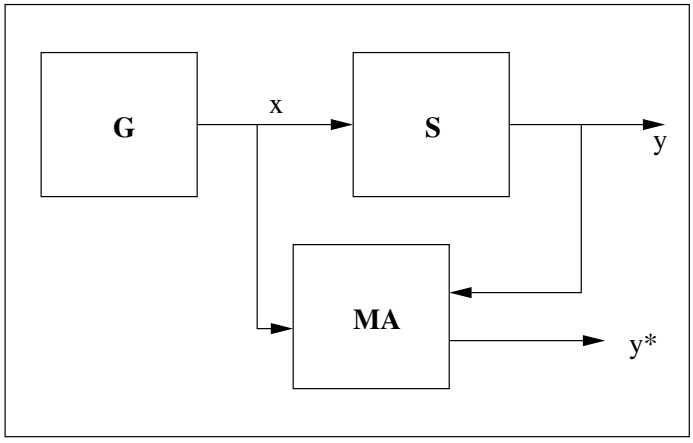

Figura 2.1: Um modelo de aprendizagem a partir de exemplos contendo um gerador G, um supervisor S, uma máquina de apredizagem MA, no qual x é um vetor de entrada randômico, y é a saída do supervisor e y^* é a saída produzida por MA.

1. Um gerador (G), ou ambiente, de vetores randômicos $x \in R_n$, selecionados independentemente a partir de uma função de distribuição de probabilidade cumulativa F(x) fixa, porém desconhecida.

2. Um supervisor (S), ou professor, que retorna um valor de saída desejada y, para cada vetor de entrada x, de acordo com uma função de distribuição condicional F($y|x$), também fixa e desconhecida.

3. Uma máquina de aprendizagem (MA), ou algoritmo, capaz de implementar um conjunto de funções $f(x, \alpha)$, $\alpha \in \Lambda$, em que Λ é um conjunto de parâmetros.

Nesse contexto, o problema de aprendizagem pode ser interpretado como um **problema de aproximação**, que envolve encontrar uma função $f(x, \alpha)$ que gere a melhor aproximação y^* para a saida y do supervisor. A seleção dessa função baseia-se em um conjunto de l exemplos de treinamento independentes e identicamente distribuídos (i.i.d.), gerados de acordo com:

$$F(x, y) = F(x)F(y|x) : (x_1, y_1), \ldots, (x_l, y_l) \tag{2.1}$$

os quais são pares de entrada e saída desejada (x_i, y_i),...,(x_l, y_l), $y_i \in R^n$, $i = 1,\ldots,l$.

Para tentar facilitar a compreensão desses termos, é apresentado o seguinte exemplo extraído de [Burges, 1998]. Em um problema de reconhecimento (detecção) de árvores, x_i pode ser um vetor de valores de pixels (por exemplo, n=256 para uma imagem 16x16) e y_i pode ser 1, se a imagem contém uma árvore e -1, caso contrário. Assume-se que há uma distribuição de probabilidade desconhecida $P(x, y)$, a partir da qual esses dados são selecionados, o que significa que os dados são gerados independentemente e distribuídos identicamente (i.i.d.).

Diante desse problema, há supostamente uma máquina (algoritmo) cuja tarefa é aprender o mapeamento $x_i \mapsto y_i$. A máquina é definida por um conjunto de possíveis mapeamentos $x_i \mapsto f(x, \alpha)$ sendo considerada determinística, ou seja, para uma dada entrada x e parêmetro de ajuste α, sempre a mesma saída $f(x, \alpha)$ será produzida. A escolha de um α particular gera o que podemos chamar de "máquina treinada", por exemplo: uma Rede Neural com arquitetura definida, com α correspondendo aos pesos e limiares, é uma máquina de aprendizagem nesse sentido.

Para alcançar a melhor aproximação entre a saída produzida pela máquina de aprendizagem e a saída desejada, é necessário trabalhar com uma medida chamada risco (erro) mínimo. O próximo tópico apresenta considerações sobre o conceito de risco.

2.1.2 Minimização do Risco

Para escolher a melhor aproximação para a resposta do supervisor, deve-se trabalhar com uma medida de *perda* (ou *discrepância*), definida pela fórmula: $L(y, f(x, \alpha))$, a qual representa uma medida de discrepância entre a resposta desejada y do supervisor, para uma entrada x e a resposta real $f(x, \alpha)$ produzida pela máquina de aprendizagem. O valor esperado da perda é definido pela seguinte fórmula, chamada de funcional de risco:

$$R(\alpha) = \int L(y, f(x, \alpha)) dF(x, y) \tag{2.2}$$

O objetivo é encontrar a função $f(x, \alpha_0)$, dentre a classe de funções $f(x, \alpha)$, $\alpha \in \wedge$, que minimiza a funcional de risco $R(\alpha)$, na situação em que a função de distribuição de probabilidade conjunta $F(x, y)$ é desconhecida e as informações disponíveis encontram-se apenas no conjunto de treinamento (Equação (2.1)).

Essa formulação do problema de aprendizagem pode ser aplicada a muitos problemas específicos como: Reconhecimento de Padrões, Estimação de Regressão e Estimação de Densidade. Em Reconhecimento de Padrões (no caso binário), considerando que a saída do supervisor y assume apenas dois valores $y = \{1, 0\}$, e $f(x, \alpha)$, $(\alpha \in \wedge)$ é um conjunto de funções indicador (funções que têm apenas dois valores: zero e um), a função perda é a seguinte:

$$L(y, f(x, \alpha)) = \begin{cases} 0 & \text{se } y = f(x, \alpha) \\ 1 & \text{se } y \neq f(x, \alpha) \end{cases} \tag{2.3}$$

A funcional de risco determina, para essa função perda, a probabilidade de erro de classificação (respostas diferentes dadas pelo supervisor e pela função indicador). O problema é encontrar uma função que minimize a probabilidade de erro de classificação quando a medida de probabilidade $F(x, y)$ é desconhecida, mas os dados são fornecidos.

A maioria da literatura em Teoria de Aprendizagem Estatística considera uma perda específica para cada problema, como a que foi citada acima para Reconhecimento de Padrões.

O problema de aprendizagem pode ser definido de uma forma mais geral considerando a medida de probabilidade $F(z)$, definida no espaço Z e um conjunto de funções $Q(z, \alpha)$, $\alpha \in \wedge$. O problema como sempre é: minimizar a funcional de risco

$$R(\alpha) = \int Q(z, \alpha) dF(z), \alpha \in \wedge, \tag{2.4}$$

na qual a medida de probabilidade $F(z)$ é desconhecida, porém dados empíricos z_1,. . . ,z_l (amostra i.i.d) são conhecidos, z descreve um par (x, y) e $Q(z, \alpha)$ é a função de perda específica de cada problema (Reconhecimento de Padrões, Regressão, etc). Para aplicar essas definições a um problema particular, deve-se substituir a função de perda correspondente na fórmula.

Entretanto, o risco não pode ser minimizado diretamente, pois depende de uma distribuição de probabilidade desconhecida. Portanto, há a necessidade de estimar uma função que seja próxima da resposta ótima, ou desejada, utilizando-se apenas os dados conhecidos. Em aprendizagem supervisionada, os únicos dados conhecidos são o conjunto de dados de treinamento. Para superar essa dificuldade matemática é usado o princípio indutivo de minimização do risco empírico. Este princípio depende inteiramente do conjunto de dados de treinamento. A próxima seção tratará da definição geral desse princípio.

2.1.3 Minimização do Risco Empírico

O risco empírico é a medida da taxa de erro no conjunto de treinamento para um número de observações finitas e fixas. Para minimizar a Funcional de Risco $R(\alpha)$, definida em termos de uma função de distribuição desconhecida $F(z)$, pode ser aplicado o princípio indutivo da Minimização do Risco Empírico (ERM do inglês "Empirical Risk Minimization"). Esse princípio indutivo pode ser descrito como segue:

1. Substituir a Funcional de Risco $R(\alpha)$ pela Funcional de Risco Empírico

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} Q(z_i, \alpha) \tag{2.5}$$

construída a partir do conjunto de treinamento $z_1, \ldots, z_l$.

2. Aproximar a função $Q(z, \alpha_0)$ que minimiza a Funcional de Risco $R(\alpha)$ pela função $Q(z, \alpha_l)$ minimizando o risco empírico.

Métodos clássicos para a solução de problemas de aprendizagem, como Mínimos Quadrados, PAC e Redes Neurais são instanciações do princípio de ERM para funções de perda específicas. A Funcional de Risco Empírico $R_{emp}(\alpha)$ difere da Funcional de Risco $R(\alpha)$ por dois motivos principais:

1. Não depende, de forma explícita, da função de distribuição desconhecida;

2. Pode ser minimizada com relação aos parâmetros de aprendizagem α (ou vetor peso w no caso específico da máquina de aprendizagem ser uma Rede Neural).

Entretanto, apenas o cálculo ERM não implica em um erro de teste ou risco pequenos, pois um pequeno erro de generalização não pode ser obtido simplesmente minimizando o erro de treinamento [Schölkopf et al., 1999a]. A Teoria VC mostra que é imperativo restringir a classe de funções a partir da qual f é escolhida para que a capacidade seja adequada à quantidade de dados de treinamento disponíveis [Burges, 1998]. O conceito de capacidade mais conhecido da Teoria VC é a dimensão VC, que é componente importante para o limite de generalização de SVM.

2.1.4 Dimensão VC

A dimensão VC é uma medida da capacidade ou poder expressivo da família de funções de classificação realizadas pela máquina de aprendizagem e pode ser definida para várias classes de funções. Grosseiramente, o número de exemplos necessários para se aprender uma classe de interesse de maneira confiável é proporcional à dimensão VC daquela classe. É um conceito puramente combinatorial que não tem conexão com a noção geométrica de dimensão [Haykin, 1999].

Considerando, por exemplo, apenas funções que correspondam ao Reconhecimento de Padrões de duas classes, tal que $f(x, \alpha) \in \{-1, 1\}\ \forall x, \alpha$ (α é usado como um conjunto genérico de parâmetros: uma escolha de α especifica uma função particular). A dimensão VC para a classe de funções $f(\alpha)$, em relação ao espaço de entrada, é definida como o número máximo de pontos de treinamento que podem ser separados em todas as combinações

possíveis, usando funções das classes dadas [Schölkopf et al., 1999a], ou seja, é o número máximo de exemplos de treinamentos que podem ser aprendidos sem erro [Haykin, 1999].

Ainda no problema específico de Reconhecimento de Padrões, uma definição mais geral é: "a dimensão VC de um conjunto de funções indicadores $Q(z, \alpha)$, $\alpha \in \Lambda$, é o número máximo h de vetores $z_i, \ldots, z_h$ (lembrando que z descreve um par (x, y)), que podem ser separados em duas classes, de todas as 2^h formas possíveis, usando funções do conjunto. Se para qualquer n há algum conjunto de n vetores que podem ser separados pelo conjunto $Q(z, \alpha)$, $\alpha \in \Lambda$, então a dimensão VC é infinita"[Vapnik, 1999].

Um exemplo de um limite VC é o seguinte: se $h < l$ é a dimensão VC da classe de funções que a máquina de aprendizagem pode implementar, então, para todas as funções dessa classe, com uma probabilidade de no mínimo $1-\eta$, o limite que aplica-se é [Schölkopf et al., 1999a]:

$$R(\alpha) \leq R_{emp}(\alpha) + \Phi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right) \tag{2.6}$$

em que Φ é chamado termo de confiança, sendo definido pela seguinte fórmula:

$$\Phi\left(\frac{h}{l}, \frac{log(\eta)}{l}\right) = \sqrt{\frac{h(\log\frac{2l}{h}+1) - \log(\frac{\eta}{4})}{l}} \tag{2.7}$$

Segundo Vapnik [Vapnik, 1999], a dimensão VC geralmente não coincide com o número de parâmetros, podendo portanto, ser bem maior ou bem menor, e é a dimensão VC do conjunto de funções, e não o número de parâmetros, a responsável pela habilidade de generalização da máquina de aprendizagem. Esse fator abre a oportunidade de superar a conhecida "*praga da dimensão de entrada*" [1], tornando possível generalizar bem nas bases de um conjunto de funções contendo um elevado número de parâmetros, mas com pequena dimensão VC. Uma das características destacáveis de SVM é o uso de famílias de superfície de decisão com dimensão VC relativamente baixa.

Como foi citado na seção anterior, ERM não determina um erro de teste pequeno. Para superar essa limitação, SVM realiza a Minimização do Risco Estrutural (SRM do inglês

[1] "*Curse of dimensionality*" Uma função definida em um espaço de alta dimensão é muito mais complexa do que uma função definida em um espaço de dimensão menor, tal complexidade torna o processo de classificação mais difícil [Haykin, 1999]

"Structural Risk Minimization"). Esse princípio mostra que o erro de generalização é limitado pela soma do erro do conjunto de treinamento (risco empírico) e um termo que depende da dimensão VC do classificador [Kwok, 1998].

2.1.5 Minimização do Risco Estrutural

O erro de treinamento (risco empírico) é a frequência de erros de uma máquina de aprendizagem durante a etapa de treinamento. O erro de generalização é definido como a frequência de erros produzidos pela máquina quando é testada com exemplos não vistos anteriormente. Nas seções anteriores foi mostrado que a teoria de controle da habilidade de generalização de máquinas de aprendizagem objetiva a construção de um princípio indutivo para a minimização da funiconal de risco $R(\alpha)$.

Considerando-se a probabilidade 1-η e utilizando-se as definições anteriores de risco e do termo de confiança, aplica-se o seguinte limite:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(2l/h)+1) - log(\eta/4)}{l}} \tag{2.8}$$

em que l é o tamanho do conjunto de treinamento e h é a dimensão VC (inteiro não-negativo), medida de capacidade mencionada na seção anterior. O lado direito da inequação é chamado Risco Garantido, embora haja divergências com esse conceito. Em [Burges, 1998], esse termo é chamado *Limite de Risco*, pois segundo o autor, não trata-se de um risco e sim, de um limite no risco e aplica-se com uma certa probabilidade (1-η), logo não é garantido. O Risco Garantido é portanto, o limite para o erro de generalização. O segundo termo no lado direito é chamado de *Confiança VC*.

Segundo Haykin [Haykin, 1999] o erro de treinamento, para um número fixo de exemplos, diminui monotonicamente quando a dimensão VC h é aumentada, enquanto que o intervalo de confiança aumenta monotonicamente. Com isso, o Risco Garantido e o erro de generalização convergem para o mínimo. Observa-se, portanto, que a meta do problema de aprendizagem supervisionada é realizar o melhor desempenho de generalização, combinando a capacidade da máquina com a quantidade de dados do problema. SRM é um princípio indutivo que alcança esse objetivo fazendo a dimensão VC da máquina de aprendizagem uma variável de controle.

Como já foi citado, o intervalo de confiança aumenta à medida que o número de exemplos de treinamento aumenta, podendo tornar-se intratável. SRM supera essa limitação usando a estratégia de introduzir uma "estrutura"a partir da divisão da classe de funções de entrada em subconjuntos aninhados. Então, dado o conjunto S de funções $Q(z, \alpha)$, $\alpha \in \Lambda$, é produzida uma estrutura que consiste de subconjuntos aninhados de funções $S_k = \{Q(z, \alpha), \alpha \in \Lambda_k)$, tal que $S_1 \subset S_2 \subset \cdots \subset S_n \ldots$. A Figura 2.2 ilustra este fato.

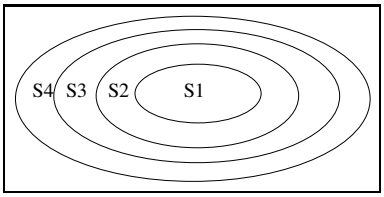

Figura 2.2: Uma estrutura em um conjunto de funções aninhadas.

Os elementos da estrutura devem satisfazer as seguintes propriedades:

1. A dimensão VC h_k de cada conjunto de funções S_k é finita. Então, $h_1 \leq h_2 \cdots \leq h_n$;

2. Qualquer elemento S_k da estrutura deve conter ou um conjunto de funções totalmente limitado $0 \leq Q(z, \alpha) \leq B_k$, $\alpha \in \Lambda_k$, ou um conjunto de funções que satisfaça a inequação:

$$sup_{\alpha \in \Lambda_k} \frac{\int Q^p(z, \alpha) dF(z))^{\frac{1}{p}}}{\int Q(z, \alpha) dF(z)} \leq \tau_k \tag{2.9}$$

   $p > 2$, para algum par (p, τ_k). As estruturas que satisfazem essas condições são chamadas estruturas admissíveis. O supremo de um conjunto não vazio de escalares A, representado por $supA$, é definido como o menor escalar x tal que $x \geq y$ para todo $y \in A$. Similarmente, o ínfimo de um conjunto A, representado por $infA$, é definido como o maior escalar x tal que $x \leq y$ para todo $y \in A$

Dado um conjunto de observações $z_1, \ldots, z_l$, o princípio SRM escolhe a função $Q(z, \alpha_l^k)$, minimizando o risco empírico no subconjunto S_k, que apresente o menor Risco Garantido, buscando assim um compromisso entre a qualidade da aproximação dos dados (erro de treinamento) e a complexidade da função de aproximação (intervalo de confiança). A Figura 2.3 ilustra essa estratégia. O risco empírico diminui com o aumento no índice do elemento da

estrutura, enquanto o intervalo de confiança aumenta. O limite mínimo do risco é alcançado para algum elemento apropriado da estrutura (S^* na Figura 2.3).

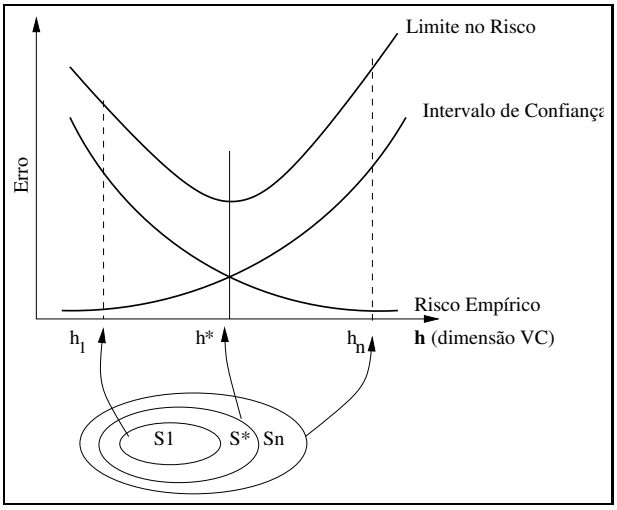

Figura 2.3: O limite de risco é a soma do risco empírico com o intervalo de confiança.

SVM implementa esse princípio estatístico para controlar a capacidade e prevenir super especialização ("*overfitting*"). Porém, essa é apenas uma das teorias que fundamentam a técnica. Para mostrar como SVM é estruturado, antes é importante descrever a outra área, Otimização Matemática, que provê as técnicas matemáticas necessárias para o treinamento de SVM e define algumas das principais características da técnica. As definições de Otimização Matemática são apresentadas a seguir.

2.2 Otimização Matemática

O treinamento de SVM envolve a resolução de um problema de Otimização Matemática [Mattera et al., 1999]. Portanto, para compeender o funcionamento dessa etapa do processo de aprendizagem do algoritmo SVM, é importante um conhecimento prévio da teoria de Otimização Matemática. As próximas seções detalham essa área.

2.2.1 Conceito

Otimização é um ramo da Matemática que envolve a caracterização de soluções para classes de problemas compostos por funções que devem ser escolhidas para minimizar ou maxi-

mizar uma certa função custo, sujeita a certas restrições. Otimização Matemática também se preocupa com o desenvolvimento de algoritmos para encontrar tais soluções. A Teoria de Otimização provê as técnicas algorítmicas e também define condições necessárias e suficientes para um função dada ser uma solução [Cristianini and Shawe-Taylor, 2000]. Os problemas normalmente começam com um conjunto de variáveis ou parâmetros independentes e frequentemente incluem condições ou restrições que definem valores aceitáveis das variáveis. A solução de um problema de otimização é um conjunto dessas variáveis que satisfazem as restrições e minimizam/maximizam a função de custo [Gottfried and Weisman, 1973].

A seguir são apresentadas algumas definições relacionadas às funções e à natureza das restrições. Essas definições são necessárias para uma distinção entre as classes de problemas de otimização, possibilitanto a identificação, no contexto amplo da Teoria de Otimização Matemática, da classe de problemas adequados para a tarefa de treinamento de SVM.

2.2.2 Definições

- **Otimização Irrestrita**: abrange os problemas em que as variáveis podem assumir qualquer valor.

- **Otimização Restrita**: abrange a classe de problemas em que as variáveis podem assumir valores condicionados ou seja, sujeitos à restrições. Essas restrições podem ser de *igualdade* e *desigualdade*. As de igualdade, descrevem frequentemente a operação do sistema que está sendo considerado, enquanto as de desigualdade impõem limites inferior e/ou superior para as variáveis [Lasdon, 1970]. A seguir é mostrado um exemplo de um problema geral de otimização.

  Problema Geral: Dadas as funções f, g_i, $i = 1, \ldots, k$, e h_i, $i = 1, \ldots, m$, definidas em um domínio $\Omega \subseteq R^n$,

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{w}), \quad && \mathbf{w} \in \Omega \\ \text{Sujeito a} \quad & g_i(\mathbf{w}) \leq 0, && i = 1, \ldots, k \\ & h_i(\mathbf{w}) = 0, && i = 1, \ldots, m \end{aligned} \tag{2.10}$$

em que $f(\mathbf{w})$ é chamada *função objetivo* e as demais relações são chamadas, respectivamente, restrição de *desigualdade* e de *igualdade*. O valor ótimo da função objetivo é chamado *valor do problema de otimização*. A região do domínio onde a função objetivo é definida e onde todas as restrições são satisfeitas é chamada *região viável*.

Uma restrição de desigualdade $g_i(\mathbf{w}) \leq 0$ é *ativa* se uma solução $\mathbf{w}^*$ satisfaz $g_i(\mathbf{w}^*) = 0$ e *inativa* se $g_i(\mathbf{w}^*) < 0$. A restrição é dita *satisfeita* se for ativa ou inativa e é dita ser *violada* se $g_i(\mathbf{w}^*) > 0$.

Em algumas situações, quantidades chamadas *variáveis de folga* (ξ) são introduzidas para transformar uma restrição de desigualdade em uma de igualdade. Isso pode ser observado no seguinte exemplo: $g_i(\mathbf{w}) \leq 0 \iff g_i(\mathbf{w}) + \xi_i = 0$, com $\xi_i \geq 0$. Quando essas variáveis são associadas com restrições ativas, são iguais a zero e quando são associadas com restrições inativas indicam a quantidade de "folga"na restrição[Cristianini and Shawe-Taylor, 2000].

- **Programação Linear**: um programa linear é um problema de otimização em que a função objetivo e todas as demais funções de restrição são lineares.

- **Programação Não-Linear**: quando a função objetivo e/ou as funções de restrição são não-lineares.

- **Programação Quadrática**: quando a função objetivo é quadrática e as restrições são lineares. Neste caso, a forma da função objetivo é: $f(x) = k^T \mathrm{w} + \frac{1}{2} x^T Q \mathrm{w}$, para algum n-vetor k, w é a variável desconhecida, T é o expoente e Q uma matriz simétrica.

- **Programação Convexa**: um problema de programação convexa envolve a minimização de uma função convexa sobre um conjunto de restrições. Uma função convexa é garantida estar livre de ótimos locais distintos [Lasdon, 1970]. Um conjunto de pontos é dito ser convexo se, dados quaisquer dois pontos no conjunto, o segmento de reta que os conecta está também no conjunto. O principal teorema da programação convexa é:

  Qualquer mínimo local de um problema de programação convexa é um mínimo global.

- **Programação Quadrática Convexa**: trata de problemas em que a função objetivo é convexa e quadrática. SVM considera problemas em que as restrições são lineares,

a função objetivo é convexa e quadrática e Ω=R^n (domínio dos reais), portanto, são programas quadráticos convexos [Cristianini and Shawe-Taylor, 2000].

Para realizar a procura de máximos e mínimos condicionados, como nos problemas quadráticos convexos e demais problemas de otimização matemática, é utilizado o método de Lagrange. Esse método é usado para solucionar o treinamento de SVM. A próxima seção apresenta o método de Lagrange restrito à programas quadráticos convexos, como em SVM.

2.2.3 Teoria de Lagrange

A teoria de Lagrange objetiva caracterizar a solução de um problema de otimização a princípio quando não há restrições de desigualdade. Este método foi desenvolvido por Lagrange em 1797 generalizando um resultado de Fermat de 1692. Em 1951, Kuhn e Tucker estenderam o método para permitir restrições de desigualdade [Gill et al., 1981]. O Teorema 1 apresenta o caso mais simples, que não contém restrições.

Teorema 1 (Fermat) Uma condição necessária para $\mathbf{w}^*$ ser um mínimo de $f(\mathbf{w})$, $f \in C^1$, é que a derivada da função seja nula nesse ponto.

$$\frac{\partial f(\mathbf{w}^*)}{\partial \mathbf{w}} = 0 \tag{2.11}$$

Uma função f é de classe C^1 se existem e são contínuas $\partial_1 f$ e $\partial_2 f$. A condição apresentada acima, juntamente com a convexidade de f, são condições suficientes para assegurar que um certo valor w^* é o resultado da minimização.

Em problemas restritos, é necessário definir a função de Lagrange, a qual incorpora informações sobre a função objetivo e as restrições. A função de Lagrange é definida (Definição 1), através da soma da função objetivo com uma combinação linear das restrições, em que os coeficientes da combinação são chamados *Multiplicadores de Lagrange* e são determinados no transcorrer da solução.

Definição 1 Dado um problema de otimização com função objetivo $f(w)$ e restrições de igualdade $h_i(w)$= 0,$i = 1, \ldots, m$, a função ou funcional de Lagrange é definida como:

$$L(w, \beta) = f(w) + \sum_{i=1}^{m} \beta_i h_i(w) \tag{2.12}$$

na qual os coeficientes β_i são os Multiplicadores de Lagrange. A partir dessa definição é possível apresentar o teorema de Lagrange.

Teorema 2 (Lagrange) Uma condição necessária para um ponto $\mathbf{w}^*$ ser um mínimo de $f(\mathbf{w})$ sujeito à $h_i(\mathbf{w}) = \mathbf{0}$, $i = 1, \ldots, m$, com $f, h_i \in C^1$, é:

$$\frac{\partial L(\mathbf{w}^*, \beta^*)}{\partial \mathbf{w}} = 0 \tag{2.13}$$

$$\frac{\partial L(\mathbf{w}^*, \beta^*)}{\partial \beta} = 0 \tag{2.14}$$

para qualquer valor β^*. Essas condições acima são também suficientes desde de que $L(w, \beta^*)$ seja uma função convexa de w. Impondo-se essas duas condições e solucionando-se os sistemas obtidos a partir delas, é alcançada a solução global.

Os dois teoremas descritos até o momento, abrem caminho para o caso mais geral, o Teorema Kuhn-Tucker, no qual o problema de otimização contém tanto restrições de igualdade quanto de desigualdade.

Teorema 3 (Kuhn-Tucker) Dado um problema de otimização com domínio convexo $\Omega \subseteq R^n$,

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{w}), \mathbf{w} \in \Omega, \\ \text{sujeito à} \quad & g_i(\mathbf{w}) \leq 0, i = 1, \ldots, k \\ & h_i(\mathbf{w}) = 0, i = 1, \ldots, m \end{aligned} \tag{2.15}$$

com $f \in C^1$ convexa e g_i, h_i afins. É importante lembrar que uma função afim apresenta a seguinte forma $f(x) = kx + p$, na qual k é uma constante que fornece o declive da reta e p é ordenada na origem ou é o ponto de interseção da reta com o eixo das ordenadas. As condições necessárias e suficientes para um ponto $\mathbf{w}^*$ ser o ponto ótimo são a existência de α^*, β^* (um multiplicador de Lagrange para cada restrição) tal que:

$$\frac{\partial L(\mathbf{w}^*,\alpha^*,\beta^*)}{\partial \mathbf{w}} = 0 \quad (2.16)$$

$$\frac{\partial L(\mathbf{w}^*,\alpha^*,\beta^*)}{\partial \beta} = 0 \quad (2.17)$$

$$\alpha_i^* g_i(\mathbf{w}^*) = 0, i = 1,\ldots,k \quad (2.18)$$

$$g_i(\mathbf{w}^*) \leq 0, i = 1,\ldots,k \quad (2.19)$$

$$(\alpha_i^*) \geq 0, i = 1,\ldots,k \quad (2.20)$$

A relação 2.20 é a chamada condição complementar Karush-Kuhn-Tucker (KKT) [Cristianini and Shawe-Taylor, 2000]. Essa relação implica em $\alpha_i^* \geq 0$ para restrições ativas e $\alpha_i^* = 0$ para restrições inativas. Também pode-se observar que um ponto de solução pode estar em uma de duas posições com relação à uma restrição de desigualdade: ou no interior da região viável, com restrições inativas, ou no limite, com restrições ativas. Assim, as condições KKT dizem que, ou uma restrição é ativa, significando $g_i(\mathbf{w}^*) = 0$, ou os multiplicadores de Lagrange correspondentes satisfazem $\alpha^* = 0$. Isto é sintetizado na Equação $g_i(\mathbf{w}^*)\alpha_i^* = 0$.

Além dessas condições, outro conceito fundamental na formulação de SVM é o conceito de dualidade. A próxima seção discorre sobre este conceito, completando assim a fundamentação teórica necessária para a descrição da técnica SVM.

2.2.4 Dualidade

Dado um problema de otimização P, o problema primal, é possível definir um problema relacionado D do mesmo tipo, o problema dual, em que os Multiplicadores de Lagrange de P são parte da solução de D, e os Multiplicadores de Lagrange de D estão contidos na solução de P [Gill et al., 1981]. Se y^* é a solução do problema dual D, a solução do problema primal P pode ser determinada a partir de y^*.

A teoria de Lagrange para problemas convexos permite uma descrição dual. Normalmente essa descrição é mais simples de solucionar computacionalmente do que o problema primal, porque o manuseio direto de restrições de desigualdade é difícil [Cristianini and Shawe-Taylor, 2000]. O problema dual é obtido introduzindo os multiplicadores de Lagrange, que são chamados de variáveis duais. Esse método é baseado na idéia de que as variáveis

duais são as variáveis desconhecidas fundamentais que precisam ser encontradas para solucionar o problema, como no exemplo abaixo .

Definição 2 Dado um problema de otimização com domínio $\mathbf{\Omega} \subseteq R^n$,

$$\begin{aligned} \text{minimize} \quad & f(w), \quad w \in \Omega, \\ \text{sujeito à} \quad & g_i(w) \leq 0, \quad i = 1 \ldots, k \\ & h_i(w) = 0, \quad i = 1 \ldots, m \end{aligned} \tag{2.21}$$

A funcional de Lagrange generalizada pode ser definida como:

$$\begin{aligned} L(w, \alpha, \beta) &= f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{m} \beta_i h_i(w) \\ &= f(w) + \alpha^{'} g(w) + \beta^{'} h(w) \end{aligned} \tag{2.22}$$

O problema dual de Lagrange é:

$$\begin{aligned} \text{Maximize} \quad & \theta(\alpha, \beta) \\ \text{Sujeito a} \quad & \alpha \geq 0 \end{aligned} \tag{2.23}$$

no qual $\theta(\alpha, \beta) = inf_{w \in \Omega} L(w, \alpha, \beta)$. O valor da função objetivo $f(w)$ na solução ótima é chamado *valor do problema*. Solucionando-se os problemas primal e dual, pode-se alcançar a solução do problema geral, simplesmente verificando a diferença entre os valores dos problemas primal e dual. Se essa diferença reduz a zero, o valor ótimo é alcançado. A diferença entre os valores dos problemas primal e dual é conhecida como *intervalo de dualidade*.

Outra forma de detectar um intervalo de dualidade nulo é verificar a presença de um ponto de sela (*saddle point*). Se $f = f(x_1, x_2, \ldots, x_n)$ é uma função escalar de n variáveis, um ponto de sela é qualquer ponto $(x_1, x_2, \ldots, x_n)$ cujo gradiente (1^a derivada) de f no ponto é zero, não sendo porém, ponto de máximo nem ponto de mínimo porque as próximas derivadas de f são de sinais diferentes e em direções diferentes. Um ponto de sela da função de Lagrange para o problema primal é uma tripla (w^*, α^*, β^*), com $w^* \in \Omega$, $\alpha^* \geq 0$, satisfazendo a seguinte propriedade: $L(w^*, \alpha, \beta) \leq L(w^*, \alpha^*, \beta^*) \leq L(w, \alpha^*, \beta^*)$, para todo $w \in \Omega$, $\alpha \geq 0$. Em SVM é considerado o teorema de dualidade forte. Esse teorema garante que os problemas dual e primal têm o mesmo valor.

Teorema de dualidade forte Dado um problema de otimização com domínio convexo

$\Omega \subseteq R^n$ e em que g_i e h_i são funções afins,

$$\begin{aligned} \text{minimize} \quad & f(w), w \in \Omega, \\ \text{sujeito à} \quad & g_i(w) \leq 0, i = 1, \ldots, k \\ & h_i(w) = 0, i = 1, \ldots, m \end{aligned} \tag{2.24}$$

De forma mais específica, a transformação de primal para dual pode ser feita assumindo as derivadas da função de Lagrange iguais a zero com relação às variáveis primais, removendo portanto, a dependência nas variáveis primais. Isso corresponde à computação da função $\theta(\alpha, \beta) = inf_{w \in \Omega} L(w, \alpha, \beta)$. A função resultante contém apenas variáveis duais e pode ser maximizada para restrições mais simples. Essa estratégia é padrão na teoria de SVM.

Alguns elementos importantes da teoria de Otimização Matemática que serão úteis na definição de SVM são listados abaixo:

- o uso de representações duais em SVM permite trabalhar em espaços com dimensões mais altas e prepara o caminho para técnicas algorítmicas derivadas da Teoria de Otimização;
- as condições complementares KKT implicam que apenas as restrições ativas terão variáveis duais não-nulas, significando que, para certas otimizações o número de variáveis envolvidas pode ser bem menor do que o conjunto de treinamento completo;
- o termo vetores de suporte de SVM refere-se aos exemplos que têm variáveis duais não-nulas;
- outras características importantes, como a ausência de mínimos locais, são consequências da utilização de programas quadráticos convexos no treinamento de SVM.

A próxima seção combina as duas teorias aprensentadas até este ponto, Aprendizagem Estatística e Otimização Matemática, para descrever a técnica de aprendizagem SVM.

2.3 Support Vector Machines

SVM pode ser aplicado a problemas de Reconhecimento de Padrões, Regressão, Extração de Características e Detecção de Novidades. Para cada um desses contextos há definições ma-

temáticas específicas. Para não extrapolar do escopo desta dissertação, esta seção descreve SVM apenas no contexto de Reconhecimento de Padrões.

O modelo mais simples de SVM, que também foi o primeiro a ser introduzido, é chamado Classificador de Margem Máxima. Ele trabalha apenas com dados linearmente separáveis, ficando restrito, portanto, à poucas aplicações práticas. Apesar dessa limitação, o Classificador de Margem Máxima apresenta propriedades importantes e é a pedra fundamental para a formulação de SVMs mais sofisticadas.

A Figura 2.4 mostra um espaço de características linearmente separável para um conjunto de treinamento bidimensional e a Figura 2.5 ilustra um espaço linearmente inseparável. A linha escura presente em ambas as figuras, separando os vetores de entrada de classes distintas, é chamada de superfície de decisão (ou separação). Em particular, na Figura 2.4, devido a linearidade da superfície de decisão, a mesma também é conhecida como hiperplano de separação.

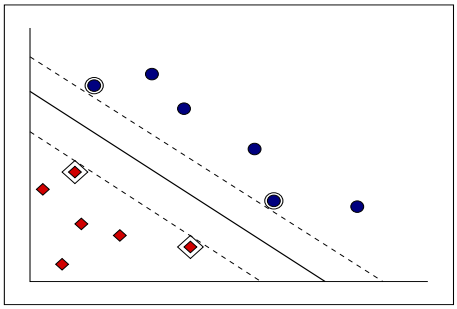

Figura 2.4: Espaço de características linearmente separável.

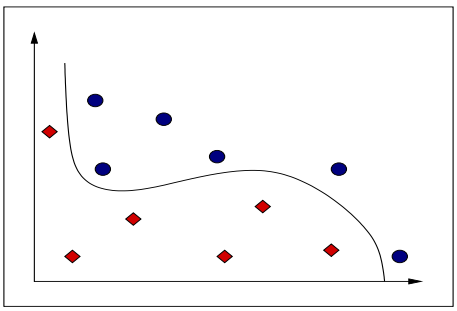

Figura 2.5: Espaço de características linearmente inseparável.

O Classificador de Margem Máxima otimiza limites no erro de generalização das má-

quinas lineares em termos da margem de separação entre as classes a qual é determinada pelo hiperplano de separação. Essa estratégia envolve separar os dados com um tipo especial de hiperplano, o Hiperplano de Margem Máxima ou de separação ótima, que é descrito na próxima seção.

2.3.1 Hiperplano de Margem Máxima

Para introduzir o conceito de Hiperplano de Margem Máxima, inicialmente será considerado o contexto de classificação binária, para conjuntos linearmente separáveis. Após, serão consideradas as demais situações, que são mais comuns em problemas práticos. Com esses conceitos definidos, será apresentado o problema da otimização quadrática convexa de SVM, cuja solução é um Hiperplano de Margem Máxima.

Um hiperplano é considerado de Margem Máxima (ou de Separação Ótima) se separa um conjunto de vetores sem erro e a distância entre os vetores (das classes opostas) mais próximos ao hiperplano é máxima [Vapnik, 1999]. A Figura 2.6 (a) mostra um hiperplano com margem pequena e 2.6 (b), um Hiperplano de Margem Máxima, para um conjunto de treinamento bidimensional. Para o caso linearmente separável, o algoritmo de SVM tem como objetivo encontrar esse hiperplano.

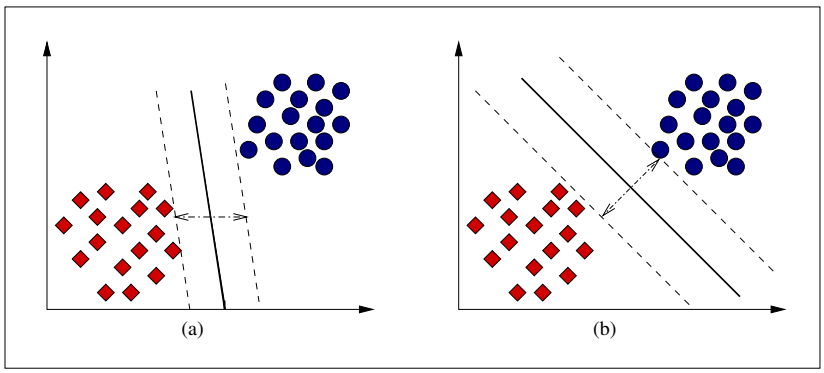

Figura 2.6: (a) Um hiperplano de separação com margem pequena. (b) Um Hiperplano de Margem Máxima.

A Teoria de Aprendizagem permite controlar a capacidade de generalização e a possibilidade de *overfitting* através do controle das medidas de margem do hiperplano. A margem é definida como a distância entre os pontos de dados, de ambas as classes, mais próximos à

superfície de decisão, nesse caso, ao hiperplano. Há diferentes limites de generalização, motivando diferentes algoritmos, tais como: otimizar a margem máxima, a distribuição de margem etc. Esta dissertação considera a mais comum e bem estabelecida estratégia, que reduz o problema à minimização da norma (comprimento) do vetor w [Cristianini and Shawe-Taylor, 2000], também conhecido como vetor peso.

Isso pode ser feito, segundo Burges [Burges, 1998], da seguinte forma. Considera-se um conjunto S de pontos de entrada $x_i \in R^N$ com $i = 1, 2, \ldots, N$. Cada ponto x_i pertence a uma das duas classes, sendo fornecido portanto um rótulo $y_i \in \{-1, 1\}$. Supõe-se que há um hiperplano que separa os exemplos positivos dos negativos. Os pontos x sobre o hiperplano satisfazem $w.x + b = 0$, em que w é normal (perpendicular) ao hiperplano, $|b|/\|w\|$ é a distância perpendicular do hiperplano à origem e $\|w\|$ é a norma Euclidiana de w. Seja d^+ a menor distância entre o hiperplano de separação e os pontos na fronteira da classe positiva $(+1)$, e d^- a menor distância entre o hiperplano de separação e os pontos mais próximos na fronteira da classe negativa (-1). A margem do hiperplano deve ser portanto $d^+ + d^-$.

O algoritmo de SVM procura o hiperplano de separação com margem máxima que pode ser construído como segue. Assume-se que todos os dados de treinamento satisfazem as seguintes restrições:

$$x_i w + b \geq +1 \qquad \text{para } y_i = +1 \tag{2.25}$$

$$x_i w + b \leq -1 \qquad \text{para } y_i = -1 \tag{2.26}$$

Isso pode ser combinado em apenas um conjunto de inequações:

$$y_i(x_i.w + b) - 1 \geq 0 \; \forall i = 1, 2, ..., N \tag{2.27}$$

Os pontos para os quais vale a igualdade na restrição (2.25) estão no hiperplano H_1 : $x_i.w + b = 1$, que possui norma w e distância perpendicular à origem igual a $|1 - b|/\|w\|$. Similarmente, os pontos para os quais vale a igualdade da restrição (2.26) estão no hiperplano $H_2 : x_i.w + b = -1$, com norma novamente w e distância perpendicular à origem igual a $|-1 - b|/\|w\|$. Portanto, $d^+ = d^- = 1/\|w\|$ e a margem é simplesmente $2/\|w\|$.

Pode-se observar que H_1 e H_2 são paralelos (eles têm a mesma norma) e que não há pontos de treinamento entre eles. Assim, é possível encontrar o par de hiperplanos que

geram a margem máxima, pela minimizaçao de $\|w\|^2$, sujeito à restrição definida em 2.27. Essa escolha deve-se a pelo menos dois fatores provenientes da teoria de Aprendizagem Estatística: dimensão VC e princípio SRM.

O primeiro (dimensão VC) refere-se ao fato de que o objetivo da aprendizagem é encontrar w e b tal que a funcional de risco ou risco esperado, Equação (2.4), seja minimizada. Entretanto, como já foi citado na Seção 2.1.2, esse risco não pode ser obtido diretamente uma vez que depende de uma distribuição de probabilidade desconhecida. Diante disso, é minimizado o limite no risco, Equação (2.8), que consiste do risco empírico e do termo de confiança (que depende da dimensão VC). Segundo Vapnik [Vapnik, 1999], a dimensão VC (representada pela variável h) da classe de hiperplanos de separação em relação à margem (ou tamanho do vetor peso w) tem o seguinte limite: $h \leq \|w\|^2 R^2 + 1$, no qual R é o raio da menor esfera envolvendo os dados de treinamento, que é fixo para um certo conjunto de dados. Assim, o termo de confiança pode ser minimizado pela minimização de $\|w\|^2$ [Müller et al., 2001].

O segundo (princípio SRM) deve-se ao fato de que para aplicar o método SRM, descrito na Seção 2.1.5, há a necessidade de se construir um conjunto de hiperplanos de separação variando a dimensão VC de tal forma que o risco empírico e a própria dimensão VC sejam minimizados ao mesmo tempo. Em SVM, uma estrutura é imposta sobre o conjunto de hiperplanos de separação através da restrição da norma Euclidiana do vetor w. A dimensão VC tem o limite apresentado no parágrafo anterior. Ainda na Seção 2.1.5 foi citado que deve-se selecionar a estrutura com menor dimensão VC e erro de treinamento. Essa condição pode ser satisfeita usando o hiperplano ótimo, que tem o vetor peso w ótimo, ou seja, com norma Euclidiana mínima. Portanto, a escolha do hiperplano ótimo como superfície de decisão não é apenas intuitivamente satisfatória mas também realiza totalmente o princípio SRM [Haykin, 1999].

A definição desse hiperplano é a seguinte: dada uma amostra de treinamento linearmente separável representada da seguinte forma: $S = ((x_1, y_1), \ldots, (x_l, y_l))$, o hiperplano $w.x + b = 0$ pode ser encontrado solucionando-se o problema de otimização:

$$\begin{aligned} &\text{minimize}_{w,b} \quad (w.w) \\ &\text{sujeito a} \quad y_i((x_i.w) + b) \geq 1, i = 1, \ldots, l \end{aligned} \tag{2.28}$$

A solução para um caso bidimensional típico deverá ter a forma representada na Figura 2.7. Os pontos para os quais aplica-se a igualdade na Equação (2.27) (isto é, aqueles que estão em um dos hiperplanos H_1 ou H_2) e que, se forem removidos, devem alterar a solução encontrada, são chamados vetores de suporte. Esses pontos são indicados na Figura 2.7 por círculos extras.

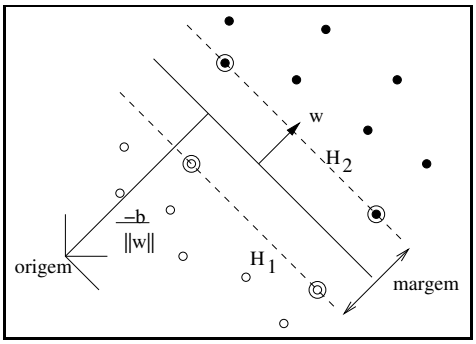

Figura 2.7: Hiperplano de separação para o caso linearmente separável. Os vetores de suporte estão circulados.

Para transformar o problema (2.28) em um problema quadrático conforme descrito na Seção 2.2.2, é necessário re-escrevê-lo da forma abaixo:

$$\begin{aligned} \text{minimize}_{w,b} \quad & \frac{1}{2}(w.w), \\ \text{sujeito a} \quad & y_i((x_i.w)+b) \geq 1, i = 1, \ldots, l \end{aligned} \tag{2.29}$$

Para solucionar esse problema de otimização é usado o método dos multiplicadores de Lagrange, conforme discutido na Seção 2.2.3. Há duas principais razões para isso:

1. A restrição (2.27) é substituída por uma nova restrição, que é definida em função dos multiplicadores de Lagrange, os quais são mais fáceis de manusear computacionalmente;

2. Nessa reformulação do problema, os dados de treinamento apenas aparecem na forma de produto interno entre vetores. Essa é uma propriedade crucial que permite generalizar o procedimento para o caso não linear [Burges, 1998].

São introduzidos os multiplicadores de Lagrange α_i, $i = 1, \ldots, l$. É realizado o produto entre a restrição (2.27) e os multiplicadores de Lagrange positivos e esse produto é subtraído da função objetivo, para formar a funcional de Lagrange, conforme foi descrito na Seção 2.2.3. Portanto, para solucionar o problema (2.29) deve-se encontrar o ponto de sela da seguinte funcional de Lagrange:

$$L_P = \frac{1}{2}(w.w)^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i.w + b) + \sum_{i=1}^{l} \alpha_i \tag{2.30}$$

Esse é um problema de Programação Quadrática Convexa já que a própria função objetivo é convexa e os pontos que satisfazem as restrições também formam um conjunto convexo. Qualquer restrição linear define um conjunto convexo e um conjunto de N restrições lineares simultâneas define a intersecção de N conjuntos convexos, que também é um conjunto convexo [Burges, 1998] .

A derivada de L_P em relação a w e b deve ser nula, isso corresponde ao fato de que no ponto ótimo, têm-se as seguintes equações de ponto de sela:

$$\frac{\partial L_P(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^{l} y_i \alpha_i x_i = 0 \tag{2.31}$$

$$\frac{\partial L_P(w, b, \alpha)}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0 \tag{2.32}$$

substituindo as relações obtidas têm-se:

$$w = \sum_{i=1}^{l} y_i \alpha_i x_i \tag{2.33}$$

$$\sum_{i=1}^{l} y_i \alpha_i = 0 \tag{2.34}$$

Dadas essas restrições acima, pode-se substituí-las na Equação (2.30) e obter a formulação dual, uma vez que é computacionalmente mais eficiente encontrar o ponto de sela na formulação dual [Roobaert, 1999], a qual é definida da seguinte forma:

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \tag{2.35}$$

É importante observar que L_P é o problema primal e L_D o problema dual, como foi descrito na Seção 2.2.4. A solução é encontrada pela minimização de L_P ou maximização de L_D. Há um multiplicador de Lagrange para cada ponto de treinamento. Na solução, os pontos para os quais $\alpha_i > 0$ são chamados **vetores de suporte** e estão em um dos hiperplanos H_1 ou H_2. Para SVM os vetores de suporte são os elementos críticos do conjunto de treinamento. Eles estão na fronteira, ou seja, mais próximos do hiperplano de decisão. Todos os outros pontos tem $\alpha_i = 0$. Se todos esses outros pontos forem removidos e o treinamento for repetido, o mesmo hiperplano deve ser encontrado [Burges, 1998].

Além das restrições apresentadas acima, as condições complemetares Karush-Kuhn-Tucker (KKT), Seção 2.2.3, fornecem informações úteis sobre a estrutura da solução. As condições dizem que, para a solução ótima, α^*, w^* e b^* devem satisfazer:

$$\alpha_i^*[y_i(w_i^*.x_i + b^*) - 1] = 0 \text{ para } i = 1, \ldots, l \tag{2.36}$$

O problema de SVM é convexo. Para problemas convexos, as condições KKT são necessárias e suficientes para α^*, w^* e b^* serem a solução, logo, encontrar a solução para SVM é equivalente a encontrar uma solução para as condições KKT.

Como os chamados vetores de suporte possuem α_i não nulos, eles são os únicos envolvidos na expressão do vetor peso w, portanto, o vetor peso que representa o Hiperplano de Margem Máxima é calculado na forma da combinação linear abaixo:

$$w^* = \sum_{i=1}^{l} y_i \alpha_i^* x_i \tag{2.37}$$

e pode ainda ser re-escrito em função apenas dos vetores de suporte:

$$w^* = \sum_{\text{vetores de suporte}} y_i \alpha_i^* x_i \tag{2.38}$$

Re-escrevendo o problema novamente, agora colocando a expressão para w^* na funcional de Lagrange dual e levando em conta as condições KKT, o problema quadrático convexo de SVM torna-se o seguinte:

$$\begin{aligned} \text{Maximize} \quad & W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i . x_j) \\ \text{Sujeito a} \quad & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & \alpha_i \geq 0, i = 1, \ldots, l \end{aligned} \tag{2.39}$$

Dado $\alpha_0 = (\alpha_1^0, \ldots, \alpha_l^0)$ ser uma solução para esse problema, então a norma do vetor w, que corresponde ao hiperplano ótimo é igual a:

$$\|w\|^2 = 2W(\alpha_0) = \sum_{\text{vetores de suporte}} \alpha_i^0 \alpha_j^0 (x_i . x_j) y_i y_j \tag{2.40}$$

A regra de separação, baseada no hiperplano ótimo é a seguinte função indicador:

$$f(x) = sgn(\sum_{\text{vetores de suporte}} y_i \alpha_i^0 (x_i . x) - b_0) \tag{2.41}$$

na qual x_i são os vetores de suporte, α_i^0 são os Coeficientes de Lagrange correspondentes e b_0 é um limiar (*threshold*) constante

$$b_0 = \frac{1}{2}[(w_0 . x^*(1)) + (w_0 . x^*(-1))] \tag{2.42}$$

em que $x^*(1)$ corresponde à qualquer vetor de suporte pertencente à primeira classe e $x^*(-1)$, um vetor de suporte pertencente à segunda classe.

O Classificador de Margem Máxima, quando aplicado a dados não separáveis linearmente, não encontra a solução desejada. Isso é evidenciado pela função objetivo (dual) que, aplicada a dados não linearmente separáveis, cresce arbitrariamente [Burges, 1998]. O principal problema desse classificador é que ele sempre constrói hipóteses que se baseiam na inexistência de erros de treinamento. Entretanto, para dados com ruídos, que geralmente implica em separação não linear, o mínimo para o risco esperado não pode ser calculado dessa forma, pois pode causar *overfitting*. Essas desvantagens motivaram o desenvolvimento de técnicas que permitem o tratamento de problemas não-linearmente separáveis via SVM.

2.3.2 Problemas Linearmente Inseparáveis

Para tornar o método descrito na seção anterior, capaz de manipular dados não linearmente separáveis, é necessário "relaxar" as restrições do problema. Diferentemente das restrições (2.25) e (2.26) da seção anterior que utilizam critérios rígidos (*hard margin*), a estratégia apresentada nesta seção utiliza um critério mais relaxado, chamado de *soft margin*. Isso pode ser feito introduzindo variáveis de folga Seção 2.2.2 (ξ_i, $i = 1, \ldots, l$) nas restrições, as quais se tornam:

$$x_i.w + b \geq +1 - \xi_i \qquad \text{para } y_i = +1 \tag{2.43}$$

$$x_i.w + b \leq -1 + \xi_i \qquad \text{para } y_i = -1 \tag{2.44}$$

$$\xi_i \geq 0 \qquad \forall i \tag{2.45}$$

Essa estratégia permite tolerar ruídos e *outliers*[2], considera mais pontos de treinamento, além dos que estão na fronteira e permite a ocorrência de erros de classificação. Portanto, $\sum_{i=1}^{N} \xi_i$ é um limite superior para o número de erros de treinamento. Além disso, a fim de poder representar o custo extra para os erros, decorrente da adição das variáveis de folga, há a necessidade de mudar a função objetivo a ser minimizada de $\frac{1}{2}\|w\|^2$ para:

$$\frac{1}{2}\|w\|^2 + C(\sum_{i=1}^{N} \xi_i)^k \tag{2.46}$$

na qual C é um parâmetro a ser escolhido pelo usuário. C é uma constante que atua como uma função de penalidade e prevenindo que *outliers* afetem o hiperplano ótimo [Kwong and Gong, 1999]. $C > 0$ determina a relação entre o erro empírico e o termo de confiança. Um C maior corresponde a assumir uma penalidade maior para os erros. Por se tratar de um problema de programação convexa, o valor de k pode ser qualquer inteiro positivo, em particular, se $k = 1$ ou $k = 2$ então o problema também é de programação quadrática. Por razões computacionais, entretanto, uma escolha típica é $k = 1$. Esse caso corresponde ao menor dos $k > 0$ e tem a vantagem de não ser necessário que ξ_i e seus multiplicadores de Lagrange, apareçam no problema dual. O problema com essa alteração, torna-se:

[2]Pontos muito distantes das classes a que pertencem.

$$\begin{aligned} \text{Maximize} \quad & L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j, \\ \text{Sujeito a} \quad & \sum_{i=1}^{l} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{2.47}$$

A solução é dada novamente por:

$$w = \sum_{\text{vetores de suporte}} \alpha_i y_i x_i \tag{2.48}$$

Assim, a única diferença do caso do hiperplano ótimo é que os α_i têm um limite superior em C. Essa situação é representada na Figura 2.8

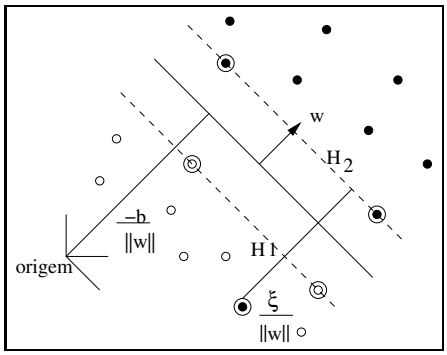

Figura 2.8: Hiperplano de separação para o caso linearmente inseparável.

Para determinar o valor ótimo do limiar (ou *threshold*) b, o procedimento é similar ao descrito anteriormente. Para esse contexto específico, as condições KKT são definidas por:

$$\alpha_i^*[\{y_i(w^*.x_i + b^*) - 1 + \xi_i\}] = 0 \text{ para } i = 1, \ldots, l \tag{2.49}$$

$$\mu_i \xi_i = 0 \text{ para } i = 1, \ldots, l \tag{2.50}$$

A Equação (2.49) é praticamente igual à Equação (2.36), com a diferença que o termo unitário (-1) foi substituído por $(-1 + \xi_i)$. Na Equação (2.50) são introduzidos os multiplicadores de Lagrange μ_i para forçar a positividade de ξ_i.

Com o conteúdo que foi descrito nas seções anteriores é possível a partir deste ponto, descrever formalmente a construção de SVM para uma tarefa de Reconhecimento de Padrões. SVM implementa basicamente duas operaçõs matemáticas:

1. Mapeamento não-linear dos vetores de entrada x em um espaço de características Z com alta dimensão.

2. Construção de um Hiperplano de Margem Máxima no espaço de características.

A Figura 2.9 ilustra essa estratégia.

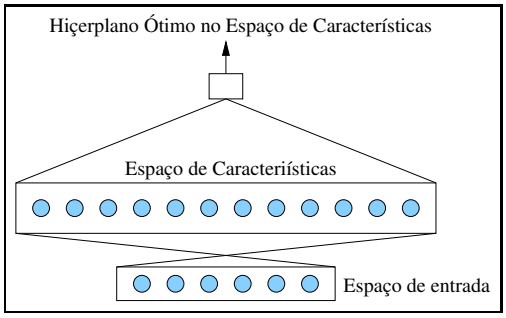

Figura 2.9: Ilustração da estratégia de SVM.

O primeira operação é realizada de acordo com o teorema de Cover, que é descrito em [Haykin, 1999] da seguinte forma:

> "Um problema complexo de classificação de padrões tem mais probabilidade de ser separável linearmente em um espaço de alta dimensão do que em um espaço de baixa dimensão".

Considerando-se um espaço de entrada em que os padrões não são linearmente separáveis, o teorema de Cover diz que esse espaço pode ser transformado em um novo espaço de características, onde os padrões têm alta probabilidade de tornarem-se linearmente separáveis, sob duas condições: a transformação deve ser não-linear e a dimensão do espaço de características deve ser muito alta com relação à dimensão do espaço de entrada. Essas duas condições são satisfeitas na primeira operação descrita acima.

A segunda operação implementa a idéia de construir um Hiperplano de Margem Máxima de acordo com a teoria descrita na Seção 2.3.2, mas com uma diferença fundamental: o Hiperplano de Margem Máxima agora é definido como uma função linear de vetores do

espaço de características e não do espaço de entrada original. A construção desse hiperplano é realizada de acordo com o princípio SRM (Seção 2.1.5), proveniente da teoria da Dimensão VC (Seção 2.1.4).

Em 1992 foi observado por Vapnik [Vapnik, 1999] que para a construção do Hiperplano de Margem Máxima no espaço de características Z não é necessário considerar tal espaço de forma explícita, e sim, apenas calcular os produtos internos entre os vetores de suporte e os vetores do espaço de características. A razão para esse fato é descrita a seguir, conforme apresentado em [Schölkopf, 1997].

Supondo-se que foram fornecidos padrões $x \in R^N$, onde a maior parte das informações está contida nos monômios de grau d das entradas x_j de x:

$$x_{j_1}, \ldots, x_{j_d} \tag{2.51}$$

em que $j_1, \ldots, j_d \in \{1, \ldots, N\}$.

Em tal situação seria interessante extrair primeiro os monômios (ou produtos de posições específicas dentro dos vetores de características) e trabalhar num espaço de características Z representando todos os monômios de grau d. Tomando como exemplo, em R^2 pode-se selecionar todos os monômios de grau 2 através do seguinte mapeamento não linear:

$$\phi : R^2 \longmapsto Z = R^3 \tag{2.52}$$

$$(x_1, x_2) \longmapsto (x_1^2, x_2^2, x_1 x_2) \tag{2.53}$$

Segundo Schölkopf [Schölkopf, 1997], esse método não funciona bem em problemas reais, pois para padrões de entrada N-dimensionais, deve haver

$$N_Z = \frac{(N + d - 1)!}{d!(N - 1)!} \tag{2.54}$$

diferentes monômios do tipo (2.51), formando um espaço de características Z de dimensão N_Z. Por exemplo, para um padrão de entrada contendo 256 posições e monômios de grau $d = 5$, existe um total de 10^{10} possíveis combinações.

Em certas situações, entretanto, há uma forma de computar produtos internos sem realizar explicitamente o mapeamento, através de kernels não-lineares. Em espaços de características polinomiais, por exemplo, isso pode ser feito, conforme está descrito abaixo.

Para computar produtos internos da forma $(\phi(x).\phi(y))$, é aplicada a seguinte representação kernel:

$$k(x, y) = (\phi(x).\phi(y)) \tag{2.55}$$

que permite computar o valor do produto interno em Z sem realizar o mapeamento. Inicialmente, essa afirmação é provada com um exemplo. Considerando-se $N = d = 2$ (os monômios devem ser de grau d), o mapeamento C_2 é:

$$C_2 : (x_1, x_2) \longmapsto (x_1^2, x_2^2, x_1x_2, x_2x_1) \tag{2.56}$$

os produtos internos em Z são da forma:

$$(C_2(x).C_2(y)) = x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 = (x.y)^2 \tag{2.57}$$

nos quais os monômios são distribuídos de uma forma que dá condições para que o kernel k desejado seja simplesmente o quadrado do produto interno no espaço de entrada. Essa estratégia é válida para valores arbitrários de $N, d \in N$, conforme definição abaixo. Uma prova formal pode ser encontrada em [Schölkopf, 1997].

Seja C_d uma função que mapeia $x \in R^N$ para o vetor $C_d(x)$, cujas entradas são todos os possiveis produtos (monônios) ordenados de grau d das entradas x. Então, a correspondente computação de kernel de produto interno dos vetores mapeados por C_d é:

$$k(x, y) = (C_d(x).C_d(y)) = (x.y)^d \tag{2.58}$$

Esse exemplo objetivou mostrar como o produto interno pode ser calculado sem que o mapeamento seja realizado no contexto do kernel polinomial. A próxima seção apresenta de forma mais detalhada as funções kernel que podem ser utilizadas em SVM..

2.3.3 Funções Kernel

As representações Kernel trabalham com a projeção dos dados em um espaço de características com alta dimensão para permitir a classificação em espaços não-linearmente separáveis. Trata-se, em primeira instância, de uma estratégia de pré-processamento que envolve mudar a representação dos dados da seguinte forma:

$$x = (x_1, \ldots, x_n) \longmapsto \phi(x) = (\phi_1(x), \ldots, \phi_N(x)) \tag{2.59}$$

Esse passo é equivalente ao mapeamento do espaço de entrada X em um novo espaço $Z = \{\phi(x)|x \in X\}$ chamado espaço de características em que ϕ_i são as funções kernel. A Figura 2.10 ilustra um mapeamento de um espaço de entrada linearmente inseparável, para um espaço de características de maior dimensão, onde os dados podem ser separados linearmente. É importante observar que ambos os gráficos representam espaços bidimensionais por razões puramente didáticas

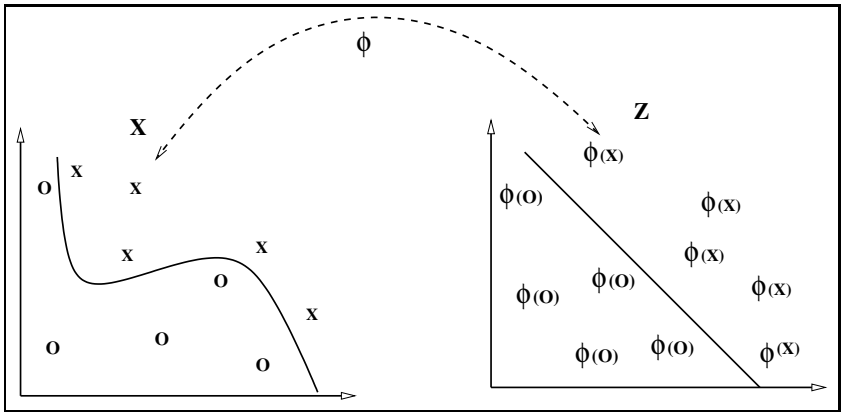

Figura 2.10: Mapeamento do espaço de entrada via função kernel.

Dada essa representação mapeada, uma classificação ou regressão simples em Z pode ser feita. A "praga da dimensão" da Teoria de Aprendizagem Estatísta relaciona-se com uma maior complexidade do problema de estimação de função à medida que a dimensão N do espaço aumenta. Entretanto, a própria Teoria da Aprendizagem Estatística diz que a aprendizagem no espaço de características Z pode ser mais simples, isto é, regras de decisão mais simples são geradas (classificadores lineares). Em outras palavras, embora a dimensão do espaço aumente em Z, a complexidade diminui, porque a classificação, que no espaço de entrada só era possível utilizando superfícies de decisão não lineares, no espaço de características, pode ser feita apenas com um simples hiperplano (superfície de decisão linear).

Outro ponto de destaque das funções Kernel é que o produto escalar pode ser computado implicitamente em Z, sem usar explicitamente ou mesmo sem conhecer o mapeamento ϕ, o que foi destacado na seção anterior. Uma consequência direta disso é que cada algoritmo (linear) que usa apenas produtos escalares pode ser executado implicitamente em Z usando kernels [Müller et al., 2001].

A escolha da função kernel é de vital importância para SVM. Segundo Cristianini e

Shawe-Taylor [Cristianini and Shawe-Taylor, 2000], o problema de escolher uma arquitetura para uma aplicação de Rede Neural é substituido pela escolha do kernel adequado para SVM. Portanto, é importante identificar que propriedades precisam ser satisfeitas para que uma função seja um kernel para algum espaço de características. Inicialmente, para uma função ser considerada kernel deve apresentar duas características:

1. Deve ser simétrica: $K(x,z) = (\phi(x).\phi(z)) = K(z,x)$;

2. Deve satisfazer as inequações de Cauchy-Schwarz:

$$K(x,z)^2 = (\phi(x).\phi(z))^2 \leq \|\phi(x)\|^2\|(z)\|^2$$
$$= (\phi(x).\phi(x))((\phi(z).\phi(z)) = K(x,x)K(z,z).$$

Essas condições, entretanto, não são suficientes para garantir a existência de um espaço de características. As funções kernel devem ainda satisfazer uma terceira propriedade que é fornecida pelo teorema de Mercer. Esse teorema provê uma caracterização de quando uma função $K(x,z)$ é um kernel. A condição de Mercer para garantir que a função simétrica $K(x,z) = \sum_i \phi(x)_i\phi(z)_i$ seja um kernel é que a seguinte intregral:

$$\iint K(x,z)g(x)g(z)dxdz > 0 \tag{2.60}$$

seja válida para todo $g \neq 0$ e

$$\int g^2(x)dx < \infty \tag{2.61}$$

A função kernel não altera muito o problema de SVM, pelo menos não explicitamente. Com a introdução da função kernel, para encontrar os coeficientes α_i é necessário agora resolver o seguinte problema:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i\alpha_j y_i y_j K(x_i,x_j) \\ \text{Sujeito a} \quad & 0 \leq \alpha_i \leq C, i = 1,\ldots,l \\ & \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned} \tag{2.62}$$

Essa funcional coincide com a funcional para encontrar o Hiperplano de Margem Máxima, exceto pela forma do produto interno, que anteriormente era representado unicamente

por produto interno (x_i, x_j) e agora passa a ser representado pela função kernel $K(x_i, x_j)$. Com a função kernel, a função de separação (ou de decisão) passa a ser representada como:

$$f(x) = sign\left(\sum_{\text{vetores de suporte}} y_i \alpha_i k(x_i.x) - b\right) \tag{2.63}$$

O uso de diferentes funções kernel $K(x, x_i)$ possibilita a construção de máquinas de aprendizagem com diferentes tipos de superfícies de decisão não-linear no espaço de entrada. Dentre as funções kernel mais usadas destacam-se: Polinômios, Funções de Base Radial Gaussiana e Rede Neural Sigmóide de duas camadas, definindo diferentes máquinas de aprendizagem conforme discutido abaixo:

1. Máquinas de Aprendizagem Polinomial: para construir regras de decisão polinomiais de grau d utiliza-se o seguinte kernel:

   $$K(x, x_i) = [(x.x_i) + 1]^d \tag{2.64}$$

   Usando essa função kernel (que satisfaz a condição de Mercer) pode-se construir um função de decisão da forma:

   $$f(x, \alpha) = sgn\left(\sum_{\text{vetores de suporte}} y_i \alpha_i [(x_i.x) + 1]^d - b\right) \tag{2.65}$$

   que é uma fatoração de polinômios em espaços de entrada n-dimensionais. A Figura 2.11 (extraída de [Burges, 1998]) mostra um problema de Reconhecimento de Padrões cujo kernel é um polinômio cúbico. Os pontos marcados com círculos extras são os vetores de suporte.

2. Máquinas de Funções de Base Radial (RBF): os kernels (produzindo uma rede do tipo RBF) são da seguinte forma:

   $$K(x, x_i) = e^{-\|x - x_i\|^2 / 2\sigma^2} \tag{2.66}$$

   Esses kernels também satisfazem a condição de Mercer. A Figura 2.12 (extraída de [Schölkopf et al., 1999a]) mostra um exemplo de um classificador SVM, utilizando

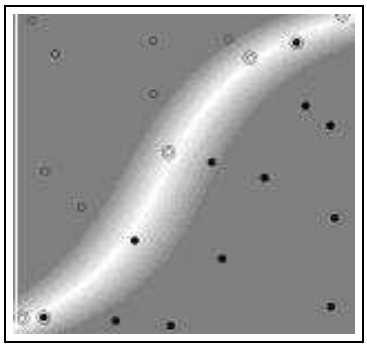

Figura 2.11: Kernel polinomial de grau 3 [Burges, 1998].

um kernel RBF em um problema de classificação binária. Os círculos escuros e claros constituem as duas classes. Novamente os vetores de suporte são marcados com círculos extras.

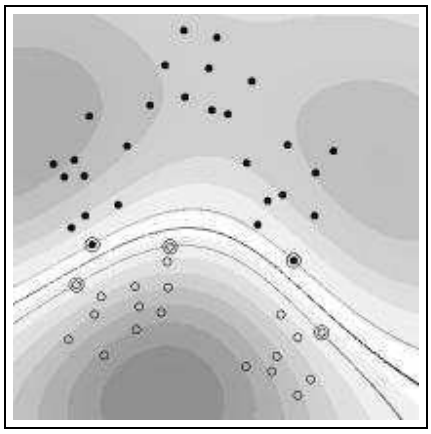

Figura 2.12: Classificador SVM usando kernel RBF [Schölkopf et al., 1999a].

3. Redes Neurais de duas camadas: pode-se definir Redes Neurais de duas camadas escolhendo o seguinte kernel:

$$K(x, x_i) = S[v(x, x_i) + c] \tag{2.67}$$

no qual v e c são parâmetros escalares e $S(u)$ é uma função sigmóide. Ao contrário dos dois tipos de kernel descritos acima, que sempre satisfazem a condição de Mercer, o kernel sigmóide $\tanh(vu + c)$, $|u| \leq 1$, satisfaz a condição de Mercer apenas para alguns valores dos parâmentros v e c. Para esses valores dos parâmetros, pode-se

construir SVMs implementando a regra:

$$f(x, \alpha) = sgn\left(\sum_{i=1}^{N} \alpha_i S(v(x.x_i) + c) + b\right) \tag{2.68}$$

Usando essa técnica, encontra-se automaticamente :

- a arquitetura da máquina de duas camadas, determinando o número N de unidades escondidas (o número de vetores de suporte);
- os vetores dos pesos $w_i = xi$ nos neurônios da primeira camada (escondida), os vetores de suporte;
- o vetor de pesos para a segunda camada (valores de α).

As máquinas de aprendizagem que constróem as funções de decisão descritas nessa seção são máquinas de vetores de suporte. Em SVM a complexidade da construção depende do número de vetores de suporte e não da dimensão do espaço de características [Vapnik, 1999]. A Figura 2.13 mostra a arquitetura de SVM. A entrada x e os vetores de suporte x_i são mapeados não linearmente (por ϕ) em um espaço de características Z, onde os produtos internos são computados. Pelo uso do kernel K, essas duas camadas são computadas, na prática, em um único passo. Os resultados são linearmente combinados pelos pesos v_i, encontrados pela solução de um problema quadrático (em Reconhecimento de Padrões, $v_i = y_i\alpha_i$). A combinação linear é reunida na função σ (em Reconhecimento de Padrões, $\sigma(x) = sgn(x + b)$).

2.4 Conclusão

A teoria que envolve SVM não é de simples entendimento. Nesse capítulo foi possível perceber o nível de complexidade que envolve essa técnica, do ponto de vista teórico. Do ponto de vista prático, a literatura da área afirma que SVM apresenta resultados equivalentes e até superiores em relação à outras técnicas. O próximo capítulo contém uma revisão bibliográfica dos principais trabalhos, tanto teóricos quanto práticos, envolvendo SVM e reconhecimento de objetos baseado na aparência. Em seguida, no Capítulo 4, é apresentada uma descrição dos experimentos práticos da aplicação de SVM ao reconhecimento de objetos baseado na

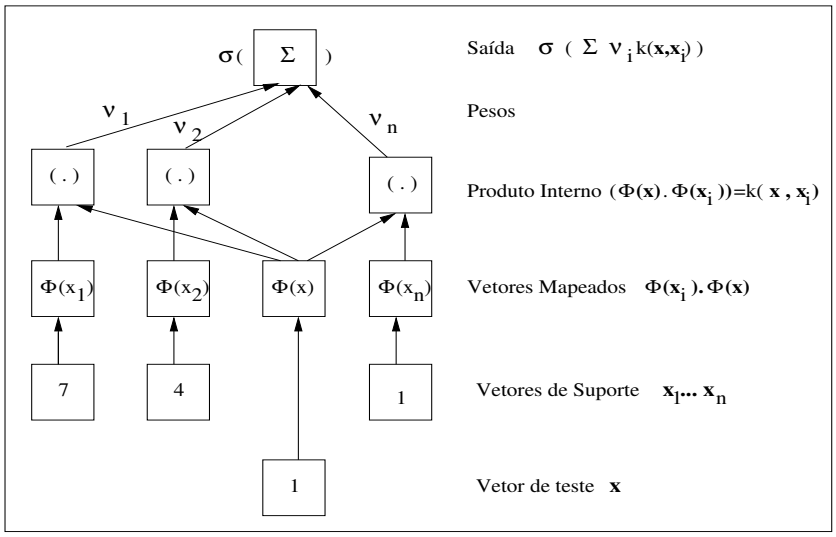

Figura 2.13: Arquitetura de SVM.

aparência. O objetivo é confirmar a alta precisão de SVM e observar o comportamento da técnica sob diferentes condições experimentais..

Capítulo 3

Revisão Bibliográfica

SVM é uma técnica de aprendizagem de máquina oriunda da Estatística. Tornou-se objeto de pesquisa em muitas universidades, laboratórios e empresas, especialmente a partir de 1992. Desde então, o interesse pela técnica tem aumentado consideravelmente. Muitos dos trabalhos têm objetivado investigar o comportamento e o desempenho de SVM em diversas áreas, além de desenvolver mais a técnica através de propostas de extensões teóricas.

O objetivo deste capítulo é apresentar uma revisão bibliográfica sobre SVM, abordando aspectos relativos à evolução conceitual da teoria descrita no capítulo anterior e a problemas em que SVM têm sido aplicado. Este capítulo também apresenta uma breve descrição de algumas metodologias que já foram aplicadas ao problema de reconhecimento de objetos baseado na aparência.

As próximas seções descreverão esses desenvolvimentos. É importante destacar que serão citados apenas os trabalhos mais representativos, com o objetivo de mostrar como esses avanços ocorreram e que aspectos foram considerados. Para facilitar a compreensão, os trabalhos foram subdivididos nos seguinte grupos: teoria, extensões de SVM, implementações, estratégias para reconhecimento envolvendo multiclasses e aplicações.

3.1 Teoria

SVM constitui uma linha de pesquisa importante em aprendizagem de máquina tanto do ponto de vista teórico quanto prático. O recente livro de Vapnik [Vapnik, 1999] descreve o aspecto teórico desenvolvido em SVM: teoria conceitual do processo de aprendizagem,

principio de minimização dos riscos empírico e estrutural, dando ênfase a métodos para estimar a generalização e à aplicação dessas definições em SVM.

Na primeira edição desse livro [Vapnik, 1995], Vapnik dizia que o foco das pesquisas com Redes Neurais alternativas (como Funções de Base Radial) e com Teoria da Aprendizagem, mostravam uma tendência de retorno à base fundamental, como no período da construção das primeiras máquinas de aprendizagem. Entretanto, na segunda edição ele reconhece que as atuais idéias são resultados de análises formais do processo de aprendizagem e não de inspiração biológica, como antigamente.

Embora o livro de Vapnik contenha toda a definição formal de SVM, ele apresenta um conteúdo complexo, especialmente para iniciantes e curiosos na técnica. Essa linguagem extremamente técnica é uma característica presente na maioria das publicações sobre SVM. Tentando deixar o problema mais didático, Burges [Burges, 1998] apresenta um tutorial introdutório de SVM. Esse tutorial descreve as idéias básicas que fundamentam a técnica e apresenta exemplos matemáticos e ilustrações.

No livro de Simon Haykin [Haykin, 1999], a ênfase são as Redes Neurais e sua natureza multidisciplinar. Nesse contexto, SVM é descrito como um dos tipos de máquina de aprendizagem com professor, como uma máquina *feedforward*. No capítulo 2 desse livro é descrito o processo de aprendizagem e suas propriedades estatísticas, destacando conceitos essenciais à SVM como: miminização do risco empírico, do risco estrutural e dimensão VC. O autor dividiu as máquinas de aprendizagem mais conhecidas em dois grupos: com professor e sem professor. As máquinas de aprendizagem com professor descritas são: *Perceptron* de uma camada, *Multilayer Perceptrons* treinadas com o algoritmo *Back-propagation*, Redes de Funções de Base Radial, SVM e Máquinas *Committee*.

Uma coleção com trabalhos dos principais pesquisadores de SVM está reunida em [Schölkopf et al., 1999a]. Essa é a produção impressa de um *Workshop* do estado da arte de SVM que foi realizado durante a Conferência *Neural Information Processing Systems* (NIPS), em dezembro de 1997 na cidade de Breckenridge, Colorado - USA. As idéias centrais de aprendizagem com vetores de suporte são descritas pelos editores, três importantes pesquisadores de SVM (B. Schölkopf, C.J.C. Burges e A.J. Smola).

O livro de Cristianini e Shawe-Taylor [Cristianini and Shawe-Taylor, 2000], pesquisadores da Universidade Royal Holloway de Londres, descreve as máquinas lineares, desde os

Perceptrons de Rosemblatt até as máquinas baseadas em kernel. Os autores destacam que desde 1960, se observava um poder computacional limitado das máquinas de aprendizagem linear, devido especialmente ao fato da maioria dos problemas reais requererem espaços de hipóteses mais expressivos do que simples funções lineares. As representações do tipo kernel constiuíram uma solução alternativa para aumentar o poder computacional das máquinas lineares. Há também uma descrição das duas teorias que formam a base de SVM: Teoria de Aprendizagem Estatística e Otimização Matemática, além de técnicas de implementação e problemas em que SVM foi investigado.

Apesar das diferenças claras entre SVM e Redes Neurais (evidenciadas no capítulo anterior desta dissertação), em [Evgeniou et al., 1999] é proposto um *framework* teórico, com o objetivo de apresentar a Minimização do Risco Estrutural como um conceito que formaliza tanto Redes Neurais quanto SVM, a partir da teoria de Vapnik. Entretanto, em [Vapnik, 1995; Cristianini and Shawe-Taylor, 2000; Haykin, 1999; Stitson et al., 1996; Barabino et al., 1999], SVM é destacada por distinguir-se de técnicas mais tradicionais, especialmente em problemas de Reconhecimento de Padrões, pelo fato de minimizar o risco estrutural e não apenas o risco empírico.

Essas características chamaram a atenção para SVM e mais recentemente para métodos de aprendizagem baseados em kernel. Esses métodos estão se tornando ferramentas muito populares para tarefas de aprendizagem envolvendo classificação, regressão e detecção de novidades [Campbell, 2000]. Nos últimos anos, além de SVM, outros importantes métodos de aprendizagem baseados em kernel foram e estão sendo propostos. Em [Müller et al., 2001], os seguintes métodos são investigados: SVM, Kernel Discriminante de Fisher e Kernel Análise dos Componentes Principais. Este último também pode ser aplicado em aprendizagem não-supervisionada. Outras fontes importantes que tratam da teoria que constitui a base formal de SVM são: [Gunn, 1998; Hearst, 1998] e [Suykens and Vandewalle, 1999].

3.2 Extensões do Algoritmo

As extensões de SVM que foram e estão sendo propostas, são nomalmente compostas por uma base teórica complexa. Como o objeto de pesquisa desta dissertação não são as exten-

sões, elas serão apresentadas de forma genérica, uma vez que uma descrição mais detalhada fugiria do escopo deste trabalho.

3.2.1 ν-SVM

Trata-se de uma modificação proposta por Schölkopf [Schölkopf, 1997] que utiliza um parâmetro ν para possibilitar um controle efetivo no número de vetores de suporte. Foi desenvolvida originalmente para problemas de regressão [Schölkopf, 1997], porém, em seguida, foi extendida para classificação. Em problemas de Reconhecimento de Padrões, o ν-SVM possibilita eliminar a constante de regularização C (Capítulo 2, Seção 2.3.2). Segundo Müller et al [Müller et al., 2001] esse novo parâmetro ν é mais claro que o C. Ele pode ser interpretado como um limite superior para o número de erros de margem e/ou um limite inferior para o número de vetores de suporte. Entretanto, segundo Chang e Lin [Chang and Lin, 2001b], comparando-se com SVM regular, a formulação de ν-SVM é mais complexa e não há métodos efetivos para solucioná-lo para muitos dados de treinamento. Em [Chang and Lin, 2001b] há uma modificação em ν-SVM que impõe um limite nas restrições do programa quadrático e utiliza apenas uma restrição de igualdade. Apresenta ainda um método de decomposição para ν-SVM similar ao de Joachims [Joachims, 1999] para SVM regular.

3.2.2 *Leave-one-out SVM*

Proposto por Weston [Weston, 1999], é baseado na generalização de margens que podem adaptar-se de forma iterativa. Segundo o autor, SVM é restrito a uma margem fixa (valor constante) em cada padrão de treinamento. As margens adaptativas poderiam ser usadas com sucesso, pois essas margens podem controlar a complexidade do modelo de maneira efetiva. O procedimento em *Leave-one-out SVM* é remover um elemento do conjunto de treinamento, construir a função de decisão apenas nas bases dos dados de treimento que ficaram, e testar o elemento removido, ou seja, são testados todos os *l* elementos do conjunto de treinamento, usando diferentes regras de decisão. O erro é limitado pela razão entre o número de vetores de suporte e o número de exemplos de treinamento *l*.

3.2.3 *One Class SVM*

É um método que usa uma estratégia de classificação chamada *one-class*. Ele treina um dado não rotulado e tenta determinar se um ponto de teste pertence à distribuição de dados de treinamento. É um problema de aprendizagem não supervisionada que está sendo investigado no contexto de SVM, em trabalhos como [Schölkopf et al., 1999b].

3.2.4 SVM Multi-Kernel

Outro método proposto por Weston [Weston, 1999]. Ele incorporou ao SVM padrão uma estratégia de decomposição para utilizar múltiplos kernels. Esse método procura eliminar duas restrições impostas pelo método SVM padrão: 1) o conjunto de funções que podem ser implementadas por SVM têm que pertencer a uma classe particular de funções (Funções de Base Radial, Polinômio, etc); 2) a maioria das classes de funções (definidas por kernels) têm pelo menos um parâmetro, a ser definido pelo usuário, que controla algumas propriedades da função gerada (grau do polinômio, por exemplo). Weston propõe o uso de dicionários de kernel, cujo principal objetivo é, a partir de uma descrição do domínio do problema a ser resolvido e suas restrições, permitir o acesso direto a um conjunto de parâmetros mais otimizado para cada função de kernel utilizada.

Um abordagem um pouco diferente é proposta em [Kwok, 1998]. O objetivo é incorporar SVM em uma mistura hierárquica de especialistas (*SVM Mixture*), que leva a problemas de Programação Quadrática similares aos de SVM padrão, com a vantagem de permitir o uso de diferentes especialistas em diferentes regiões do espaço de entrada e também dar suporte a combinações simples de diferentes arquiteturas.

3.2.5 Kernel Análise dos Componentes Principais

PCA (*Principal Component Analysis*) é uma técnica para a extração de estruturas de conjuntos de dados com dimensões elevadas, através da solução de um problema de autovalores e autovetores (*eigenvalues* e *eigenvectors*) [Schölkop et al., 1999]. É uma transformação ortogonal do sistema de coordenadas em que os dados são descritos. Os novos valores das coordenadas que representam os dados são chamados componentes principais. Em grande parte dos problemas, são necessários poucos componentes principais para representar a

maioria das estruturas presentes nos dados.

Na tese de doutorado de Schölkopf [Schölkopf, 1997] há uma aplicação de PCA em um contexto onde os componentes principais do espaço de entrada não são o foco de interesse e sim, componentes principais de variáveis, ou características, que são relacionadas de forma não linear às variáveis de entrada. Para obter esse resultado é realizada a computação de produtos internos através de funções kernel no espaço de entrada, como em SVM.

O algoritmo PCA padrão foi alterado de uma maneira que use exclusivamente produtos internos. Com isso, é possível produzir um algoritmo PCA não linear e baseado em kernel. Esse método não é propriamente uma extensão de SVM e sim, a definição de um tipo de kernel.

3.2.6 Outros

Há ainda outras modificações do algoritmo base de SVM. Em [Osuna and Girosi, 1999] é proposta uma mudança no problema de treinamento para aumentar o desempenho da técnica. Como foi descrito no capítulo anterior, o problema de treinamento original é solucionado a partir de sua formulação dual, para possibilitar que a função kernel seja usada sem que haja a necessidade de realizar o mapeamento explicitamente. Osuna apresenta uma reformulação no problema de treinamento (primal) de SVM, que tem a mesma estrutura da formulação primal original, porém, incorpora o mapeamento implícito a partir da função kernel.

DirectSVM, proposto em [Roobaert, 2000], é uma implementação de SVM para Reconhecimento de Padrões que não é baseada na solução de um problema de Programação Quadrática e sim, em um algoritmo com poucas heurísticas. O objetivo é combinar em um algoritmo, a fundamentação teórica de SVM e a formulação simples de Redes Neurais.

Outro método, Máquina de Vetor de Relevância (*Relevance Vector Machines*-RVM) apresentado em [Tipping, 2000], dá um tratamento baysiano a um modelo linear generalizado que funciona de forma idêntica à SVM.

3.3 Implementações

O treinamento de SVM pode ser reduzido à solução de um problema de otimização quadrático convexo sujeito à restrições lineares (Capítulo 2, Seção 2.3.1). Esse tipo de problema

têm sido amplamente estudado, especialmente no caso convexo. A maioria dos métodos padrão podem ser aplicados diretamente no treinamento de SVM [Cristianini and Shawe-Taylor, 2000]. Várias técnicas podem ser utilizadas para realizar a Programação Quadrática, dentre elas: Quasi-Newton, Gradiente Conjugado e Ponto Interior Dual-Primal [Gill et al., 1981]. Há variados pacotes de software para Programação Quadrática gratuitos e comerciais. Alguns exemplos de pacotes gratuitos são: MINOS [Murtagh and Saunders, 1993], do Laboratório de Otimização de Stanford que usa uma estratégia híbrida e LOQO [Vanderbei, 1997], que usa um método Ponto-Interior Primal Dual.

Entretanto, apesar dessa facilidade, há um consenso entre os pesquisadores de que esses métodos de programação matemática não são adequados para grandes bases de dados. Alguns desses métodos armazenam a matriz kernel diretamente na memória, implicando em uma complexidade quadrática do espaço de decisão em relação ao tamanho da amostra. Usando esses códigos padrão, a aplicação de SVM a um problema com pouco mais de cem variáveis, produziria resultados que poderiam consumir muito tempo de treinamento e quantidade de memória mais do que necessária. Portanto, as rotinas de Programação Quadrática para pequenas bases de dados são a melhor escolha, mas, para grandes bases de dados, há a necessidade de usar técnicas alternativas. A estrutura do problema de otimização de SVM permite derivar algoritmos que possibilitam uma convergência de forma mais rápida e com menor necessidade de memória [Müller et al., 2001]. A seguir serão descritos brevemente os três métodos mais conhecidos e usados.

3.3.1 *Chunking*

É um método para solucionar o problema quadrático de SVM. Começa com a escolha de um subconjunto arbitrário (*chunk*) dos dados. O treinamento de SVM é realizado através de uma rotina de Programação Quadrática nesse subconjunto. Em seguida, o algoritmo guarda os vetores de suporte do *chunk* e descarta os outros pontos. Para o próximo passo, ele adiciona aos vetores de suporte, extraídos na etapa anterior, os M pontos (onde M é um parâmetro do sistema) que mais violam as condições KKT (Capítulo 2, Seção 2.2.3) para formar um novo *chunk*. Essas condições KKT impõem restrições ao problema convexo quadrático com o intuito de facilitar a identificação dos vetores de suporte. As iterações são realizadas até a margem ser maximizada, ou seja, o conjunto *chunk* cresce ou diminui, até que na última

iteração a máquina tenha treinado todo o conjunto de vetores de suporte. Segundo Platt [Platt, 1999] o método *Chunking* reduz o tamanho do conjunto de treinamento à aproximadamente o número de vetores de suporte, que são apenas os pontos que ficam na extremidade da separação entre as classes. Porém, segundo Cristianini e Shawe-Taylor [Cristianini and Shawe-Taylor, 2000], há problemas em que o tamanho do conjunto de vetores de suporte é grande demais para ser tratado por rotinas de otimização, o que leva à falhas na estratégia *Chunking*.

3.3.2 Decomposição

Método similar ao *Chunking* uma vez que também soluciona uma sequência de pequenos Programas Quadráticos, porém, neste método, o tamanho dos sub-problemas é fixo. Eles são baseados nas pesquisas de Osuna et al [Osuna et al., 1997a]. Eles observaram que uma sequência de programas quadráticos, que contém pelo menos uma amostra que viole as condições KKT, convergirá eventualmente para a solução ótima. O algoritmo de decomposição atualiza apenas um subconjunto (conjunto de trabalho) de tamanho fixo, adicionando ou removendo uma amostra em cada iteração, enquanto os outros são mantidos constantes. O objetivo é otimizar o problema global, atuando apenas em um pequeno subconjunto de dados a cada instante de tempo. Porém, a convergência desse método é, na prática, muito lenta e, como no esquema anterior, ainda é necessário utilizar um otimizador de Programa Quadrático convencional [Müller et al., 2001]. Em [Osuna et al., 1997a] não há provas teóricas da convergência desse método [Cristianini and Shawe-Taylor, 2000]. Essas provas são o enfoque em [Chang et al., 2000]. Eles apresentam um algoritmo mais geral que tenta provar a convergência de métodos de decomposição genéricos. Também há alguns resultados numéricos em [Joachims, 1999], que usa o método com uma nova técnica para a seleção do conjunto de trabalho. A biblioteca SVM^{light}, http://www-ai.cs.uni-dortmund.de/SOFTWARE/SVM-LIGHT/svm-light.eng.html, implementa esse método.

3.3.3 Otimização Sequencial Mínina

O método SMO (*Sequential Minimal Optimization*), proposto por Platt [Platt, 1999], baseia-se na idéia do método de decomposição, no caso extremo de solucionar o menor problema

de otimização possivel a cada passo, ou seja, subconjuntos de apenas dois pontos em cada iteração. O poder da técnica consiste no fato do problema de otimização para dois pontos admitir uma solução analítica, eliminando a necessidade do uso de um otimizador de Programa Quadrático como parte do algoritmo [Cristianini and Shawe-Taylor, 2000]. Segundo Müller et al [Müller et al., 2001], o principal problema com o método SMO é a escolha de um bom par de pontos para otimizar em cada iteração. A escolha dos dois pontos é determinada por uma heurística. Apesar de haver mais sub-problemas de otimização para solucionar durante a execução do algoritmo, eles são rápidos e o problema de otimização geral também pode ser solucionado de forma rápida [Platt, 1999]. O SMO original foi desenvolvido para problemas de classificação, porém já foi extendido para regressão e estimação de densidades.

A Figura 3.1 (extraída de [Platt, 1999]), ilustra hipoteticamente os três métodos para o treinamento de SVM citados até aqui. São ilustrados três passos para cada método. As linhas horizontais representam o conjunto de treinamento em cada etapa, enquanto que, as caixas representam os multiplicadores de Lagrange otimizados em cada iteração. O método SMO escolhe dois multiplicadores de Lagrange para otimizar a cada passo, encontra os valores ótimos para esses multiplicadores e atualiza o classificador para refletir os novos valores.

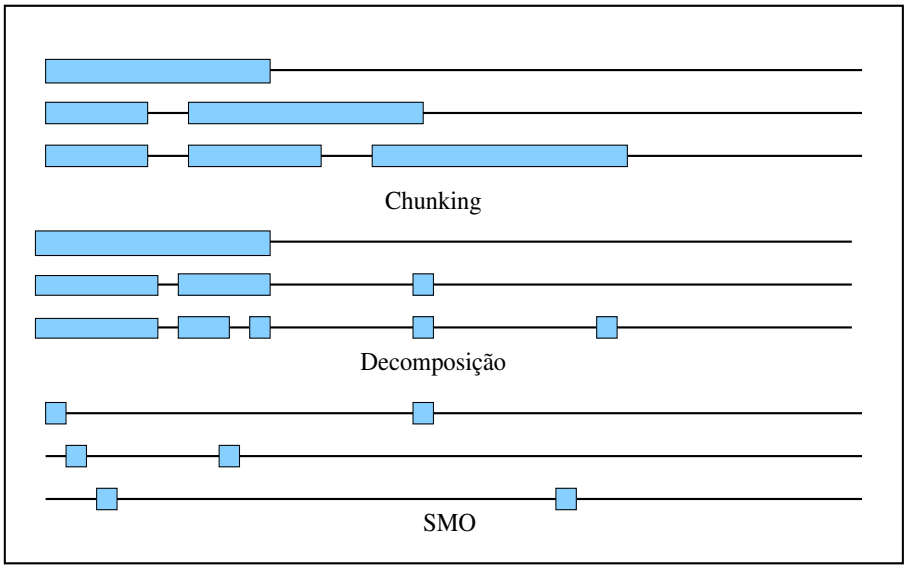

Figura 3.1: Três métodos alternativos para o treinamento de SVM: *Chunking*, Decomposição e SMO.

3.3.4 Outros métodos

Além das estratégias citadas acima, outros algoritmos de menor destaque foram encontrados na literatura. Em [Z. Li and Licheng, 2000] é proposta uma estratégia para pré-extrair os vetores de suporte a partir de conhecimentos a priori fundamentados em uma medida de distância específica. Em [Hadzic and Kecman, 2000], os autores propõem um método para treinar SVM não com Programação Quadrática e sim, com Programação Linear. Em [Yang and Ahuja, 2000] há a proposta de um método geométrico para extrair superconjuntos de vetores de suporte baseado na informação estrutural dos vetores de treinamento.

3.4 Problemas Multiclasses

A solução para problemas de classificação binária usando SVM, está bem desenvolvida, entretanto, os problemas com múltiplas classes são solucionados utilizando-se algumas estratégias. A maioria faz uma combinação de classificadores binários produzidos independentemente [Weston and Watkins, 1999]. A seguir serão descritas três estratégias mais conhecidas:

1. Um-versus-resto (*"one-against-rest"*). Neste método, N diferentes classificadores são construidos, um para cada classe. O $l-th$ classificador é treinado sobre todo o conjunto de dados de treinamento para classificar os membros da classe l contra o restante. Para isso, os exemplos de treinamento devem ser rotulados novamente. Os membros da $l-th$ classe são chamados de 1 e os membros das outras classes de -1. Na fase de teste, o classificador com saída máxima define o rótulo da classe estimada do vetor de entrada corrente.

2. Um-versus-um (*"one-against-one"*). Para cada par de classes possíveis, um classificador binário é calculado. Cada classificador é treinado usando parte do conjunto de treinamento contendo apenas exemplos das duas classes envolvidas. Tal como na estratégia anterior, os conjuntos de treinamento devem ser rotulados novamente. Um total de N(N-1)/2 classificadores são construídos e combinados através de um esquema de votação para verificar o que mais se aproximou do dado de teste e estimar a classificação final.

3. Hierarquias ou árvore de classificadores SVM. Neste método, o problema de classificação é decomposto em uma série de subproblemas de classificação binária, organizados em um esquema hierárquico. A Figura 3.2 ilustra essa estratégia graficamente. Esse método é detalhado em [Schwenker, 2000].

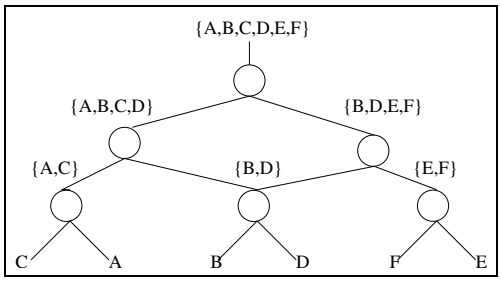

Figura 3.2: Classificador hierárquico geral. A,B,C,D,F,E representam as classes definidas no esquema hierárquico, a partir dos conjuntos {A,C},...,{E,F}, que por sua vez, foram produzidos a apartir do conjunto {A,B,C,D,E,F}.

Há ainda uma proposta de SVM multi-classes [Weston and Watkins, 1999] que é uma extensão natural de SVM binário. Esse método soluciona o problema de classificação de N-classes diretamente. Não é necessário rotular novamente os dados de treinamento. Todas as N-classes são consideradas de uma só vez e as condições de separação são integradas em um único problema de otimização.

Dentre essas estratégias a mais utilizada é *"one-against-one"*. Segundo Chang e Lin [Chang and Lin, 2001a], técnicas como de Weston [Weston and Watkins, 1999] apresentam desempenho inferior ao *"one-against-one"* e embora, muitos classificadores sejam treinados, N(N-1)/2, cada classificador recebe menos dados de treinamento (dados de duas classes apenas) e o tempo de treinamento total usualmente é inferior ao tempo dispendido pelo método "*one-against-rest*".

3.5 Aplicações

Desde meados da década de 90, o método SVM têm sido aplicado a uma série de classes de problemas de Reconhecimento de Padrões, Regressão, Estimação de Densidade, Detecção de Novidades e outros. Abaixo serão listados alguns destes problemas.

3.5.1 Categorização de Texto

Trata-se da classificação de documentos de texto (ou hipertexto) em um número fixo de categorias pré-definidas, baseada no contéudo dos documentos. É um componente importante em muitas tarefas de organização e gerenciamento de informações. Devido ao volume crescente de informações eletrônicas, há muito interesse no desenvolvimento de ferramentas que ajudem as pessoas na filtragem de e-mails, buscas na Web, automação de escritório etc. Esse problema pode ser visto como uma série de problemas de classificação binária, um para cada categoria [Cristianini and Shawe-Taylor, 2000].

Em [Dumais et al., 1998] foi aplicado SVM linear simples na coleção de notícias com rótulos da empresa *Reuters* Ltda, disponível em http://www.research.att.com/ lewis/reuters21578.html. Foram usadas 12.902 notícias da *Reuters*, classificadas em 118 categorias, 75% (9.603) para construir os classificadores e 25% (3.299) para serem classificados. Eles utilizaram o método SMO de Platt. SVM alcançou melhor desempenho em comparação com outras técnicas padrão como Redes Baysianas e Árvores de Decisão. A taxa de acerto média foi de 91,3% para as 10 categorias mais frequentes e 85,5% para todas as 118 categorias.

O trabalho de Joachims [Joachims, 1998] também utilizou a coleção *Reuters*, mas não com SVM linear e sim com os kernels polinomial e Função de Base Radial. A divisão entre os dados de treinamento e de teste foi a mesma de [Dumais et al., 1998]. Ele também desenvolveu experimentos (usando o programa SVM^{light}) na base de dados *Oshumed* (Oregon Health Sciences University) que contém 50.216 documentos com resumos, os 10.000 primeiros foram usados para treinamento e os 10.000 seguintes para teste. A tarefa foi classificar cada documento em uma ou mais categorias de doenças, de um total de 23 possíveis. Independente do kernel usado, SVM alcançou desempenho superior aos métodos padrão.

3.5.2 Reconhecimento de Dígitos Manuscritos

É uma aplicação que surgiu originalmente da necessidade do serviço postal americano automatizar a classificação de correspondências, usando os códigos de endereçamento postal escritos a mão. Segundo Cristianini e Shawe-Taylor [Cristianini and Shawe-Taylor, 2000], este teria sido o primeiro problema real em que SVM foi aplicado.

Em [Vapnik, 1999] foram realizados experimentos na base de dados USPS (*United States Postal Service*), disponível para domínio público, que contém 7.300 padrões de treinamento e 2.000 padrões de teste. SVM foi aplicado com três tipos de kernel: polinomial de grau 3, Função de Base Radial e Rede Neural com duas camadas. Todos eles apresentaram aproximadamente os mesmos resultados. Esse desempenho superou outros métodos como Árvores de Decisão e Redes Neurais clássicas.

3.5.3 Bioinformática

As pesquisas em bioinformática buscam interfaces entre as ciências da vida e da computação. Há muitos problemas que são investigados pela bioinformática, tais como: predição de características estruturais e funcionais de proteínas, categorização automática de expressão de dados de gene de microvetores de DNA, reconhecimento de padrões de estruturas celulares, sequências de DNA etc. A seguir serão descritas de forma sucinta, a aplicação de SVM nos dois primeiros problemas citados acima.

A predição de características estruturais e funcionais de uma proteína pode ser realizada com base em sua sequência de aminoácidos. Uma proteína é formada por uma sequência de aminoácidos (de um conjunto de 20 possíveis), havendo milhares de proteínas diferentes. Uma das estratégias para realizar essa tarefa é detectar homologias entre as proteínas e relacionar novas sequências de proteínas à outras que possuam propriedades conhecidas. Nesse sentido, as proteínas são agrupadas em famílias que, por sua vez, são agrupadas em super-famílias a partir de suas propriedades.

Em [Jaakkola and Haussler, 1998] foi investigada a capacidade de SVM classificar famílias de proteínas em suas super-famílias. O sistema foi implementado usando um otimizador padrão (gradiente ascendente). Os experimentos foram realizados da seguinte forma: uma super-família particular (*glycosyltransferase*) foi dividida; uma das quatro maiores famílias dessa super-família foi separada para constituir o grupo de teste e as três restantes, o grupo de treinamento. Exemplos negativos foram extraídos de outras superfamílias para ambos os conjuntos. SVM superou métodos mais tradicionais nessa área, como o modelo de Markov escondido.

Um segundo problema envolve a tecnologia de microvetores de DNA, a qual possibilita a detecção dos genes que são expressos em um tecido particular e a comparação dos níveis

de expressões de genes entre tecidos. Isto é realizado através da extração de informações biológicas e da definição de funções para genes automaticamente. Em [Brown et al., 2000] pode-se encontrar a descrição de uma aplicação de SVM para a classificação de genes usando expressões de dados de gene de microvetores de DNA. Foram analisadas as expressões de dados de 2.467 genes *SAC - Charomices Cgrevisiae*, representados por um vetor de expressão de gene de 79 dimensões. Foram investigados quatro tipos de kernel. Função de Base Radial foi o kernel que apresentou os melhores resultados. O problema também foi investigado usando discriminante linear de Fisher, janelas de Parzen e Árvores de Decisão. SVM foi superior aos outros métodos em todas as bases de dados testadas. Em alguns casos, SVM apresentou o dobro da precisão alcançada pelos outros métodos.

3.5.4 Reconhecimento de Objetos

Recuperação de informação, filtragem de dados na Internet, aplicações médicas e detecção de objetos em cenas visuais, são algumas das áreas que precisam de reconhecimento de imagens. Algumas pesquisas que envolvem o reconhecimento, baseiam-se na extração de características de baixo nível de imagens, como *primal sketch* [M.Gomes et al., 1998], e de alto nível como detecção de bordas e descrição de formas, para capturar atributos relevantes da imagem [Gonzalez, 1993]. Essas formas de representação da imagem são úteis em aplicações específicas, porém, apresentam desvantagens, especialmente quando o problema é tentar desenvolver um sistema automático de reconhecimento. Ainda estão sendo pesquisadas técnicas para adquirir modelos da forma de amostras de objetos automaticamente [Nayar et al., 1996].

Há também problemas em que é difícil obter modelos geométricos de certos objetos. Para tentar solucionar esses problemas, estão sendo desenvolvidas nos últimos anos, pesquisas que utilizam estratégias de reconhecimento baseadas no aspecto, ou aparência, dos objetos. Essas estratégias vêem o problema de aprendizagem como a aquisição de um modelo compacto da aparência dos objetos, sob diferentes poses e direções de iluminação. Durante a fase de aquisição, os objetos são mostrados ao sensor de imagem em diversas orientações e condições de iluminação. O resultado é um grande conjunto de imagens de objetos. A COIL100 (Columbia Image Library) [Nayar et al., 1996] é uma base de dados padrão, disponível na Internet, cujas imagens foram adquiridas dessa forma e possibilitam esse tipo de reconhe-

cimento. A Figura 3.3 mostra imagens de um objeto, visto sob 10 ângulos diferentes, que pode ser reconhecido pela aparência.

Figura 3.3: O objeto durex com 10 visões diferentes.

Muitas técnicas têm sido utilizadas para a tarefa de reconhecer objetos a partir de várias visões dos mesmos. Em [Nayar et al., 1996] foi utilizada a técnica PCA para realizar o casamento entre objetos. O trabalho foi desenvolvido na base de dados COIL100. Essa base contém 100 imagens coloridas de objetos que foram adquiridas em intervalos de 5^0, 72 poses por objeto, formando um total de 7200 arquivos de imagens. O treinamento foi dividido em duas etapas: 1) PCA foi aplicado para reduzir a dimensão dos dados; 2) assumindo-se que as imagens de treinamento pertenciam a uma rotação uni-dimensional em 3D, um *spline* quadrático foi obtido através dos dados no *eigenspace* construído. O reconhecimento foi então realizado através de buscas do objeto cujo *spline* mais se aproximava da imagem que estava sendo reconhecida. Portanto, foi feita uma combinação das três técnicas: PCA, *Spline* e classificador Vizinho mais Próximo.

A mesma base de dados foi usada no reconhecimento de objetos utilizando o modelo de aprendizagem PAC (*Probably Approximately Correct*) em [Roth et al., 2000]. O método proposto não gera suposições a priori na distribuição estatística dos objetos e sim, evolui através das experiências anteriores. O objetivo do trabalho foi avaliar a possibilidade de aprendizagem e estudar os limites teóricos do que poderia ser aprendido. Foi usada a arquitetura de aprendizagem *SNoW (Sparse Network of Winnows)* para reconhecer os 100 objetos. Essa arquitetura, disponível em http://l2r.cs.uiuc.edu/$\sim$ cogcomp, constitui uma rede de unidades lineares sobre um espaço de características pré-definido e aprendido incrementalmente.

SVM também já foi aplicado ao problema de reconhecimento baseado na aparência. Além das vantagens do método SVM citadas anteriormente, SVM não requer extração de características ou redução da dimensão dos dados e pode ser aplicado diretamente em imagens, as quais são consideradas como pontos em um espaço N-dimensional [Pontil and Verri, 1998]. A alta dimensão do espaco de objetos contribui para que o Hiperplano de Margem

Máxima possa ser efetivamente encontrado. Isso faz a fase de reconhecimento ficar reduzida à decisão de qual lado do hiperplano um objeto de teste se encontra.

Um dos primeiros trabalhos envolvendo a aplicação de SVM ao reconhecimento de objetos baseado na aparência usando a base de imagens COIL100, foi [Pontil and Verri, 1998]. Eles não utilizaram todas as 100 imagens e o conjunto de treinamento foi construído com 36 visões dos objetos (variando a cada 10^o), sendo que as visões restantes, constituíram o conjunto de teste. Foi aplicado SVM linear que alcançou 100% de precisão. Foi realizado ainda outro experimento adicionando ruídos às imagens. Os resultados foram comparados à média obtida a partir de dez *Perceptrons* simples. A média de acertos obtida com SVM foi 93,00% enquanto que a média dos *Perceptrons* foi 87,34%. As ótimas taxas de reconhecimento obtidas foram provas de que SVM é adequado para esse tipo de tarefa de reconhecimento.

Em termos de comparação, em [Roth et al., 2000] foi realizado um estudo comparativo entre as técnicas Vizinhos Mais Próximos, SVM linear e SNoW, embora todas as 100 classes de objetos tenham sido usadas, foram considerados apenas quatro tamanhos de conjuntos de treinamento, com 4, 8, 18 e 36 visões cada.

Em [Roobaert, 1999], foram comparadas em termos de desempenho de classificação e custo computacional, as técnicas: SVM, Vizinhos Mais Próximos e *DirectSVM* (citado na seção 3.2). O *DirectSVM* apresentou desempenho similar à SVM, porém utilizou menos recursos computacionais e tempo de treinamento. Entretanto, o autor usou uma versão da base de dados com apenas 30 objetos e considerou apenas os seguintes subconjunto de visões (36, 12, 8, 4 e 2) para constituírem os conjuntos de treinamento.

3.5.5 Outros

Além das áreas citadas acima, SVM foi aplicado a outros problemas mais específicos. Em [Barabino et al., 1999] foi feito um estudo comparativo entre SVM e *Multi-Layer Perceptron* no problema de identificação de partículas em experimentos físicos de alta energia. Os resultados das duas técnicas foram similares. Em Visão de Máquina, SVM foi aplicado na modelagem de percepção através de atenção visual [Eghbalnia and Assadi, 2001]; em detecção de faces[Osuna et al., 1997b]; autenticação de faces[Jonsson et al., 1999] e aprendizagem de faces com multi-visões[Kwong and Gong, 1999].

3.6 Conclusão

Esta dissertação investiga a aplicação de SVM no problema de reconhecimento de objetos baseado na aparência. Os trabalhos citados acima, apontam para excelentes taxas de reconhecimento com SVM. Esta dissertação objetiva reforçar os resultados obtidos anteriormente.

O próximo capítulo descreve, inicialmente a utilização de SVM em uma base de imagens pequena (Minolta). Em seguida, são avaliados três tipos diferentes de kernel polinomial (linear, quadrático e cúbico) na base COIL100. Ambos os experimentos utilizam todas as classes de objetos e todas as visões. Também variam o tamanho dos conjuntos de treinamento e de teste. Na segunda etapa de experimentos ainda é realizado um *k-fold cross-validation* que divide os dados em k subconjuntos de tamanho aproximadamente igual, para reforçar a precisão dos resultados obtidos. Finalmente, a última etapa descreve um estudo comparativo entre SVM e Redes Neurais *Multilayer Perceptron Backpropagation*.

Capítulo 4

Reconhecimento de Objetos Baseado na Aparência Usando SVM

O reconhecimento baseado na aparência, descrito no Capítulo 3, é a solução mais adequada para problemas em que há dificuldades na obtenção de modelos geométricos dos objetos, desde que as imagens não apresentem oclusões e possam ser segmentadas da cena [Nayar et al., 1996]. No contexto de Visão de Máquina, SVM pode ser aplicado a muitos problemas, dentre eles, o reconhecimento de objetos 3D a partir de visões 2D.

Este capítulo apresenta experimentos práticos da aplicação de SVM a esse tipo de reconhecimento, que considera visões dos objetos e não sua geometria. É possível encontrar alguns trabalhos relacionados na literatura técnica. Em [Pontil and Verri, 1998] e [Roth et al., 2000] foram realizados experimentos simples (de 1 a 4 séries de treinamento/teste) e com apenas um tipo de kernel, o linear. Esta dissertação, entretanto, procurou aprimorar o que foi feito, utilizando exaustivamente o conjunto de imagens disponível, permitindo diferentes tamanhos dos conjuntos de treinamento/teste e implementando validação cruzada (*cross-validation*).

Adicionalmente, foram realizados experimentos e testes para comparar SVM sob três tipos de kernel polinomial diferentes: linear (grau 1), quadrático (grau 2) e cúbico (grau 3), e identificar o tipo mais adequado a este problema, a partir da precisão e do número de vetores de suporte apresentados por cada kernel. Sendo realizado ainda um experimento de validação cruzada utilizando o kernel que apresentou os melhores resultados, objetivando reforçar a precisão obtida. Finalmente, foi realizado também um estudo comparativo entre

SVM e Redes Neurais *Multilayer Perceptron Backpropagation*, devido ao fato desta última técnica ser tradicionalmente difundida em problemas de reconhecimento de objetos a partir de imagens e características.

Os experimentos foram separados em três grupos. No primeiro, que objetivou investigar o comportamento inicial de SVM no contexto do problema, a técnica foi aplicada usando kernel polinomial quadrático na pequena base de imagens Minolta, que contém apenas 10 classes de objetos. No segundo, o objetivo foi comparar os 3 tipos de kernel citados acima, usando a base de dados COIL100, que contém 100 classes de objetos. O terceiro experimento envolveu um estudo comparativo entre SVM e Redes Neurais, submetidos ao mesmo problema e estratégia de treinamento, utilizando um subconjunto de imagens COIL100.

Considerando que não foi objetivo desta dissertação construir uma implementação de SVM e sim realizar um estudo téorico e avaliar experimentalmente a aplicabilidade da técnica ao problema de reconhecimento baseado na aparência, e que já existem várias implementações disponíveis no domínio público, decidiu-se por fazer uma análise das ferramentas existentes afim de escolher as que mais se adequariam ao problema em questão. A próxima seção apresenta as ferramentas testadas, dando destaque às características, estratégias de implementação, vantagens, desvantagens, e os resultados obtidos. Essas descrições objetivam esclarecer que fatores contribuíram para as escolhas das bibliotecas para cada experimento. Concluindo este capítulo, os experimentos e resultados são apresentados.

4.1 Avaliação das Ferramentas

Como as pesquisas com SVM sofreram um maior desenvolvimento a partir de 1992, comparativamente com Redes Neurais, ainda há poucas ferramentas disponíveis que implementam essa técnica. Conforme foi destacado no Capítulo 3, Seção 3.4, os problemas com SVM para classificação binária estão bem estabelecidos, porém, para múltiplas classes, como nesta dissertação, ainda há pesquisas buscando métodos mais gerais. Portanto, essa característica do problema limitou ainda mais o número de ferramentas disponíveis.

Para realizar o primeiro experimento, buscou-se uma biblioteca para o programa Matlab. Por tratar-se de uma base de dados pequena e de um experimento que objetivava simplesmente construir um protótipo para investigar o comportamento de SVM no problema de

reconhecimento de objetos, uma *toolbox* para o simulador Matlab parecia o mais ideal. Três *toolboxs* foram avaliadas.

4.1.1 *Toolboxs* Matlab

Matlab SVM Toolbox - desenvolvida por Steve Gunn [Gunn, 1998] e disponível em http://www.isis.ecs.soton.ac.uk/resources/svminfo. Trata-se de uma biblioteca simples que realiza classificação e Regressão com SVM. A principal desvantagem é que essa *toolbox* requer o uso de uma rotina de Programação Quadrática à parte. É indicada para este propósito a *toolbox* de otimização do próprio Matlab. Diante dessa limitação está claro que essa ferramenta não utiliza nenhuma das estratégias de implementação citadas no Capítulo 3, Seção 3.3 e portanto, não pareceu adequada, especialmente por necessitar de uma rotina adicional para realizar o treinamento.

SVM Toolbox - esse programa desenvolvido por Gavin Cauley e disponível em http://theoval.sys.uea.ac.uk/∼ gcc/svm/toolbox/, realiza o treinamento usando o algoritmo SMO de Platt [Platt, 1999], implementado como uma biblioteca híbrida em Matlab e C++. A desvantagem apresentada é que, segundo o próprio autor, o programa ainda está em um estágio bastante inicial. Há muitas versões lançadas em curto espaço de tempo e praticamente não há documentação. Todos esses fatores indicaram que essa *toolbox* ainda está instável.

OSU SVM Classfier Matlab Toolbox - trata-se de uma *toolbox* recente para o Matlab (ano de 2000). Utiliza SVM padrão, chamado C-SVM, e duas das modificações citadas no Capítulo 3, Seção 3.2, ν-SVM e *One Class SVM*. Não precisa de nenhum pacote de software de programação quadrática adicional. Na prática, trata-se de uma interface para o Matlab da biblioteca LIBSVM [Chang and Lin, 2001a] desenvolvida por Junshui Ma e Stanley Ahal da Universidade do estado de Ohio, USA e disponível em http://eewww.eng.ohio-state.edu/∼ maj/osu_svm. O algoritmo base de LIBSVM e portanto, da OSU SVM é uma simplificação de SMO de Platt e de SVM^{light} de [Joachims, 1999].

Além dessas vantagens, essa biblioteca apresenta uma boa documentação e um curto tempo de resposta, pelo menos para SVM padrão (o qual foi testado nesta dissertação).

Todos esses fatores determinaram a escolha de OSU SVM para desenvolver o primeiro experimento. As características mais importantes tais como: forma de apresentação dos dados de entrada, variáveis e apresentação dos resultados, são descritas na seção que trata do experimento propriamente dito.

Para os outros experimentos entretanto, a OSU SVM apresentou alguns problemas com respeito à plataforma operacional. O Matlab é ideal quando há poucos dados para manipular. Como, a partir do segundo experimento, a base de dados utilizada foi a COIL100, com seus 7.200 arquivos, isso tornou-se um problema intratável pelo Matlab, devido ao ineficiente gerenciamento de memória do Sistema Operacional Windows, especialmente porque os conjuntos de treinamento e de teste tinham alta dimensão. Se o problema a ser resolvido manipular poucos dados, essa *toolbox* é mais indicada, devido às características destacadas e à conveniência da plataforma Matlab. Além disso, é possível também realizar testes com as outras extensões de SVM suportadas pela biblioteca, embora não tenham sido investigadas nesta dissertação. Outro fator importante é que com OSU SVM há a possibilidade de utilizar os principais tipos de kernel, com excessão do sigmóide.

Essas limitações do Matlab inviabilizaram o uso das *tooboxs* analisadas acima em problemas envolvendo grandes bases de imagens. Portanto, houve a necessidade de mais uma etapa de buscas por bibliotecas que apresentassem uma maior eficiência no gerenciamento dos recursos computacionais associado a alta dimensão dos dados. A plataforma escolhida foram programas C/C++ no Sistema Operacional Unix. Dentre as opções encontradas, as seguintes bibliotecas foram investigadas:

4.1.2 C/C++

SVMTorch - É uma implementação recente (dezembro de 2000) de SVM para classificação e egressão, desenvolvida por R. Collobert e S. Bengio [Collobert and Bengio, 2001], disponível em http://www.idiap.ch/learning/SVMTorch.html. O SVMTorch II, investigado nesta dissertação, foi totalmente escrito em C++. A estratégia de implementação é um método de decomposição que utiliza um *chunk* de tamanho 2 (ver Capítulo 3, Seção 3.3), que é similar ao método SMO. Trata-se de uma biblioteca ra-

zoavelmente documentada e com muitas opções para testes com SVM. Os arquivos de treinamento e de teste são matrizes onde cada linha representa uma amostra e o último termo de cada linha, representa um rótulo, 1 e -1 para classificação binária e 0, 1,. . . ,N para problemas com múltiplas classes. Além dessas características, é possível aplicar SVM com um kernel definido pelo usuário, além dos outros tipos padrão, incluindo o kernel sigmóide.

Entretanto, essa biblioteca apresenta a seguinte desvantagem: gera um arquivo de resultado (contendo número de vetores de suporte, os próprios vetores de suporte e demais informações pertinentes) para cada classe, após o treinamento. Na base de dados COIL100, por exemplo, que contém 100 classes, SVMTorch gerou 100 arquivos com resultados. A consequência foi a necessidade de muito espaço em disco, com o agravante de que, à medida que o conjunto de treinamento cresce, o espaço requerido pelos arquivos de resultados cresce proporcionalmente. Uma outra característica negativa da SVMTorch está relacionada ao tempo requerido para o treinamento, que foi o mais alto (aproximadamente 10 vezes mais lento que a biblioteca LIBSVM, descrita posteriormente). Esses fatores determinaram a interrupção dos experimentos usando SVMTorch, o que ocasionou a necessidade de continuar a busca por mais bibliotecas.

BSVM - Desenvolvida por Chih-Wei Hsu and Chih-Jen Lin e disponível em http://www.csie.ntu.edu.tw/∼ cjlin/bsvm/. Essa biblioteca foi a primeira implementação baseada em LIBSVM, descrita abaixo, usando inclusive as mesmas opções de parâmetros. Tem como diferencial a possibilidade de usar dois tipos adicionais de formulação para problemas com múltiplas classes, além de uma formulação diferente para problemas de Regressão, não pertinente a esta dissertação. Como o objetivo do segundo experimento foi comparar tipos de kernel e não tipos de algoritmos de SVM ou mesmo tipos de algoritmos para múltiplas classes, essa biblioteca não foi usada. Se porém, o problema envolvesse variações de formulação de SVM para múltiplas classes, a BSVM poderia ser uma opção viável.

LIBSVM - É uma ferramenta integrada para problemas de classificação, incluindo SVM padrão e ν-SVM, Regressão e também *One Class SVM*, que é uma tentativa de realizar aprendizagem não-supervisionada. Foi desenvolvida por Chih-

Chung Chang e Chih-Jen Lin [Chang and Lin, 2001a], está disponível em http://www.csie.ntu.edu.tw/∼ cjlin/libsvm/. O algoritmo base é uma simplificação de SMO e do método de decomposição de [Joachims, 1999]. As principais características são: diferentes formulações de SVM, eficiente em problemas multiclasse (*one-against-one*), validação cruzada para a seleção de variáveis, código fonte em C++ e Java. O código usado nesta dissertação foi o disponível em C++. Além dessas vantagens, os testes realizados com LIBSVM mostraram que trata-se de uma biblioteca estável. Disponibiliza todos os tipos de kernel mais comuns, incluindo o kernel sigmóide, gera apenas um arquivo com resultados a cada treinamento, tempo de execução mais rápido em relação às demais ferramentas investigadas, possibilita ao usuário gerenciar o espaço de memória requerido para a realização do treinamento e está bem documentada. Todas essas características determinaram a escolha de LIBSVM para a realização dos dois últimos e mais importantes experimentos descritos nesta dissertação.

Com as ferramentas escolhidas, a próxima etapa foi realizar os experimentos, que foram divididos em três grupos, como já foi citado anteriormente. As próximas seções descreverão passo-a-passo esses experimentos.

4.2 Experimentos Utilizando a Base de Dados Minolta

O objetivo destes primeiros testes foi avaliar o comportamento de SVM diante do problema de reconhecimento de objetos baseado na aparência. Para esse tipo de problema, como foi citado no Capítulo 3, Seção 3.5.4, várias imagens do objeto, visto sob ângulos diferentes ou de um ponto fixo apenas rotacionando o objeto, são capturadas. Isso possibilita a obtenção de várias visões de um mesmo objeto, imagens 2D, que podem ser usadas para simular sistemas de reconhecimento de objetos de natureza 3D. As próximas seções descrevem a base de dados, a biblioteca escolhida e o experimento propriamente dito.

4.2.1 Base de dados

Para realizar os experimentos foi utilizada uma base de dados da Universidade do Estado de Ohio, USA, disponível em http://sampl.eng.ohio-state.edu/∼ sampl/data/3DDB/RID/minolta. Essa base de imagens é composta por classes heterogêneas, organizadas em grupos de objetos, que não apresentam o mesmo número de visões por classe. Dentre as muitas opções disponíveis, foram selecionadas 10 classes de objetos (*angel, brain, bottle, duck, face, frog, horn, lobster, pooh* e *valve*) com 20 visões diferentes de cada. Essas visões foram escolhidas aleatoriamente,e, portanto, não seguem uma sequência de ângulos.

Conforme foi citado no Capítulo 1, Seção 1.3.5, as classes são conjuntos de objetos apresentando propriedades em comum e os objetos são instâncias concretas de uma classe. Entretanto, na base de dados Minolta, utilizada nesta etapa e na base de dados Coil100, utilizada nos próximos experimentos, as classes são constituídas unicamente por conjuntos de visões ou poses de um mesmo objeto.

4.2.2 Pré-processamento

Todas as imagens I=(R,G,B) foram convertidas para nível de cinza NC através da fórmula de conversão $NC = (R+G+B)/3$. As imagens foram também re-escalonadas para o tamanho 100x100 *pixels*, para este fim foi utilizada uma abordagem baseada em amostragem para fazer a conversão das imagens de tamanho arbitrário da base MINOLTA para o tamanho fixo de 100x100 pixels escolhido. Sejam f a imagem original e g a imagem re-dimensionada. Sejam (Xf, Yf) e (Xg, Yg) as dimensões das imagens originais e re-dimensionadas. Então a transformação de f para g pode ser expressa pela seguinte equação:

$$g(x,y) = f(\mathrm{INT}(\frac{x*Xf}{Xg}), \mathrm{INT}(\frac{y*Yf}{Yg})), \quad \forall x \in \{0,\ldots,Xg\}, y \in \{0,\ldots,Yg\} \quad (4.1)$$

na qual, INT é um operador que trunca seu argumento para um valor inteiro.

A Figura 4.1 mostra 10 visões de uma das classes utilizadas. A Figura 4.2 ilustra uma outra classe em que não apenas o ângulo variou, como também a escala, tornando mais difícil o problema de aprendizagem.

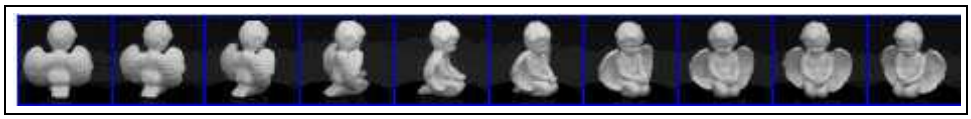

Figura 4.1: O objeto *angel* com 10 visões diferentes.

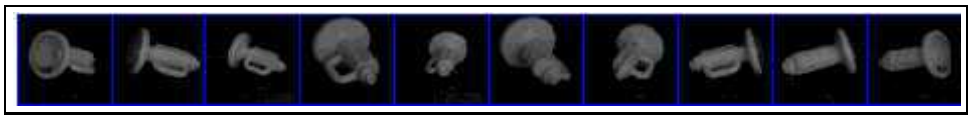

Figura 4.2: O objeto *horn* com 10 visões diferentes.

4.2.3 Ferramenta

O experimento foi realizado usando a *toolbox* para o Matlab OSU SVM, conforme seção anterior. Essa biblioteca lida com problemas de múltiplas classes, usando o método *" one-against-one"* (Capítulo 3, Seção 3.4) e não requer pacote de otimização extra. O kernel utilizado foi o polinomial quadrático e os padrões foram treinados usando o classificador SVM padrão (C-SVM na OSU SVM), que necessita de apenas três programas *osuSVMTrain*, *osuSVMClass* e *osuSVMTest*.

Conforme foi citado na seção anterior, essa biblioteca disponibiliza três tipos de algoritmos SVM, sendo que o algoritmo padrão, chamado C-SVM na *toolbox*, foi escolhido, já que o objetivo do experimento era investigar esse algoritmo. Também disponibiliza quatro tipos de kernel: linear, polinomial, Função de Base Radial e sigmóide. O tipo escolhido para este experimento foi o polinomial quadrático $K(x, x_i) = [(x.x_i) + 1]^d$, onde d representa o grau do polinômio (2 neste caso), por dois motivos: (a) segundo [Osuna et al., 1997b] esse tipo de kernel provê boa capacidade de generalização e (b) na maioria dos trabalhos envolvendo a aplicação de SVM ao reconhecimento de objetos baseado na aparência, o kernel utilizado foi o linear [Roth et al., 2000; Pontil and Verri, 1998].

Primeiramente, as matrizes de pixels das imagens foram atribuídas à vetores no Matlab. A biblioteca exige que as amostras sejam representadas por vetores coluna. Como o tamanho das imagens era 100x100, a dimensão de cada vetor foi 10.000x1. Outra exigência da OSU SVM para os dados de entrada é que os valores sejam normalizados para o intervalo [0...1] e convertidos para ponto flutuante. Com os dados preparados, a próxima etapa foi construir

os conjuntos de treinamento e de teste.

4.2.4 Construção dos Conjuntos de Treinamento e Teste

Foi utilizada a estratégia de construção de conjuntos de treinamento e de teste com diferentes tamanhos. Como tratavam-se de 10 classes de objetos, com 20 visões (imagens) diferentes de cada objeto, um total de 19 conjuntos de treinamento/teste foram gerados. Cada conjunto de treinamento usou T amostras por classe, e cada conjunto de teste usou (20-T) amostras por classe, onde T=1,2,. . . ,19. A seleção das amostras foi aleatória, embora manual, neste estágio. Portanto, os conjuntos de treinamento variaram de 1 a 19 visões por classe, formando grandes matrizes de pixels com dimensões variando entre 10.000x10 e 10.000x190. Os conjuntos de teste apresentaram a mesma variação, mas na direção inversa.

Para cada conjunto de treinamento e de teste foi construído um vetor contendo: os rótulos das classes de objetos, os rótulos para os treinamentos e as saídas desejadas, para verificar-se a precisão na fase de testes. Os valores dos rótulos variaram de 1 a 10 (10 classes). O parâmetro T determinou também o tamanho dos vetores de rótulos. Os vetores de rótulos referentes aos conjuntos de treinamento variaram de 1x10 a 1x190. Os dos conjuntos de teste apresentaram a mesma variação, mas na direção oposta.

4.2.5 Treinamentos/Testes

Prosseguindo, 19 sequências de treinamento/teste foram construídas. Para realizar o treinamento, o programa usado foi o *osuSVMTrain*. Como já foi citado, o algoritmo utilizado foi o classificador padrão (CSVM). O kernel foi o polinomial quadrático (grau $d = 2$). Foi mantido o valor *default* para o parâmetro de regularização C (igual a 1) e demais variáveis. Apesar da maioria dos autores sugerirem que seja feita uma validação cruzada com valores de C de 1 a 1.000, para definir o mais adequado ao problema, não foi necessário fazer isso, uma vez que os resultados obtidos com o valor *default* foram satisfatórios. A biblioteca caracteriza-se por gerenciar espaço em disco, o usuário pode especificar o tamanho de espaço que deve ser reservado para o treinamento. A Tabela 4.1 sintetiza os parâmetros acima.

O *osuSVMTrain* foi executado com cada arquivo de treinamento, o respectivo vetor de rótulos e os parâmetros citados acima. As saídas do programa foram as seguintes:

Tipo de SVM	Tipo de Kernel	Grau	C
Classificador padrão C-SVM	Polinomial	2	1

Tabela 4.1: Valores definidos para o treinamento na base Minolta.

- AlphaY - Alpha*Y, onde Alpha são os coeficientes de Lagrange não nulos e Y, os rótulos correspondentes;
- SVs: Padrões correspondentes ao Alphas não nulos, ou seja, os vetores de suporte.
- Bias: Limiar da função de decisão.
- Para: Os mesmos parâmetros de entrada.
- Ns: Número de vetores de suporte para cada classe.

Com os resultados obtidos, o segundo programa *osuSVMClass*, foi utilizado para classificar os dados de teste. As entradas para *osuSVMClass* foram as saídas do algoritmo anterior, acrescidas do conjunto de teste. As saídas foram os rótulos da classe estimada para cada imagem do conjunto de teste e os *scores*, que equivalem à saída das funções de decisão de cada classe.

Finalmente, o terceiro programa, *osuSVMTest*, foi utilizado para verificar a precisão do classificador gerado. As entradas foram as mesmas do algoritmo anterior, acrescidas dos rótulos (saídas desejadas) dos padrões de teste. As saídas do programa *osuSVMTest* foram a matriz de confusão, mostrando a precisão alcançada para cada classe e os mesmos *scores* associados à funções de decisão de cada classe.

4.2.6 Resultados

A taxa média de reconhecimento foi 90%. A melhor taxa foi 98%, para T=13 e a pior foi 71%, para T=1. A Figura 4.3 ilustra o desempenho de SVM no problema de reconhecimento. É importante observar que a curva dessa figura representa a média de reconhecimento para as 10 classes. A curva de reconhecimento começa a cair com os maiores conjuntos de treinamento porque para algumas classes (1 e 8 em especial), as imagens que constituíram o conjunto de teste foram as mais difíceis de reconhecer.

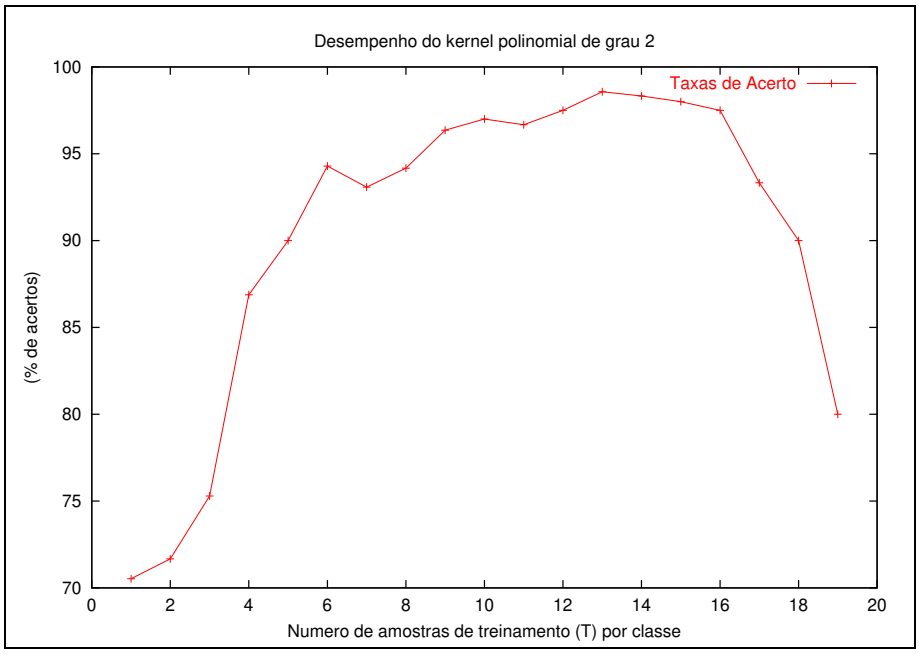

Figura 4.3: Desempenho de SVM na base Minolta.

A Tabela 4.2 mostra a matriz de confusão da fase de teste, para T=1, ou seja, cada conjunto de teste continha 19 visões de cada classe. Cada linha e coluna da tabela representam classes, sendo que em cada célula está contido o valor percentual obtido quando o padrão pertencente à classe referenciada na linha da célula foi classificado como sendo da classe referenciada na coluna da célula. A soma dos valores das células deve ser igual à 100%, para cada linha. A Tabela 4.3 mostra a matriz de confusão para T=19 (apenas 1 visão para cada classe). Os valores estão distribuídos como na tabela anterior. É importante observar que em uma situação de classificação ideal, a diagonal principal da matriz deverá possuir valores altos (próximos de 100%). Percebe-se portanto, que na Tabela 4.2 as classes 4 e 7 foram as que mais tiveram problemas de serem classificadas e na Tabela 4.3, as classes 1 e 8 foram problemáticas.

Os resultados obtidos ratificaram que SVM é uma técnica totalmente aplicável ao problema, com altas taxas de reconhecimento. Esse experimento revelou que a precisão de SVM diminui com os conjuntos de treinamento pequenos, que é um comportamento comum entre as técnicas aplicadas em Reconhecimento de Padrões.

Classes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	57.89	0.00	31.58	0.00	0.00	0.00	0.00	0.00	10.53	0.00
2	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	26.32	73.68	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	26.32	**21.05**	0.00	15.79	31.58	5.26	0.00	0.00
5	0.00	0.00	0.00	5.26	84.21	0.00	0.00	10.53	0.00	0.00
6	0.00	0.00	5.26	0.00	0.00	57.89	36.84	0.00	0.00	0.00
7	0.00	68.42	0.00	0.00	0.00	0.00	**31.58**	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	10.53	89.47	0.00	0.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100	0.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100

Tabela 4.2: Matriz de confusão para T=1. Cada célula contém o valor percentual de confusão para as classes indicadas por sua linha quando classificadas como sendo das classes indicadas por suas colunas.

Classes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**0.00**	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	100	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	100	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	100	**0.00**	0.00	0.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100	0.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100

Tabela 4.3: Matriz de confusão para T=19.Cada célula contém o valor percentual de confusão para as classes indicadas por sua linha quando classificadas como sendo das classes indicadas por suas colunas.

4.3 Experimentos Utilizando a Base de Dados COIL100

Os experimentos com a base de imagens COIL100 foram divididos em duas etapas: testar diferente kernels e comparar SVM com uma abordagem mais tradicional utilizando Redes Neurais do tipo *Multilayer Perceptron Backpropagation*. A seguir são descritos com mais detalhes a base de imagens, as ferramentas utilizadas e as duas etapas de experimentos.

4.3.1 Base de Dados

A base de imagens COIL100 utilizada nestes experimentos, está disponível em http://www.cs.columbia.edu/CAVE. Como foi citado no Capítulo 3, Seção 3.5.4, essa é uma das melhores bases de dados para investigar algoritmos de reconhecimento baseado na aparência, em particular [Roth et al., 2000; Pontil and Verri, 1998] e [Murase and Nayar, 1995] realizaram experimentos utilizando esta base de imagens. COIL100 consiste de 7.200 imagens coloridas de 100 objetos diferentes.

As imagens foram adquiridas com os objetos posicionados no centro de uma plataforma giratória (*turntable*) e fotografados por uma câmera numa posição fixa. Um total de 72 poses por objeto foram obtidas rotacionando-se a mesa giratória a cada 5^o. As imagens foram normalizadas para uma retina de 128x128 pixels. Em [Murase and Nayar, 1995] encontram-se maiores detalhes sobre o processo de aquisição das imagens. A Figura 4.4 mostra alguns dos objetos que compõem a COIL100, vistos sob o ângulo 10^o e convertidos para nível cinza.

Figura 4.4: Alguns dos objetos que compõem a base COIL100.

Como ocorreu uma normalização de todas as imagens para uma retina fixa, dependendo do ângulo de visão, algumas imagens de um mesmo objeto se tornaram relativamente maiores que outras. A Figura 4.5 mostra 9 imagens, do objeto 44, variando do ângulo 260^o ao 300^o.

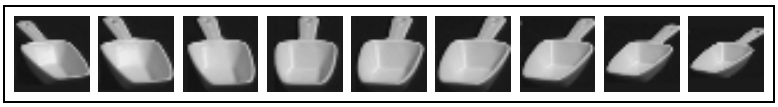

Figura 4.5: Visões do objeto 44, do ângulo 260^o ao 300^o.

4.3.2 Ferramentas

A Seção 4.1 mostrou a dificuldade de encontrar uma ferramenta de SVM que suprisse a maioria das exigências de cada experimento. A biblioteca escolhida para o estudo comparativos entre tipos kernel foi a LIBSVM, que apresentou-se mais adequada ao problema, segundo razões já descritas na Seção 4.1.

Para os experimentos com *Multilayer Perceptron Backpropagation* foi utilizado o simulador SNNS (Stuttgart Neural Network Simulator), disponível em http://www-ra.informatik.uni-tuebingen.de/SNNS.

4.3.3 Testando Diferentes Kernels

O objetivo nesta etapa, foi fazer uma avaliação prática sobre a precisão, comportamento e número de vetores de suporte produzidos por três tipos de kernel polinomial: linear, quadrático e cúbico. Uma das características de SVM destacada na literatura é o fato do usuário definir poucos parâmetros livres para realizar alguma tarefa. Com o kernel polinomial é necessário definir apenas o grau do polinômio.

No Capítulo 2, Seção 2.3.3, além do kernel polinomial, dois outros tipos de kernel foram apresentados: Função de Base Radial (RBF) e sigmóide. Entretanto, estes não foram utilizados nos experimentos por dois motivos principais: no kernel RBF $K(x, x_i) = e^{-\|x-x_i\|^2/2\sigma^2}$ há dificuldade na definição do parâmetro σ. Segundo [Matter and Haykin, 1999] essa variável não tem um valor fixo, assim como o grau do polinômio para o kernel polinomial, porém, normalmente é necessário usar alguma estratégia de treinamento, como validação cruzada, para determinar que valor de σ é mais indicado para cada tipo de problema. Entretanto, essa investigação ultrapassa o escopo desta dissertação.

Segundo [Vapnik, 1999], o kernel sigmóide $K(x, x_i) = S[v(x, x_i) + c]$ não satisfaz as condições de Mercer (Capítulo 2, Seção 2.3.3) para todos os valores de v e c. Diante disso, também seria necessário realizar experimentos para definir que valores dessas variáveis (sa-

tisfazendo as condições de Mercer) seriam mais adequados para o problema em foco. Essa restrição também impossibilitou o uso desse kernel. A investigação desses valores ideiais é objeto de uma pesquisa em separado que também ultrapassaria o escopo desta dissertação.

O sistema de reconhecimento pode ser dividido em quatro estágios: pré-processamento, constução dos conjuntos de treinamento/teste, treinamento dos classificadores e avaliação de desempenho. A Figura 4.6 ilustra as etapas envolvidas no experimento.

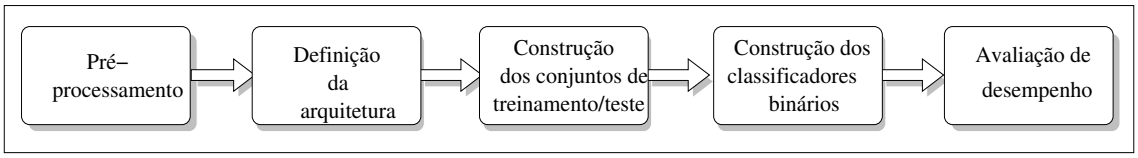

Figura 4.6: Arquitetura do experimento 2.

Pré-processamento

Esta etapa objetivou preparar as imagens para o reconhecimento. Como no experimento anterior, todas as imagens I=(R,G,B) da COIL100 foram convertidas para níveis de cinza NC, através da fórmula de conversão $NC = (R + G + B)/3$, com pixels assumindo valores no intervalo $[0 \ldots 255]$. A seguir, as imagens foram redimensionadas para uma retina de 32x32 pixels, através da média dos níveis de cinza sobre subconjuntos de 4x4 pixels, para reduzir a dimensão do espaço de entrada utilizado nos treinamentos e testes. As imagens portanto, foram transformadas em vetores de 1024 componentes. Os trabalhos descritos em [Pontil and Verri, 1998; Roth et al., 2000] e [Roobaert et al., 2000], aplicaram pré-processamento idêntico a este.

É importante notar que, embora o pré-processamento acima reduza o tempo computacional e memória necessários para a realização do algoritmo SVM, algumas classes de objetos podem tornar-se mais difíceis de discriminar uma vez que os padrões tornam-se mais próximos uns dos outros devido à redução do nível de informação condificada nas imagens.

Formação dos conjuntos de treinamento/teste

O método usado para criar os grupos de treinamento e de teste foi o mesmo do experimento anterior, ou seja, construção de conjuntos de treinamento/teste de diferentes tamanhos. As

100 classes foram utilizadas e todas as visões destas também. Um total de 71 conjuntos de treinamento e de teste foram criados, cada um com T amostras de treinamento e (72-T) amostras de teste por classe, onde T=1,2,. . . ,71. É importante destacar que as visões que constituíram cada conjunto de treinamento foram escolhidas de forma aleatória, tomando-se as devidas precauções para que as visões não se repetissem.

Essa estratégia difere de outros trabalhos publicados em três aspectos relativos à criação dos grupos de treinamento e teste:

1. Em [Pontil and Verri, 1998] as 100 classes não foram manipuladas juntas. Para cada experimento, apenas um subconjunto de 32 classes de objetos, de um total de 100, foi manipulado. O fato de trabalhar apenas com 32 classes, pode elevar artificialmente as taxas de reconhecimento. Entretanto, nesta dissertação, os grupos de treinamento e de teste foram constituídos envolvendo todas as 100 classes.

2. Em [Pontil and Verri, 1998], os conjuntos de treinamento consistiram de 36 imagens (36 visões) das 32 classes escolhidas. Em [Roth et al., 2000] foram construídos quatro grupos de treinamento, com respectivamente 4, 8, 18 e 36 visões. Nesta dissertação, porém, foram construídos 71 conjuntos de treinamento e de teste, variando de 1 a 71 visões do objeto, para realizar um experimento exaustivo na avaliação do comportamento de SVM com poucas e muitas visões.

3. Nesta dissertação, cada um dos cojuntos de treinamento foi constituído por visões dos objetos escolhidas aleatoriamente, para obter resultados mais realísticos. Em [Pontil and Verri, 1998], 36 imagens constituíram os conjuntos de treinamento (uma a cada 10^o) e as outras 36 formaram os conjuntos de teste. Pela Figura 4.5 é possível perceber que as imagens em uma sequência de 10^o são altamente correlacionadas, logo, se a cada 10^o uma imagem constou no conjunto de treinamento e a outra no grupo de teste, o problema de classificação torna-se mais simples. Enquanto que nesta dissertação, como a seleção das visões foi aleatória, elas podem tanto ser de ângulos próximos, como distantes.

Treinamento dos classificadores

Como já foi citado em capítulos anteriores, o padrão utilizado para representar o objeto no reconhecimento baseado na aparência são as próprias imagens. O conjunto de treinamento portanto, torna-se uma grande matriz de pixels. A biblioteca LIBSVM, usada para realizar os experimentos, apresenta algumas especificações para a formatação dos dados da matriz de pontos. Para LIBSVM, cada linha da matriz que constitui o conjunto de treinamento, deve seguir a seguinte ordem:

- <rótulo> - É o valor alvo do dado de treinamento. Deve ser um inteiro que identifica uma classe. Nesta dissertação, o rótulo variou de 1 a 100, já que a COIL100 é composta por 100 classes. Os rótulos nos arquivos de teste são usados para medir a precisão da classificação.

- <índice> - É um inteiro começando por 1. Como as imagens foram redimensionadas para 32x32, o índice aqui, assume valores entre 1 e 1024.

- <valor> - É um número real. Corresponde ao valor dos pixels das imagens. Como foi citado acima, as imagens foram convertidas para nível de cinza, logo, aqui o valor varia entre 0.000000 e 255.000000.

  A Figura 4.7 ilustra um arquívo típico para LIBSVM. É importante destacar que essa figura foi montada com partes do arquivo de entrada de LIBSVM, apenas a título de ilustração.

Portanto, os arquivos contendo as matrizes dos conjuntos de treinamento variaram de 100 a 7.100 linhas, e os de teste o inverso, de acordo com parâmetro T. As colunas seguiram o formato requerido pela biblioteca. O esquema utilizado por LIBSVM para solucionar o problema de múltiplas classes é o "um-versus-um" (*"one-against-one"*), descrito no Capítulo 3, Seção 3.4 (para maiores detalhes ver [Chang and Lin, 2001a]). A biblioteca apresenta dois programas: *svm-train* para realizar os treinamentos e *svm-predict* para realizar os testes. As opções para os tipos de SVM são: C-SVM (classificador padão); ν-SVM (extensão de C-SVM); *one-class SVM* (extensão de C-SVM); ϵ-SVR (SVM padrão para Regressão) e ν-SVR (extensão de SVR).

```
1 1:0.360784 2:0.098039 3:0.109804 4:0.329412 5:0.101961 6:0.105882 7:0.113725
8:0.101961 9:0.098039 10:0.109804 11:0.098039 12:0.098039 13:0.098039 14:0.098039
15:0.098039 16:0.098039 17:0.098039 18:0.098039 19:0.098039 20:0.098039 21:0.098039
22:0.098039 23:0.098039 24:0.098039 25:0.098039 26:0.098039 27:0.098039 28:0.098039
29:0.098039 30:0.098039 31:0.098039 32:0.098039 33:0.701961 34:0.098039 35:0.109804
36:0.341176 37:0.101961 38:0.105882 39:0.439216 40:0.101961 41:0.098039 42:0.109804
43:0.098039 . . . 1021:0.098039 1022:0.098039 1023:0.098039 1024:0.098039
.
.
.
100 1:0.439216 2:0.121569 3:0.580392 4:0.650980 5:0.156863 6:0.611765 7:0.745098
8:0.125490 9:0.611765 10:0.772549 11:0.125490 12:0.635294 13:0.803922 14:0.101961
15:0.666667 16:0.458824 17:0.098039 18:0.803922 19:0.109804 20:0.101961 21:0.101961
22:0.101961 23:0.101961 24:0.101961 25:0.101961 26:0.101961 27:0.101961 28:0.101961
29:0.101961 30:0.101961 31:0.101961 32:0.101961 33:0.600000 34:0.121569 35:0.580392
36:0.729412 37:0.156863 38:0.611765 39:0.760784 40:0.125490 41:0.611765 42:0.780392
43:0.125490 . . . 1021:0.101961 1022:0.101961 1023:0.101961 1024:0.101961
```

Figura 4.7: Arquivo típico de entrada para a biblioteca LIBSVM.

O tipo de SVM aplicado foi o classificador padrão e os tipos de kernel usados foram: polinomial (graus 1, 2 e 3). As demais opções disponíveis na biblioteca (RBF e sigmoide) não constam nesse experimento por razões já explicadas anteriormente.

Antes de iniciar os treinamentos propriamente ditos, foi necessário realizar alguns testes para verificar os valores mais indicados para algumas variáveis. Com relação ao parâmetro de regularização C é aconselhável testar valores entre 1 e 1000 para uma definição mais precisa. As precisões mais elevadas estiveram associadas ao valor 1000 para a variável C. Em LIBSVM, o kernel polinomial usado é da forma: $(\gamma.x.x_i)^d$, onde γ é uma constante multiplicada por $(x.x_i)$ e d representa o grau do polinômio. Alguns valores de γ foram testados e as precisões mais elevadas estiveram associadas ao valor 10.

Como já foi citado anteriormente, LIBSVM utiliza um método de decomposição (conforme Capítulo 3, Seção 3.3), para a solução do problema quadrático convexo. Para tal, utiliza uma variável que implementa o critério de parada ϵ para os subconjuntos criados pelo método de decomposição. O valor de ϵ deve ser pequeno e positivo. Resultados de testes variando o valor de ϵ, indicaram o valor 1 como ideal. A Tabela 4.4 sintetiza os parâmetros definidos para o treinamento.

Grau do polinômio (d)	C	γ	ϵ
1, 2 e 3	1000	10	1

Tabela 4.4: Valores definidos para as variáveis: grau do polinômio (d), C, γ, ϵ.

Inicialmente os 71 arquivos de treinamento e de teste foram gerados, T amostras de visões para treinamento e (72-T) amostras de visões para os conjuntos de teste. Para cada conjunto de treinamento, o programa *svm-train* foi executado. Primeiro, com kernel linear, em seguida com kernel quadrático e, finalmente, cúbico. *Svm-train* gera um arquivo de resultados contendo: tipo de SVM, tipo de kernel, número de classes, total de vetores de suporte, rótulos, e os próprios vetores de suporte. A Figura 4.8 ilustra uma parte do arquivo de resultado associado a T=51 do kernel linear. Essa figura foi montada com partes do arquivo T=51 apenas a título de ilustração, para mostrar que informações ficam disponíveis nos arquivos de resultados.

```
svm_type c_svc
kernel_type linear
nr_class 100
total_sv 4093
rho −2.25531 −3.9943 −0.355322 −2.52852 6.13458 −1.06955 −0.642801 −1.41527 −3.34511 0.51
2457 −1.35894 −1.22204 −1.4646 7.79981 −1.65038 −1.32257 −1.259 5.72349 −1.33399 −1.38653
−1.22223 2.94618 −1.72973 −1.41028 −1.14974 2.48644 0.296467 −1.19072 −1.17959 −3.50551
−1.21261 −0.833655 −0.305283 −2.77361 1.0973 −3.81818 3.80001 −1.61083 −4.35315 −1.43541
2.99603 −1.38233 −1.15241 −1.47414 −6.54067 −2.03663 −2.48414 −1.01391 −1.51639 −1.27488
label 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 6
1 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100
nr_sv 49 30 50 47 24 50 38 49 39 40 48 45 37 49 49 49 34 29 50 49 47 43 48 31 23 20 49 50
45 29 50 46 20 37 16 36 49 50 44 39 48 48 41 45 44 50 28 38 41 18 48 50 42 49 42 28 50 3
1 50 50 26 48 50 27 49 42 50 50 50 14 19 31 12 37 48 50 44 49 50 50 44 49 39 49 44 41 36
22 45 45 49 24 48 29 23 49 50 50 43 50
SV
0 0 0 0 0 0 5.05749740679224e−06 −0 0 0 0 0 0 6.24267703482509e−05 0 0 0 2.37754618210262
e−05 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8.687928491799258e−08 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 −0 0 0 0 0 0 0 0 0 0 0 0 0 −0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1:120 2:28 3:25 4:25 5:25 6:25 7:25 8:25 9:25 10:25 11:25 12:25 13:25 14:25 15
:25 16:25 17:25 18:25 19:25 20:25 21:25 22:25 23:25 24:25 25:25 26:25 27:25 28:25 29:25 3
0:25 31:25 32:25 33:120 34:28 35:25 36:25 37:25 38:25 39:25 40:25 41:25 42:25 43:25 44:25
45:25 46:25 47:25 48:25 49:25 50:25 51:25 52:25 53:25 54:25 55:25 56:25 57:25 58:25 59:2
```

Figura 4.8: Parte do contéudo de um arquivo de resultado de treinamento, correpondendo a T=51.

A relação entre o número de vetores de suporte obtidos após o treinamento e o tamanho do conjunto de treinamento correspondente pode ser visto na Figura 4.9. Uma vez que tratam-se de 100 classes, o número de padrões de treinamento na figura é mostrado como

$(100 * T)$. É importante notar que à medida que o número de visões aumenta, o número de vetores de suporte também aumenta, mas não em proporções iguais, o que confirma o comentário de [Pontil and Verri, 1998] que a fração de vetores de suporte é maior quando há uma alta dimensão de entrada combinada com um número pequeno de exemplos de treinamento. Para conjuntos de treinamento pequenos, é possível notar que o número de vetores de suporte é praticamente igual ao número de padrões de treinamento, o que torna os requerimentos de tempo e memória da técnica SVM muitos próximos de um classificador Vizinho Mais Próximo [Duda et al., 2000], por exemplo.

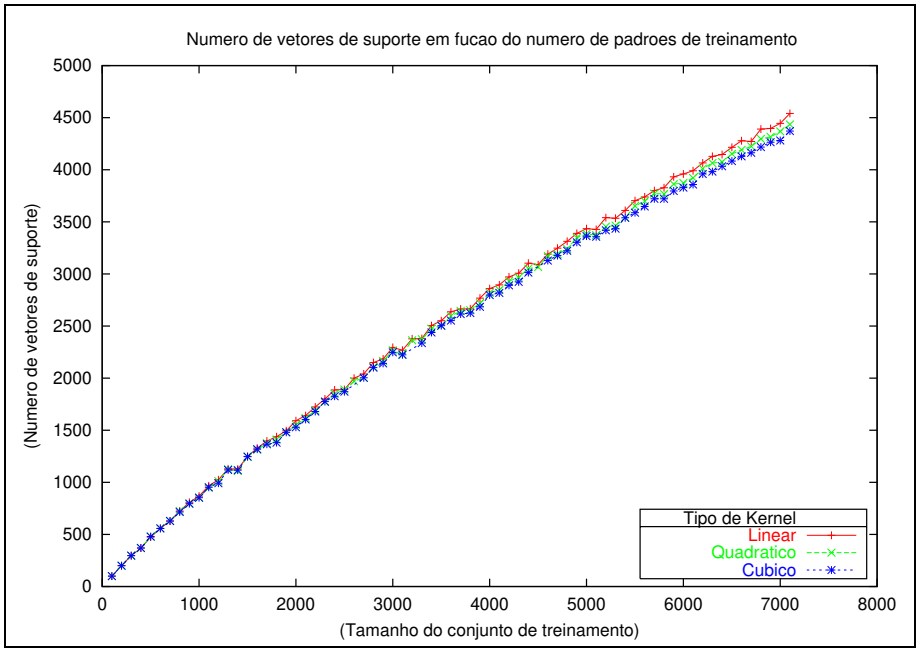

Figura 4.9: Número de vetores de suporte criados versus tamanho do conjunto de treinamento $(100 * T)$ para os três tipos de kernel investigados.

Avaliação de Desempenho

Com os arquivos de resultados gerados no treinamento, foi possível realizar a etapa de avaliação do desempenho de SVM na classificação de objetos baseada na aparência. O programa *svm-predict* foi então utilizado. Para executar o programa foi nessário passar como parâmetro o arquivo de teste (variando de 1 a 71 visões por classe) e o arquivo de resultado. *Svm-predict* gerou um arquivo de saída ilustrando a classificação de cada objeto e exibindo a precisão alcançada. Para uma avaliação mais completa dos resultados obtidos, essas

precisões foram todas armazenadas em arquivos em disco.

Dentre os três tipos de kernel investigados, o kernel polinomial quadrático apresentou desempenho um pouco superior aos demais. A Figura 4.10 ilustra esses resultados através da curva de reconhecimento obtida a partir da média das 100 classes. As curvas de reconhecimento não decaíram com os últimos conjuntos de treinamento, como no experimento anterior, porque desta vez, tratava-se de um número elevado de classes e visões, portanto, mesmo que para algumas classes tenham existido imagens mais difíceis de reconhecer, esse fato não determinou uma queda brusca na taxa de reconhecimento.

Pode-se notar que a diferença de desempenho entre os kernels é praticamente mínima. Isto confirma os resultados obtidos por [Vapnik, 1999], o qual mostrou que o tipo de kernel escolhido não possui uma importância fundamental para os resultados finais de reconhecimento.

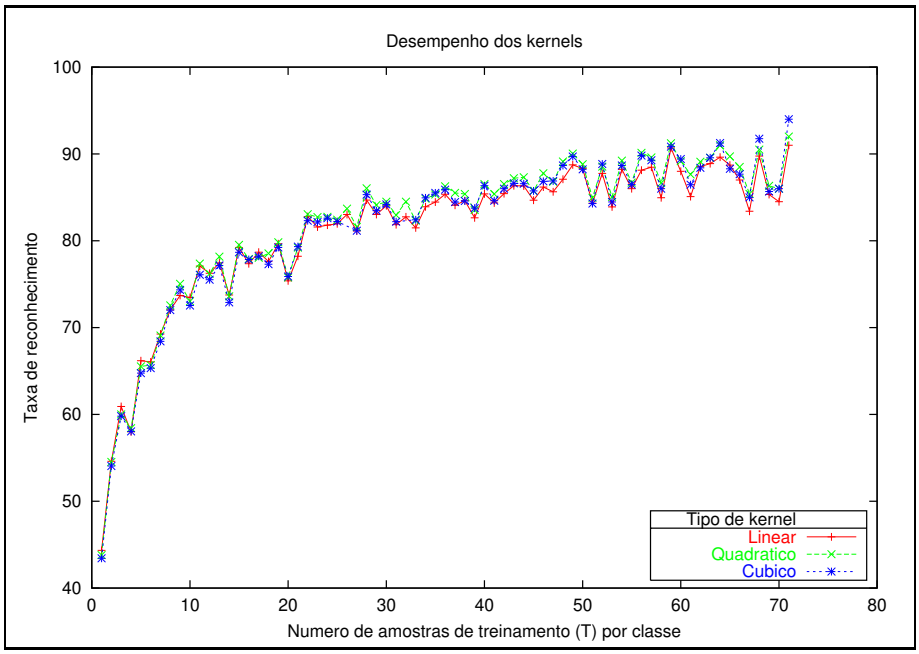

Figura 4.10: Curvas de reconhecimento para os kernels: linear, quadrático e cúbico, na base COIL100.

A taxa média de reconhecimento foi 82,00% para o kernel quadrático, 81,48% para o cúbico e 81,20% para o linear. A melhor taxa foi 94,00% para o kernel cúbico, com T=71 e a pior taxa foi 43,42% também para o kernel cúbico, a Tabela 4.5 resume esses dados.

É importante notar que as piores taxas de reconhecimento ocorreram para os menores

Tipo de Kernel	**Pior Taxa**	**Melhor Taxa**	**Taxa Média**
Linear	44,34% T=1	91,00% T=71	81,20%
Quadrático	43,80% T=1	92,00% T=71	82,00%
Cúbico	43,42% T=1	94,00% T=71	81,48%

Tabela 4.5: Taxas de reconhecimento obtidas por três diferentes kernels: linear, quadrático e cúbico na base COIL100.

tamanhos de conjunto de treinamento e a taxa aumentou gradualmente para os tamanhos de conjuntos maiores, fato que já era esperado. As taxas de reconhecimento foram em geral menores do que as taxas obtidas em trabalhos anteriores, como em [Pontil and Verri, 1998; Roth et al., 2000]. Algumas das razões para esses resultados obtidos são:

1. Condições de treinamento diferentes. Como já foi citado, a estratégia de usar diferentes tamanhos de conjuntos de treinamento/teste, forçou um treinamento exaustivo dos dados, diferentemente de outros trabalhos que realizaram 1, como em [Pontil and Verri, 1998] ou no máximo 4 treinamentos como em [Roth et al., 2000].

2. Escolha aleatória das visões. As visões selecionadas para constituírem os conjuntos de treinamento poderíam tanto facilitar quanto dificultar a classificação.

3. Utilização de todas as 100 classes. Esse fator também pode ter influenciado nos resultados, uma vez que essa base de dados apresenta alguns objetos muito parecidos.

Para reforçar os resultados obtidos nesta etapa, foi implementada a estratégia *k-fold cross validation* com o kernel polinomial quadrado, que apresentou uma pequena superioridade em relação aos dois outros tipos de kernel testados. Os valores dos parâmetros C, ϵ e γ foram os mesmos do estudo comparativo entre kernels, ou seja, 1.000, 1 e 10, respectivamente. A biblioteca LIBSVM disponibiliza a opção de *k-fold cross validation*, o usuário escolhe apenas o valor de k. Foi escolhido $k = 10$, conforme recomendado pela literatura [Haykin, 1999].

A estratégia de *cross-validation* é utilizada para avaliar o desempenho de generalização de algoritmos de aprendizagem. A variante desta estratégia denominada *k-fold cross-validation*, utilizada nesta dissertação, divide os dados em k subconjuntos de tamanho apro-

ximadamente igual. O treinamento é repetido k vezes, sendo que, a cada iteração, um subconjunto é retirado do conjunto geral de padrões e utilizado como conjunto de teste. Ao final, todos os subconjuntos são treinados e testados. A taxa de erro de generalização é a média das taxas de erro nas diferentes partições.

Foi gerado um conjunto contendo todas as classes e visões dos objetos. Portanto, o conjunto foi composto por 7.200 padrões. Como o valor de k escolhido foi 10, o grande conjunto de dados foi particionado em 10 subconjuntos, com 720 padrões cada. A precisão média obtida por *k-fold cross validation* foi 87,56%. A Tabela abaixo resume esses dados.

Valor de k	**Precisão Média**
10	87,56

Tabela 4.6: Precisão média alcançada utilizando *k-fold cross validation*, com k=10.

Esses resutados, indicam que SVM utilizando o kernel polinomial quadrático (poderia ser outro tipo de kernel, conforme experimento anterior) pode alcaçar a precisão 87,56% na etapa de classificação. A diferença que houve entre o resultado obtido com os diversos tamanhos de conjuntos de treinamento/teste e seleção aleatória das visões em relação ao resultado de *k-fold cross-validatio* deve-se à influência que as visões exerceram na classificação. A validação cruzada permite uma avaliação mais precisa do classificador.

4.3.4 Estudo Comparativo: SVM x Redes Neurais

O objetivo nesta etapa foi realizar experimentalmente um estudo comparativo entre SVM e Redes Neurais do tipo MLP *Backpropagation* enfatizando apenas aspectos relativos à precisão de reconhecimento. Não foi objetivo desta dissertação comparar o desenvolvimento das técnicas em termos de tempo de execução e memória. Redes Neurais é uma técnica muito utilizada em problemas de reconhecimento de padrões, como o reconhecimento baseado na aparência.

As ferramentas utilizadas nesta etapa, foram: LIBSVM para SVM e SNNS para Redes Neurais MLP *Backpropagation*. Essas bibliotecas foram descritas no início da seção. A base de dados foi novamente a COIL100.

Inicialmente tentou-se realizar os experimentos com uma rede monolítica, com 1024

neurônios na camada de entrada (devido à dimensão das imagens, 32x32), 100 neurônios na camada de saída (devido às 100 classes de objetos) e a quantidade de neurônios na camada escondida sendo definida por experimentação. Essa estratégia porém, não mostrou-se adequada devido ao elevado tempo de convergência das diversas configurações testadas. Esse fator deve-se especialmente à quantidade de classes envolvidas no problema de classificação. À medida que os conjuntos de treinamento aumentavam de tamanho, a rede necessitava de muito mais tempo para convergir e algumas vezes não convergia após vários dias de simulação

Diante desse problema, foi necessário buscar-se uma outra estratégia. Em seções anteriores foi citado que a estratégia de classificação multiclasses utilizada por LIBSVM é "um-versus-um", que constrói $N(N-1)/2$ classificadores binários, onde N é o número de classes. Para realizar o experimento comparativo optou-se por reproduzir a mesma estratégia de treinamento com Redes Neurais. Entretanto, percebeu-se que para as 100 classes de objetos da COIL100, seria produzida uma quantidade elevada de classificadores (4.950) para cada conjunto de treinamento. Considerando-se a mesma variação no tamanho dos conjuntos de treinamento/teste do experimento anterior (71 conjuntos), teriam que ser construídos portanto, um total de (4.950*71) classificadores binários.

O elevado número de classificadores a serem construídos, determinou uma redução no número de classes envolvidas para que o experimento fosse realizado no tempo disponível para conclusão desta dissertação. Optou-se, portanto, pela redução das 100 classes para apenas 10 classes de objetos, as quais geraram 45 classificadores binários para cada tamanho de conjunto de treinamento/teste (71 conjuntos). Foram escolhidas as 10 primeiras classes que compõem a COIL100. A Figura 4.11 apresenta os objetos utilizados.

Figura 4.11: 10 classes da COIL100 usadas no experimento 3.

O sistema de reconhecimento usando SVM foi implementado seguindo a mesma ordem do experimento anterior: pré-processamento, formação dos conjuntos de treinamento/teste, treinamento de classificadores e avaliação de desempenho. A diferença nesta etapa foi a

quantidade de classes envolvidas, apenas as 10 primeiras, e o tipo de kernel, polinomial quadrático, o qual apresentou os melhores resultados na etapa anterior. É importante destacar que parâmetros como: C, ϵ e γ, assumiram também os mesmos valores anteriores. Esses estágios não serão descritos nesta seção, uma vez que podem ser vistos na seção anterior. Esta seção apresentará o experimento utilizando Redes Neurais, os resultados obtidos e fará a comparação com os resultados obtidos com SVM.

O sistema de reconhecimento utilizando Redes Neurais pode ser dividido em cinco etapas: pré-processamento, definição da arquitetura, construção dos conjuntos de treinamento/teste, construção dos classificadores binários e avaliação de desempenho. A Figura 4.12 ilustra essas etapas.

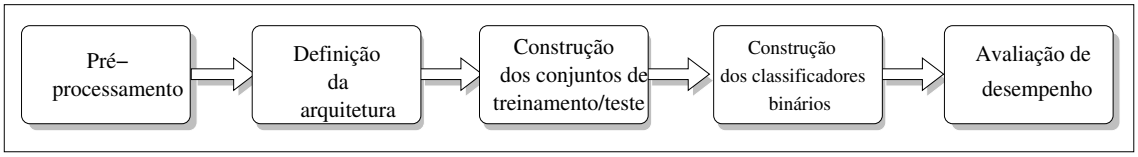

Figura 4.12: Arquitetura do experimento com Redes Neurais.

Pré-processamento

Esta etapa foi a mesma realizada no experimento anterior. As imagens foram convertidas para níveis de cinza e redimensionadas para uma retina de 32x32 pixels.

Definição da Arquitetura

Considerando que as imagens foram redimensionadas da forma acima, a primeira camada da rede foi constituída por 1024 neurônios. Como tratavam-se de classificadores binários, a camada de saída foi constituída por 2 neurônios. Utilizou-se apenas uma camada escondida. O número de neurônios na camada escondida foi determinado através de experimentos que objetivaram minimizar o número de ciclos necessários para a rede convergir. Os resultados ocorreram com 4 neurônios na camada escondida. A Tabela 4.7 sumariza a arquitetura da rede.

Camadas	**N^o de Neurônios**
Entrada	1024
Escondida	4
Saída	2

Tabela 4.7: Arquitetura das Redes Neurais.

Construção dos conjuntos de treinamento/teste

Foram construídos conjuntos de treinamento/teste de diferentes tamanhos, como nos experimentos anteriores. Todas as 72 visões das 10 classes foram utilizadas. Como na fase anterior, 71 conjuntos de treinamento/teste foram criados, com T amostras de treinamento e $(72 - T)$ amostras de teste por classe, com $T = 1, \ldots, 71$. As visões também foram selecionadas aleatoriamente.

Construção dos classificadores binários

Para cada valor de T, 45 classificadores binários foram construídos, manipulando as 10 classes envolvidas no problema. A cada iteração (mudança no valor de T) as visões aleatórias foram selecionadas. As visões escolhidas para cada classe, considerando um certo valor de T, mantiveram-se as mesmas para todos os classificadores. Por exemplo, as visões da classe 1 que constituíram o classificador binário 1-2, foram as mesmas que constituíram o classificador 1-3, e assim sucessivamente, para cada valor de T.

Todos os classificadores foram produzidos usando a mesma arquitetura descrita anteriormente. Portanto gerou-se 45 redes treinadas para cada valor de T. Ao final de cada iteração, a precisão do classificador foi avaliada.

Avaliação de desempenho

Para cada conjunto de treinamento com T visões, um conjunto de teste com (72-T) visões foi gerado. Os conjuntos de teste foram utilizados para avaliar a precisão alcançada pelas Redes Neurais.

É importante destacar que as mesmas visões usadas para treinar os classificadores neurais binários foram também treinadas com SVM. Os mesmos arquivos de teste foram utilizados

para a avaliação de desempenho de ambas as técnicas.

O processo envolvendo SVM foi o mesmo do experimento anterior. Quanto às Redes Neurais, os conjuntos de teste foram utilizados como padrões de entrada para as redes treinadas na fase anterior, ou seja, o conjunto de teste (72-T) foi submetido aos 45 classificadores gerados para T visões. Para verificar a precisão alcançada, a mesma estratégia de votação utilizada pela biblioteca LIBSVM foi implementada. Cada padrão que compunha o conjunto de teste foi analisado, verificando-se a quantidade de vezes (votos) que o padrão foi reconhecido como sendo de uma determinada classe. Ao final, quando o mesmo padrão foi apresentado a todos os 45 classificadores, verificou-se a classe que obteve o maior número de votos. Um acerto foi computado toda vez que a classe com o maior número de votos (vencedora) coincidiu com a classe real do padrão. Quando existiu 2 ou mais classes vencedoras, então foi considerada apenas a classe de menor índice como vencedora.

SVM apresentou uma pequena superioridade em relação às Redes Neurais. A Figura 4.13 mostra esses resultados através das curvas de reconheciemento obtidas a partir da média das 10 classes.

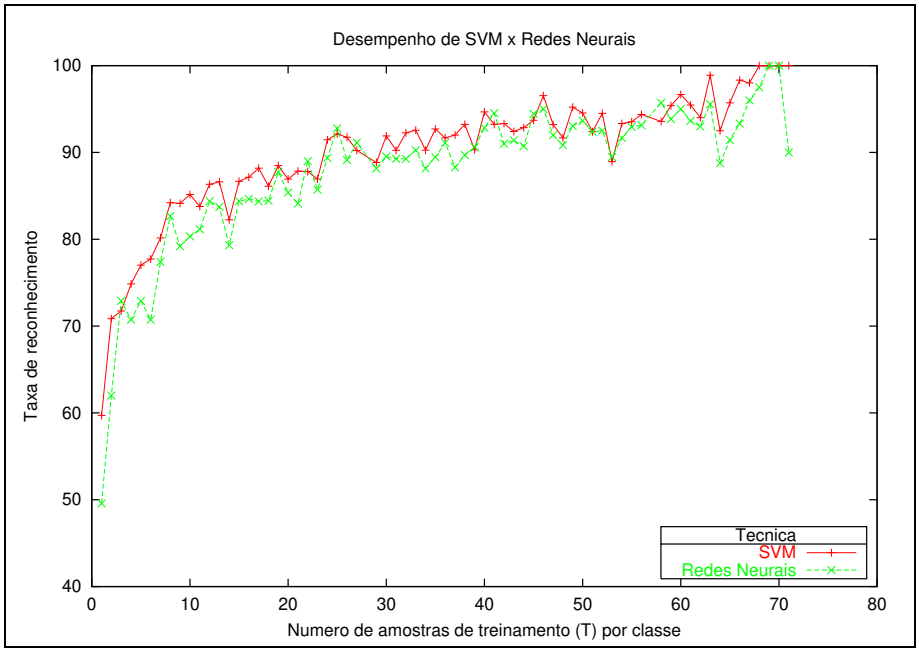

Figura 4.13: Curvas de reconhecimento para SVM e Redes Neurais, na base COIL100.

A taxa média de reconhecimento de Redes Neurais foi 87,79% e 89,98% para SVM. A melhor taxa de MLP foi 100% para T=69 e 100% para SVM, com T=68. A pior taxa foi

49,58% para MLP, com T=1 e 59,72% para SVM, com T=1. A Tabela 4.8 resume esses resultados.

Técnica	**Pior Taxa**	**Melhor Taxa**	**Taxa Média**
SVM	59,72% T=1	100% T=68	89,98%
MLP	49,58% T=1	100% T=69	87,79%

Tabela 4.8: Taxas de reconhecimento obtidas com Redes Neurais e SVM na base COIL100.

Os resultados obtidos mostraram que ambas as técnicas apresentam desempenho semelhante, com uma superioridade para SVM, no problema de reconhecimento de objetos baseado na aparência, usando as 10 primeiras classes de objetos da base de imagens COIL100. Além do desempenho, as técnicas apresentaram comportamento semelhante, à medida que a quantidade de visões aumentou, as taxas de reconhecimento aumentaram também. Ambas alcançaram 100% de precisão antes de utilizar as 71 visões para treinamento.

4.4 Conclusão

Os experimentos descritos neste capítulo objetivaram reforçar a compreensão da técnica SVM, reforçar resultados obtidos na literatura recente da técnica e realizar um estudo experimental comparativo entre SVM e Redes Neurais MLP *Backpropagation*, no problema de reconhecimento de objetos baseado na aparência. Os resultados do primeiro experimento deste capítulo foram publicados em [dos Santos and Gomes, 2001]. Os resultados do segundo experimento foram aceitos para publicação em [dos Santos and Gomes, 2002]. Um trabalho descrevendo todos os capítulos desta dissertação está sendo submentido para o I Workshop de Teses e Dissertações em Inteligência Artificial. O próximo capítulo apresenta as conclusões gerais desta dissertação e as perspectivas para trabalhos futuros.

Capítulo 5

Conclusões

Esta dissertação apresentou a técnica de aprendizagem de máquina SVM como uma opção para realizar a tarefa de reconhecimento de objetos baseado na aparência. Investigou o desempenho da técnica na solução desse tipo de problema, comparou três tipos de SVM entre si e comparou SVM com outra técnica mais clássica, Redes Neurais.

Além da parte experimental discutida acima, esta dissertação também procurou apresentar a teoria que formaliza SVM. O Capítulo 2 descreve o esquema matemático que envolve as etapas de treinamento e teste, especificamente para a área de Reconhecimento de Padrões. É importante ressaltar que a teoria fundamental para a compreensão de SVM envolve duas áreas de pesquisa, Teoria da Aprendizagem Estatística e Otimização Matemática, as quais não são geralmente parte dos currículos de Cientistas da Computação. Portanto, nesta dissertação procurou-se apresentar SVM utilizando uma linguagem mais didática e acessível.

5.1 Sumário da Dissertação

O Capítulo 2 apresentou as duas teorias fundamentais de SVM. Sem esse conhecimento seria difícil assimilar como o treinamento de SVM é realizado (através da solução de um problema quadrático convexo) e como os vetores de suporte são extraídos (considerando-se a margem de separação máxima). Os conceitos acima são igualmente importantes, para uma melhor compreenção da fase de teste, quando os vetores de suporte são utilizados como parâmetros que determinam a que classe um padrão não conhecido pertence.

Um outro conceito importante para SVM é a função kernel. Os métodos de aprendi-

zagem basedos em kernel estão ganhando muito espaço nas conferências e periódicos de aprendizagem de máquina. Esse crescimento deve-se especialmente, ao desenvolvimento das pesquisas com SVM. Constitui uma outra área de pesquisa em separado e envolve outras metodologias de aprendizagem de máquina como, Redes Neurais, PCA, Funções de Base Radial, etc.

Apesar da complexidade conceitual, o Capítulo 3 procurou relacionar as principais linhas de pesquisa na área de SVM. O objetivo do capítulo foi apresentar os conceitos de forma clara, evitando muitos termos técnicos, mas sem negligenciar os conceitos fundamentais. Essa meta foi proposta para apresentar um conteúdo mais compreensível, o que não acontece na literatura típica de SVM.

Trata-se portanto, de uma área de pesquisa que está em franco desenvolvimento, restrita ainda a poucos grupos, porém já alcançando um público maior. Como a teoria é totalmente formalizada, embora ainda haja muito a ser explorado, o enfoque da maioria dos trabalhos são as aplicações. Busca-se um dimensionamento do poder de abrangência de SVM em problemas práticos. Um dos problemas investigados é o reconhecimento de objetos baseado na aparência, que pode ser útil na maioria das tarefas que envolvem navegação, Robótica, sistemas de segurança, entre outras tarefas em que a Visão de Máquina é requerida.

Esta dissertação, além de apresentar a descrição conceitual, também investigou o comportamento de SVM no problema prático de reconhecimento baseado na aparência. Para a realização de um estudo prático, descrito no Capítulo 4, foram utilizadas bibliotecas e bases de imagens disponíveis ao público, para pesquisas com fins não comerciais.

Os experimentos foram divididos em três grupos, utilizando uma estratégia de variar o tamanho dos conjuntos de treinamento e teste. A primeira etapa de experimentos avaliou o comportamento e a precisão de SVM no problema de classificação de 10 classes de objetos, com 20 visões de cada. O kernel utilizado foi o polinomial quadrático. A taxa média de reconhecimento foi 90%. Os resultados obtidos confirmaram que SVM é aplicável ao problema, têm seu desempenho de reconhecimento aumentado quando treinada com muitas visões e que se comporta como a maioria das demais técnicas.

O segundo experimento constou de um estudo comparativo entre três tipos de kernel polinomial: linear, quadrático e cúbico, na base de imagens COIL100. Os resultados obtidos reforçaram outros resultados da literatura de que o tipo de kernel normalmente não interfere

muito na precisão final do classificador. Os três kernels pesquisados apresentaram resultados muito próximos, com uma pequena superioridade para o quadrático.

O terceiro experimento envolveu um estudo experimental comparativo entre SVM e Redes Neurais do tipo *Multilayer Perceptron Backpropagation*, para a verificação do desempenho de ambas as técnicas, considerando-se apenas 10 classes de objetos da base de imagens COIL100. SVM apresentou uma pequena superioridade em média de (2,19%) em relação à Redes Neurais, o que reforça que SVM é uma técnica totalmente viável para problemas de classificação de padrões, em especial para problemas de reconhecimento baseado na aparência.

Todos esses experimentos contribuíram para uma compreensão mais prática da técnica, reforçando o conhecimento teórico e confirmaram a aplicabilidade de SVM ao problema de reconhecimento de objetos 3D, a partir de visões 2D.

5.2 Perspectivas de Trabalhos Futuros

Algumas sugestões de trabalhos futuros são apresentadas a seguir:

- Realizar experimentos para avaliação de SVM em comparação com Redes Neurais, utilizando as 100 classes da base de imagens COIL100. O que não foi possível ser realizado nesta dissertação por limitações de tempo.

- Realizar experimentos variando apenas o tamanho dos conjuntos de treinamento e mantendo um único conjunto de teste.

- Testar extensões de SVM. Conforme foi citado, já foram propostas modificações para o algoritmo padrão. Uma possibilidade seria realizar um estudo comparativo entre esses algoritmos para determinar o mais indicado ao problema.

- Estudo comparativo entre metodologias de classificação multi-classes. A maioria das estratégias já foram objeto de experimentos, porém metodologias mais recentes como por exemplo [Weston and Watkins, 1999] indicam a possibilidade de adicionar a representação multi-classes diretamente ao problema de otimização resolvido em SVM. Portanto, um possível trabalho poderia ser um estudo comparativo entre essas metodologias.

- Estudo de implementação com processadores dedicados em que não seja necessário diminuir a dimensão das imagens das bases de dados.

- Uma outra perspectiva de trabalho futuro seria estender o domínio de aplicação de SVM. Esta dissertação restringiu-se ao domínio de Reconhecimento de Padrões, mas SVM pode ser aplicado a outros tipos de problemas como Regressão e Detecção de Novidades. As formulações de SVM sofrem algumas mudanças dependendo do tipo de problema. O que não foi explorado nesta dissertação.

- Divulgar a contribuição teórica através de tutorial, artigo etc, com o intuito de transferir o conhecimento sistematizado nesta dissertação.

Bibliografia

[Barabino et al., 1999] Barabino, N., Pallavicini, M., et al. (1999). Support vector machines vs multi-layer perceptrons in particle identification. In *Proceedings of European Symposium on Artificial Neural Network*.

[Brown et al., 2000] Brown, M. P. S., Grundy, W., et al. (2000). Knowledge-based analysis of microarray gene expression data using support vector machines. *National Academy of Sciences*, (1):262–267.

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, pages 243–245.

[Campbell, 2000] Campbell, C. (2000). Kernel methods: A survey of current techniques. http://citeseer.nj.nec.com/campbell00kernel.html.

[Chang et al., 2000] Chang, C.-C., Hsu, C.-W., and Lin, C.-J. (2000). The analysis of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 11(4):1003–1008.

[Chang and Lin, 2001a] Chang, C.-C. and Lin, C.-J. (2001a). Libsvm: a library for support vector machines (version 2.31). http://citeseer.nj.nec.com/chang01libsvm.html.

[Chang and Lin, 2001b] Chang, C.-C. and Lin, C.-J. (2001b). Training u-support vector classifiers: Theory and algorithms. *Neural Computation*, (9):2119–2147.

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

[Correia, 2000] Correia, S. E. N. (2000). Reconhecimento de caracteres numéricos manuscritos usando transoformada de wavelet. Dissertação de mestrado, COPELE-Universidade Federal da Paraíba.

[Cortez, 1996] Cortez, P. C. (1996). Reconhecimento de formas 2d usando uma técnica sequencial integrada e modelos poligonais. Dissertação de mestrado, COPELE-Universidade Federal da Paraíba.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

[da Silva, 1999] da Silva, V. C. (1999). Identificação de vértices em imagens multi-vista de formas 3d. Dissertação de mestrado, COPIN-Universidade Federal da Paraíba.

[de Alcântara Morais, 1998] de Alcântara Morais, M. R. (1998). Sistema de prototipagem rápida dirigida ao processamento de imagens. Dissertação de mestrado, COPELE-Universidade Federal da Paraíba.

[de Arruda, 1997] de Arruda, A. E. M. (1997). Avaliação de métodos de limiarização para segmentação de imagens de tomografia do fígado em oncologia. Dissertação de mestrado, COPELE-Universidade Federal da Paraíba.

[de Fátima Santos Farias, 1997] de Fátima Santos Farias, M. (1997). Reconhecimento de objetos 3d através de atributos de superfícies de modelos poliedrais. Dissertação de mestrado, COPELE-Universidade Federal da Paraíba.

[dos Santos and Gomes, 2001] dos Santos, E. M. and Gomes, H. M. (2001). Appearance based object recognition using support vector machines. In *XIV Brazilian Symposium on Computer Graphics and Image Processing*, page 399.

[dos Santos and Gomes, 2002] dos Santos, E. M. and Gomes, H. M. (2002). A comparative study of polynomial kernel svm applied to appearance based object recognition. A ser publicado em Proceedings of the International Workshop on Pattern Recognition with Support Vector Machines, Lecture Notes in Computer Science, Windsor, Canadá.

[Duda et al., 2000] Duda, R. O., Hart, P. E., and Stork, D. (2000). *Pattern Classification*. Wiley-Interscience, 2nd edition.

[Dumais et al., 1998] Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *In Proceedings of ACM 7th International Conference on Iformation and Knowledge Management*, pages 148–155.

[Eghbalnia and Assadi, 2001] Eghbalnia, H. and Assadi, A. (2001). An application of support vector machines and symmetry to computational modeling of perception through visual attention. *Neurocomputing*, (40):1193–1201.

[Evgeniou et al., 1999] Evgeniou, T., Pontil, M., and Poggio, T. (1999). A unified framework for regularization networks and support vector machines. Technical Report AIM-1654.

[Franklin, 1995] Franklin, S. P. (1995). *Artificial Minds*. MIT Press.

[Gill et al., 1981] Gill, P., Murray, W., and Wright, M. H. (1981). *Practical Optimization*. Academic Press.

[Ginsberg, 1993] Ginsberg, M. (1993). *Essentials of Artificial Intelligence*. Morgan Kaufmann Publishers.

[Gonzalez, 1993] Gonzalez, R. (1993). *Digital Image Processing*. Addison-Wesley Publising Company.

[Gottfried and Weisman, 1973] Gottfried, B. S. and Weisman, J. (1973). *Introduction to Optimization*. Series in Industrial and Systems Engineering. Prentice-Hall.

[Grove and Fisher, 1996] Grove, T. and Fisher, R. (1996). Attention in iconic object matching. In *Proceedings of British Machine Vision Conference*, volume 1, pages 293–302, Edinburgh, Scotland.

[Gunn, 1998] Gunn, S. (1998). Support vector machines for classification and regression. Technical Report ISIS-1-98, Department of Electronics and Computer Science, University of Southampton.

[Hadzic and Kecman, 2000] Hadzic, I. and Kecman, V. (2000). Support vector machines trained by linear programming. theory and application in image compression and data classification. In *Proceedings of Neurel 2000-IEEE Fifth Seminar on Neural Network Applications in Electrical Engineering*, pages 18–23.

[Haykin, 1999] Haykin, S. (1999). *Neural Networks a Comprehensive Foundation*. Prentice Hall.

[Hearst, 1998] Hearst, M. A. (1998). Support vector machines. *IEEE Intelligent Systems*, pages 18–28.

[Jaakkola and Haussler, 1998] Jaakkola, T. and Haussler, D. (1998). Exploiting generative models in discriminative classifiers. In Kearns, M., Solla, S., and Cohn, D., editors, *Advances in Neural Information Processing Systems 11*. MIT Press.

[Joachims, 1998] Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In *Proceedings of 10th European Conference on Machine Learning*, number 1398, pages 137–142. Springer Verlag, Heidelberg, DE.

[Joachims, 1999] Joachims, T. (1999). Making large-scale svm learning practical. In B. Schölkopf, C. B. e. A. J. S., editor, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press.

[Jonsson et al., 1999] Jonsson, K., Kittler, J., Li, Y., and Matas, J. (1999). Support vector machines for face authentication. In *Proceedings British Machine Vision Conference (BMVC'99)*, pages 543–553.

[Kwok, 1998] Kwok, J. T.-Y. (1998). Support vector mixture for classification and regression problems. In *International Conference on Pattern Recognition (ICPR)*, pages 255–258.

[Kwong and Gong, 1999] Kwong, J. N. S. and Gong, S. (1999). Learning support vector machines for a multi-view face model. In *Proceedings of British Machine Vision Conference, September 1999.*, pages 503–512, Nottingham, UK.

[Lasdon, 1970] Lasdon, L. S. (1970). *Optimization Theory for Large Systems*. Macmillan Publishing.

[Marr, 1982] Marr, D. (1982). *Vision*. W.H. Freeman & Company.

[Matter and Haykin, 1999] Matter, D. and Haykin, S. (1999). Support vector machines for dynamic reconstruction of chaotic system. In B. Schölkopf, C. B. e. A. J. S., editor, *Advances in Kernel Methods - Support Vector Learning*, pages 211–241. MIT Press.

[Mattera et al., 1999] Mattera, D., Palmieri, F., and Haykin, S. (1999). An explicit algorithm for training support vector machines. *IEEE Signal Processing Letters*, 6(9):243–245.

[M.Gomes et al., 1998] M.Gomes, H., Fisher, R. B., and Hallan, J. (1998). A retina-like image representation of primal sketch features extracted using neural network approach. In *Proceedings Noblesse Workshop on Non-Linear Model Based Image Analysis*, pages 251–256.

[Müller et al., 2001] Müller, K.-R., Mika, S., et al. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.

[Murase and Nayar, 1995] Murase, H. and Nayar, S. K. (1995). Visual learning and recognition of 3-d object from appearance. *Computer Vision*, pages 5–24.

[Murtagh and Saunders, 1993] Murtagh, B. A. and Saunders, M. A. (1993). Minos 5.4 user's guide. Technical Report SOL 83.20.

[Nayar et al., 1996] Nayar, S. K., Nene, S. A., and Murase, H. (1996). Real-time 100 objects recognition system. In *Proceedings of ARPA Image Understanding Workshop*.

[Osuna et al., 1997a] Osuna, E., Freund, R., and Girosi, F. (1997a). Support vector machines: Training and applications. Technical Report AIM-1602.

[Osuna et al., 1997b] Osuna, E., Freund, R., and Girosi, F. (1997b). Training support vector machines:an application to face detection. In *Proceedings Conference on Computer Vision and Pattern Recognition (CVPR 1997)*.

[Osuna and Girosi, 1999] Osuna, E. and Girosi, F. (1999). Reducing the run-time complexity in support vector machines. In B. Schölkopf, C. B. e. A. J. S., editor, *Advances in Kernel Methods - Support Vector Learning*, pages 271–283. MIT Press.

[Penrose, 1995] Penrose, R. (1995). *Shadows of the mind*. Vintage.

[Platt, 1999] Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. B. e. A. J. S., editor, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press.

[Pontil and Verri, 1998] Pontil, M. and Verri, A. (1998). Support vector machines for 3-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(February):637–646.

[Ribeiro, 2001] Ribeiro, S. F. (2001). Um sistema de visão inteligente para detecção e reconhecimento e peças em um tabuleiro de xadrez em tempo real. Dissertação de mestrado, COPIN-Universidade Federal da Paraíba.

[Rich and Knight, 1991] Rich, E. and Knight, K. (1991). *Artificial Intelligence*. McGraw-Hill.

[Roobaert, 1999] Roobaert, D. (1999). Improving the generalization of linear support vector machines: an application to 3d object recognition with cluttered background. In *Proceedings of SVM Workshop at the 16th International Joint Conference on Artificial Intelligence*.

[Roobaert, 2000] Roobaert, D. (2000). Directsvm: a simple support vector machine perceptron. In *IEEE Int. Workshop Neural Networks for Signal Processing*, pages 356–365.

[Roobaert et al., 2000] Roobaert, D., Nillius, P., and Eklundh, J.-O. (2000). Comparison of learning approaches to appearance-based 3d object recognition with and without cluttered background. In *Proceedings of the Fourth Asian Conference on Computer Vision (ACCV00*, pages 443–448.

[Roth et al., 2000] Roth, D., Yang, M.-H., and Ahuja, N. (2000). Learning to recognize objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–731.

[Russel and Norvig, 1995] Russel, S. and Norvig, P. (1995). *Artificial Intelligence, a modern approach*. Series in Artificial Intelligence. Prentice Hall.

[Schölkop et al., 1999] Schölkop, B., Smola, A. J., and Müller, K. R. (1999). Kernel principal component analysis. In B. Schölkopf, C. B. e. A. J. S., editor, *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press.

[Schölkopf, 1997] Schölkopf, B. (1997). *Support Vector Learning*. PhD thesis, Universidade de Berlin.

[Schölkopf et al., 1999a] Schölkopf, B., Burges, C., and Smola, A. (1999a). *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

[Schölkopf et al., 1999b] Schölkopf, B., J.Platt, et al. (1999b). Estimating the support of a high-dimensional distribution. Technical Report Microsof Research 99-87.

[Schwenker, 2000] Schwenker, F. (2000). Hierarchical support vector machines for multiclass pattern recognition. In *Proceedings Fourth International Conference on Knowledge-Based Intelligent Engineering Systems, September 2000*, pages 561–565, Brighton, UK.

[Stitson et al., 1996] Stitson, M., Weston, J., Gammerman, A., et al. (1996). Theory of support vector machines. Technical Report CSD-TR-96-17, Computational Intelligence Group, Royal Holloway, University of London.

[Suykens and Vandewalle, 1999] Suykens, J. and Vandewalle, J. (1999). Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks*, (4):907–911.

[Tipping, 2000] Tipping, M. (2000). The relevance vector machine. In *Advances in Neural Information Processing Systems*. MIT Press.

[Vanderbei, 1997] Vanderbei, R. J. (1997). Loqo 3.10 user's manual. Technical Report SOR-97-08.

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

[Vapnik, 1999] Vapnik, V. N. (1999). *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 2nd edition.

[Veloso, 1998] Veloso, L. R. (1998). Reconhecimento de caracteres numéricos manuscritos. Dissertação de mestrado, COPELE-Universidade Federal da Paraíba.

[Weston and Watkins, 1999] Weston, J. and Watkins, C. (1999). Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*.

[Weston, 1999] Weston, J. A. E. (1999). *Extensions to the Support Vector Method*. PhD thesis, Royal Holloway University of London.

[Yang and Ahuja, 2000] Yang, M.-H. and Ahuja, N. (2000). A geometric approach to train support vector machines. In *Proceedings Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, pages 430–437.

[Z. Li and Licheng, 2000] Z. Li, Z. W. and Licheng, J. (2000). Pre-extracting support vectors for support vector machine. In *Proceedings of the 5th International Conference on Signal Processing*, pages 1432–1435.