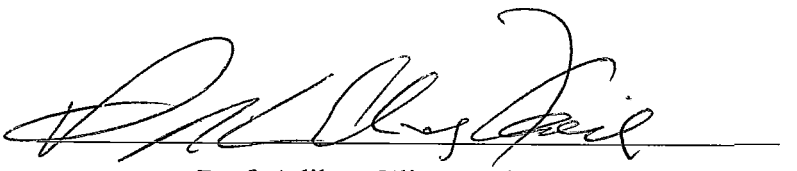
ALGORITMO DE CLASSIFICAÇÃO MÁQUINA DE VETORES SUPORTE VIA SUAVIZAÇÃO HIPERBÓLICA

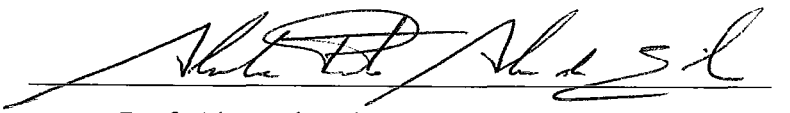
Victor Ströele de Andrade Menezes

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

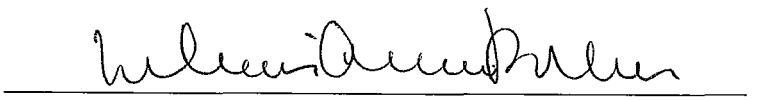
Aprovada por:



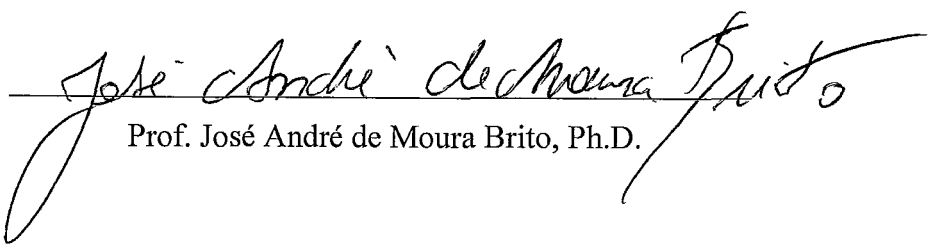
Prof. Adilson Elias Xavier, D.Sc.



Prof. Alexandre Pinto Alves da Silva, Ph.D.



Prof. Valmir Carneiro Barbosa, Ph.D.



Prof. José André de Moura Brito, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 2007

MENEZES, VICTOR STRÖELE DE ANDRADE

Algoritmo de Classificação Máquina de Vetores Suporte via Suavização Hiperbólica [Rio de Janeiro] 2007

X, 85 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2007)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Classificação
2. Suavização Hiperbólica
3. SVM

I. COPPE/UFRJ II. Título (série)

Dedicatória

Dedico esta tese aos meus pais,
à minha esposa Patrícia,
por todo carinho, apoio e compreensão
durante minha presença *ausente*,
necessária para a realização
desse trabalho.

Agradecimentos

A Deus, por me guiar durante esse caminho.

Ao professor Adilson Elias Xavier pela amizade, orientação e total incentivo durante o desenvolvimento deste trabalho. Por sua inteira disposição e paciência para transmitir todo o conhecimento necessário para o desenvolvimento desta dissertação de mestrado, já que sem seu apoio não seria possível a conclusão da mesma.

Aos meus pais, que estão sempre prontos para me ouvir e me aconselhar nos momentos de indecisão.

Ao meu irmão, que, mesmo distante, sempre torceu por mim e me apoiou durante as minhas decisões.

À minha avó, por todo seu carinho e afeto.

À minha esposa Patrícia que sempre me acalmou nas horas de insegurança e ajudou de maneira ativa no desenvolvimento deste trabalho.

Aos amigos: Luiz Felipe, Tato e Melissa que sempre me apoiaram e apoiarão em todos os momentos da minha vida.

Aos professores Alexandre, André e Valmir, pelo aceite imediato de participação na Banca de Defesa de Tese.

À CAPES pelo fomento da pesquisa por meio da bolsa de Mestrado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMO DE CLASSIFICAÇÃO MÁQUINA DE VETORES SUPORTE VIA SUAVIZAÇÃO HIPERBÓLICA

Victor Ströele de Andrade Menezes

Dezembro/2007

Orientador: Adilson Elias Xavier

Programa: Engenharia de Sistemas e Computação

Este trabalho é destinado a mostrar uma nova abordagem para a resolução do problema linear de Máquina de Vetores Suporte (SVM) O modelo matemático considerado conduz a uma formulação que tem uma característica significante, de ser não-diferenciável. Para superar esta dificuldade, o método de resolução proposto adota uma estratégia de suavização usando uma função suavizadora especial pertencente à classe de funções C^{∞}.

A solução final é obtida através da resolução de uma seqüência de subproblemas de otimização diferenciáveis irrestritos, definidos em um espaço com dimensão pequena, que gradativamente se aproximam do problema original. A utilização dessa técnica, denominada Suavização Hiperbólica, permite superar as principais dificuldades presentes no problema original. Um algoritmo simplificado contendo somente o essencial do método é apresentado. Com o objetivo de ilustrar tanto a confiabilidade e eficiência do método, realizou-se um conjunto de experimentos computacionais para problemas teste padrão existentes na literatura.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SUPPORT VECTOR MACHINE CLASSIFICATION ALGORITHM BY HYPERBOLIC SMOOTHING APPROACH

Victor Ströele de Andrade Menezes

December/2007

Advisor: Adilson Elias Xavier

Department: Systems Engineering and Computer

This work is intended to show a new approach for solving the linear Support Vector Machine (SVM) problem. The mathematical modeling of this problem leads to a formulation which has the significant characteristic of being non-differentiable. In order to overcome these difficulties, the resolution method proposed adopts a smoothing strategy using a special C^{∞} differentiable class function.

The final solution is obtained by solving a sequence of low dimension differentiable unconstrained optimization subproblems which gradually approach the original problem. The use of this technique, called Hyperbolic Smoothing, allows the maim difficulties presented by the original problem to be overcome. A simplified algorithm containing only the essential of the method is presented. For the purpose of illustrating both the reliability and the efficiency of the method, a set of computational experiments was performed making use of traditional test problems described in the literature.

Sumário

DEDICATÓRIA III

AGRADECIMENTOS IV

SUMÁRIO VII

ÍNDICE DE FIGURAS IX

ÍNDICE DE TABELAS X

CAPÍTULO 1 – INTRODUÇÃO 1

1.1 – INTRODUÇÃO 1

1.2 – PROBLEMAS DE CLASSIFICAÇÃO 3

1.2.1 – Classificação de Texto *3*

1.2.2 – Análise de Seqüências biológicas *3*

1.2.3 – Diagnóstico de doenças *5*

1.3 – SEQÜÊNCIA DA APRESENTAÇÃO 6

CAPÍTULO 2 – TÉCNICAS DE ORGANIZAÇÃO DE DADOS 9

2.1 – INTRODUÇÃO 9

2.2 – TÉCNICAS DE CLASSIFICAÇÃO 9

2.3 – *BOOSTING* 10

2.3.1 – AdaBoost *12*

2.4 – ANÁLISE *CLUSTER* (AGRUPAMENTO) 13

2.4.1 – Algoritmos Hierárquicos *14*

2.4.2 – Algoritmos não Hierárquicos *15*

2.4.3 – Formulação Matemática *16*

2.5 – MÁQUINA DE VETOR SUPORTE (SVM) 17

2.5.1 – Descrição *18*

2.5.2 – Formulação *19*

2.5.3 – Condições de existência das funções Kernel *23*

CAPÍTULO 3 – SUAVIZAÇÃO HIPERBÓLICA 25

3.1 – INTRODUÇÃO 25

3.2 – DEFINIÇÃO DO PROBLEMA 25

3.2.1 – Formulação SVM de Mangasarian *26*
3.2.2 – Formulação SVM adotada *28*
3.3 – TRANSFORMAÇÃO DO PROBLEMA 31
3.4 – SUAVIZAÇÃO DO PROBLEMA 33
3.5 – RESOLUÇÃO DO PROBLEMA 35
3.6 – PENALIZAÇÃO HIPERBÓLICA 37
3.7 – ALGORITMO SHSVM SIMPLIFICADO 41
CAPÍTULO 4 – IMPLEMENTAÇÃO E RESULTADOS COMPUTACIONAIS 43
4.1 – IMPLEMENTAÇÃO 43
4.2 – SVMLIGHT 48
4.2.1 – Decomposição e Redução do Conjunto de Treinamento *49*
4.3 – RESULTADOS COMPUTACIONAIS 51
4.3.1 – Problemas não linearmente separáveis *53*
4.3.2 – Problemas linearmente separáveis *54*
4.4 – SÍNTESE DOS RESULTADOS 57
CAPÍTULO 5 – PODA 59
5.1 – CRITÉRIO PARA SELEÇÃO DE DADOS 59
5.1.1 – Cálculo da Variação Mínima para pontos não suporte *60*
5.1.2 – Cálculo da Variação Máxima para Pontos Suporte *62*
5.2 – ESPECIFICAÇÃO INICIAL DO ÂNGULO DE PODA α 63
5.3 – CRITÉRIOS ADICIONAIS DO PROCESSO DE PODA 64
5.4 – PODA POR SIMILARIDADE 67
5.4.1 – Análise da Poda por Similaridade *72*
5.5 – RESULTADOS 73
5.6 – SÍNTESE DOS RESULTADOS DOS PROCEDIMENTOS DE PODA 77
CAPÍTULO 6 – CONCLUSÃO 79
6.1 – PROPOSTAS PARA TRABALHOS FUTUROS 80
APÊNDICE A 81
REFERÊNCIAS 83

Índice de Figuras

FIGURA 1.1 - REPRESENTAÇÃO DA ETAPA DE APRENDIZADO DE UM CLASSIFICADOR..... 2

FIGURA 1.2 - APLICAÇÃO DE UMA AMOSTRA DE DNA A UM CLASSIFICADOR..... 4

FIGURA 2.1 - AGRUPAMENTO HIERÁRQUICO AGLOMERATIVO E PARTICIONADO 14

FIGURA 2.2 - HIPERPLANOS SEPARADORES PARA DOIS CONJUNTOS DE DADOS 17

FIGURA 2.3 - SVM LINEAR..... 18

FIGURA 2.4 - DEFINIÇÃO DA MÁXIMA MARGEM..... 20

FIGURA 3.1 - PROBLEMA NÃO LINEARMENTE SEPARÁVEL 27

FIGURA 3.2 - DISTÂNCIA MÍNIMA ENTRE A CLASSE E O HIPERPLANO 30

FIGURA 3.3 - SOLUÇÃO DO PROBLEMA NÃO LINEARMENTE SEPARÁVEL ($C=10^{-8}$)..... 31

FIGURA 3.4 - SOMATÓRIO DA LADO DIREITO DAS RESTRIÇÕES (3.17)..... 32

FIGURA 3.5 - SOMATÓRIO ORIGINAL E SUAVIZADO DAS RESTRIÇÕES DO PROBLEMA..... 34

FIGURA 3.6 – FUNÇÃO DE PENALIZAÇÃO HIPERBÓLICA 38

FIGURA 3.7 - VARIAÇÃO DO PARÂMETRO λ 39

FIGURA 3.8 - VARIAÇÃO DO PARÂMETRO ρ 40

FIGURA 4.1 - CENTROS DE GRAVIDADE E HIPERPLANO INICIAL..... 43

FIGURA 4.2 – PARÂMETRO τ 44

FIGURA 4.3 – PARÂMETRO ε 45

FIGURA 4.4 - SOLUÇÃO GRÁFICA SHSVM E SVMLIGHT 52

FIGURA 5.1 - MENOR DISTÂNCIA POSSÍVEL PARA $x_j, j \notin J^*$ 60

FIGURA 5.2 - MENOR DISTÂNCIA POSSÍVEL PARA $x_i, i \in J^*$ 62

FIGURA 5.3 - VARIAÇÃO DO ÂNGULO α ENTRE DUAS DIREÇÕES CONSECUTIVAS..... 65

FIGURA 5.4 - VARIAÇÃO ANGULAR DENTRO DO CONE 66

FIGURA 5.5 - VARIAÇÃO ANGULAR FORA DO CONE..... 66

FIGURA 5.6 - COMPORTAMENTO TÍPICO DA SEQÜÊNCIA DE CONES (A);..... 67

FIGURA 5.7 - OBSERVAÇÕES AGLOMERADAS..... 68

FIGURA 5.8 - OBSERVAÇÕES AGLOMERADAS NÃO PODADAS 69

FIGURA 5.9 - SOLUÇÃO DO PROBLEMA SEM AS OBSERVAÇÕES AGLOMERADAS (ESQUERDA) E SOLUÇÃO DO PROBLEMA COM A OBSERVAÇÃO QUE VIOLA A MARGEM (DIREITA).. 70

FIGURA 5.10 - ANÁLISE DO COMPORTAMENTO DA PODA POR SIMILARIDADE 72

Índice de Tabelas

TABELA 4.1 - VARIAÇÃO DOS PARÂMETROS 47
TABELA 4.2 - COMPARAÇÃO DOS RESULTADOS ENTRE SVMLIGHT E SHSVM 52
TABELA 4.3 - COMPARAÇÃO ENTRE AS MARGENS 53
TABELA 4.4 - COMPARAÇÃO DOS RESULTADOS PARA BASE NÃO LINEARMENTE SEPARÁVEL 53
TABELA 4.5 - SHSVM COMPARADO AO SVMLIGHT 55
TABELA 4.6 - COMPARAÇÃO DOS RESULTADOS PARA BASES ALEATÓRIAS 57
TABELA 5.1 - COMPARAÇÃO DE RESULTADOS ENTRE OS CLASSIFICADORES SHSVM, ... 74
TABELA 5.2 - COMPARAÇÃO DE RESULTADOS ENTRE OS CLASSIFICADORES SHSVM, SVMLIGHT, SHSVMPODA 74
TABELA 5.3 - COMPARAÇÃO DE RESULTADOS PARA BASES ALEATÓRIAS COM DADOS AGLOMERADOS 75
TABELA 5.4 - COMPARAÇÃO DE RESULTADOS PARA BASES ALEATÓRIAS COM DADOS ESPARSOS 76

Capítulo 1 – Introdução

1.1 – Introdução

Um problema muito comum nos dias atuais está relacionado à tomada de decisão em ambientes cercados de incertezas e de imprecisões. O ser humano, ao se ver cercado de alternativas e opções, não é capaz de tomar uma decisão com a certeza de que ela seja a mais correta. Isso ocorre por ele não ser capaz de analisar todos os dados que estão ao seu redor. Para evitar enganos que podem até mesmo ser fatais em alguns casos, esses problemas têm sido resolvidos utilizando-se técnicas que empregam, sobretudo, o conceito de aprendizado. O aprendizado se dá a partir de dados experimentais ou da experiência do agente com o ambiente no qual o problema está inserido. Assim, é desenvolvido um sistema, a partir de um conjunto de dados, denominado conjunto de treinamento, capaz de extrair informações mais precisas sobre o problema.

Neste contexto, procura-se o desenvolvimento de técnicas e algoritmos no sentido de solucionar problemas relacionados à identificação, classificação e predição e controle de sistemas adaptativos. Esse algoritmo é desenvolvido através de um processo contínuo de treinamento, considerando a existência de um conjunto de dados ou informações do ambiente do problema.

Particularmente, em problemas de classificação relacionados a um conjunto de treinamento, há um processo inicial, onde se busca inferir uma hipótese caracterizada pelo projeto de um classificador, que possui um conjunto de parâmetros. Essa hipótese é construída representando os dados do conjunto de entrada em vetores e treinando o classificador para que ele classifique corretamente todos, ou a maioria dos dados do conjunto de treinamento. Posteriormente, o classificador pode estabelecer uma categorização para uma nova amostra, desde que esta seja representada de forma vetorial como os dados de entrada.

A Figura 1.1 ilustra a etapa de aprendizado do classificador. Nesse exemplo, pode ser observado que existe um conjunto de classes distintas do mundo real. O objetivo é representar cada uma dessas classes de forma vetorial, passar por uma etapa de treinamento e construir o classificador. A etapa de treinamento tem a função de

"ensinar" o classificador para que ele "aprenda" sobre as características de cada classe do conjunto de entrada.

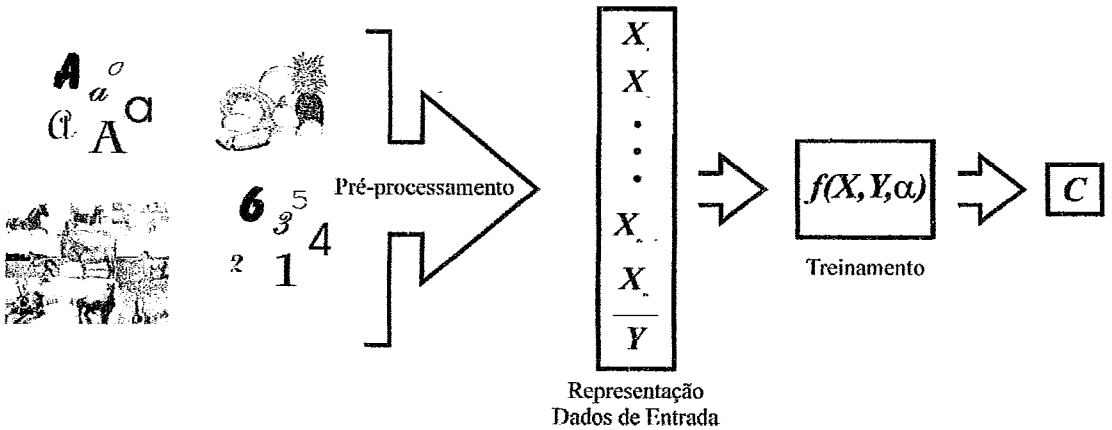


Figura 1.1 - Representação da etapa de aprendizado de um classificador.

Na maioria das vezes, esses classificadores são treinados para atribuir os dados a duas classes diferentes. Assim, ele é representado por uma função discriminante que será um hiperplano, quando os dados forem linearmente separáveis, ou uma outra função, para dados não linearmente separáveis.

Outra questão importante na resolução de problemas de classificação é o tamanho do conjunto de treinamento do classificador. Alguns problemas possuem muitos dados de entrada, o que torna o processo de treinamento extremamente lento.

Para driblar esse problema do tamanho do conjunto de treinamento, a maioria dos softwares desenvolvidos para solucionar problemas de classificação utilizam técnicas de redução do conjunto de treinamento, a fim de melhorar a eficiência do software.

Uma das estratégias de maior sucesso no equacionamento de problemas de classificação é a denominada Máquina de Vetor Suporte, mais conhecida por denominação em inglês *Support Vector Machine* (SVM) (VAPNIK, 1995).

Um dos softwares de mais amplo uso na resolução de problemas de classificação, segundo a abordagem SVM, é o *SVMLight*. No decorrer do aprimoramento desse software, também foram desenvolvidas técnicas para redução do conjunto de treinamento, de forma a reduzir o tempo de processamento gasto pelo classificador durante a fase de treinamento. Assim, esse software tornou-se um dos mais eficientes na resolução de problemas SVM dentre os softwares conhecidos na literatura.

Neste trabalho será apresentada uma técnica, segundo uma nova metodologia, para resolução do problema de classificação, que igualmente inclui estratégias para a redução do conjunto de treinamento, de forma a melhorar a eficiência do classificador na obtenção de soluções.

1.2 – Problemas de classificação

Existe uma grande variedade de problemas que podem ser solucionados por métodos de classificação de dados. Neste tópico, serão descritos, de forma ampla, alguns problemas, de áreas distintas, que são solucionados com o auxílio das máquinas de vetores suporte.

1.2.1 – Classificação de Texto

O grande crescimento da quantidade de informação eletrônica frente à falta de ferramentas que possam manuseá-la de forma eficiente, acaba por originar um fenômeno conhecido como *sobrecarga de informação*. Diversas áreas sofrem com esse excesso de dados, sem ter como extrair informações importantes a partir deles, ou até mesmo sem conseguirem inferir um resultado satisfatório.

Esse crescimento dificulta a localização de informações importantes disponíveis, por exemplo, na WEB, que tem crescido exponencialmente em conjunto com a Internet. Os sistemas de mensagens eletrônicas, como *e-mail*, estão entre os mais prejudicados, não só pelo crescimento intrínseco da quantidade de informação, mas também pelo crescimento do número de *spams*. Existem algumas metodologias capazes de extrair as informações necessárias e classificar os dados que estão sendo analisados a fim de diminuir essa sobrecarga, como é o caso das máquinas de vetores suporte.

1.2.2 – Análise de Seqüências biológicas

Outra área que sofre com o excesso de informação é a área biológica. Com o mapeamento do genoma humano uma série de perguntas e dúvidas surgiram. Para solucionar grande parte dessas questões é necessário que existam formas de extrair informações desses novos dados. Seqüências protéicas homólogas, por exemplo, são classificadas como seqüências que compartilham de um ancestral comum, isto é, em

algum ponto da história evolucionária houve uma seqüência protéica única, a qual, através de processos de especialização ou duplicação gênica e divergência, produziu as duas seqüências homólogas atuais.

A inferência de homologia, ancestralidade comum, é a conclusão mais poderosa que pode ser inferida através do processo de busca por similaridade entre seqüências protéicas. A importância desse resultado deve-se ao fato de proteínas homólogas compartilharem estruturas similares. Como as funções das proteínas são determinadas por suas estruturas, ao concluir que duas proteínas são homólogas, pode-se afirmar que elas possuem funções semelhantes. Assim, o uso do classificador SVM pode ajudar na formação de grupos de proteínas, permitindo que uma nova proteína possa ser analisada a partir de informações de outras proteínas analisadas e conhecidas previamente.

Outro resultado importante é que a partir da descoberta de homologia entre seqüências de aminoácidos pode se determinar a que família protéica a nova proteína pertence, fornecendo os primeiros indícios sobre a funcionalidade da proteína. Nos últimos anos, os bancos de dados de seqüências protéicas e de DNA cresceram tanto em tamanho quanto em importância. Esse crescimento está relacionado à maior facilidade de se encontrar seqüências homólogas, mas por outro lado, houve uma maior exigência para o desenvolvimento de técnicas cada vez mais eficientes para a solução destes tipos de problemas.

Na Figura 1.2 é exibido um exemplo do funcionamento de um classificador para o caso da análise de uma seqüência de DNA. O dado de entrada, ou amostra, é representado por um vetor e aplicado à função do classificador. Essa função retorna um valor que será referente à classe mais indicada para a amostra.

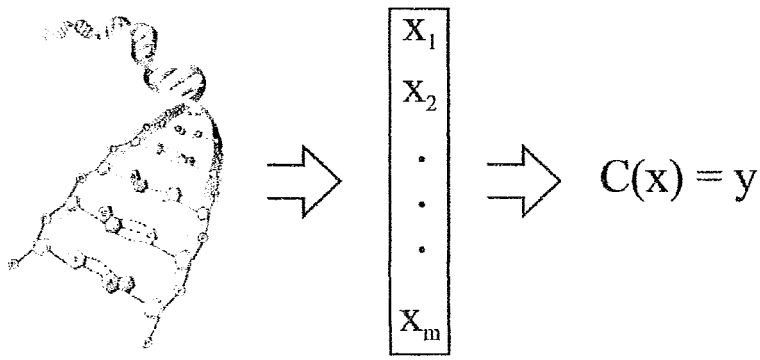


Figura 1.2 - Aplicação de uma amostra de DNA a um classificador.

Utilizando um sistema semelhante ao ilustrado na Figura 1.2 é possível unir as seqüências biológicas em grupos de acordo com suas similaridade. Sendo possível inferir qual a funcionalidade da nova seqüência classificada.

1.2.3 – Diagnóstico de doenças

O uso de classificadores para inferir diagnósticos também tem sido bastante estudado. Basicamente, o classificador tem como dados de entrada um conjunto de informações de um paciente e como resultado informa se o mesmo está doente ou não.

O problema *Breast Cancer Diagnosis*, ou Diagnóstico de Câncer de Mama tornou-se um dos problemas mais importantes e utilizados pelos softwares de classificação. Esse problema foi estudado por Bennett e Mangasarian (BENNETT *et al.*, 1992) e por Bradley, Fayyad e Mangasarian (BRADLEY *et al.*, 1998), dentre outros.

O SVM tem sido utilizado com grande sucesso na definição de uma função capaz de diagnosticar se um indivíduo possui câncer de mama ou não. De maneira reduzida, uma amostra celular de um paciente é classificada ou como benigno ou como maligno.

Primeiramente, é extraída uma amostra da mama do paciente, através de um procedimento ambulatorial, utilizando-se uma pequena agulha. A amostra é analisada fazendo um contraste no núcleo das células. Esse contraste é então digitalizado produzindo imagens que serão utilizadas no processo de obtenção das características das células que estão sendo analisadas.

Com a imagem obtida, são definidos os limites exatos dos núcleos e dez características são computadas para cada núcleo. O valor médio, o valor extremo e o erro padrão são então computados para os núcleos de uma imagem, para cada uma das características. Dessa forma, esse procedimento mapeia cada amostra da mama em um vetor de dimensão igual a trinta, sendo cada posição do vetor ocupada por valores reais.

A base de dados do problema de diagnóstico de câncer contém 198 observações e cada uma delas possui 32 características, sendo cada observação obtida pelo método descrito anteriormente. Cada um desses casos possui diagnóstico conhecido e são classificados como malignos ou benignos. Esse problema de diagnóstico de câncer de mama é, em particular, não linearmente separável, ou seja, a função de separação dos dados é não linear.

A partir do uso desse conjunto de dados é possível treinar um classificador de forma que ele seja capaz de inferir a qual grupo um novo paciente pertence. Assim, tendo como entrada a análise feita para as células da mama de um paciente, é possível determinar se ele está doente ou não.

Em geral, todos os classificadores possuem a mesma lógica de funcionamento descrito na Figura 1.2.

1.3 – Seqüência da apresentação

No presente trabalho, considera-se a resolução do problema conhecido como Máquinas de Vetores Suporte, ou *Support Vector Machine* (SVM). Esse problema é relativamente novo na literatura, sendo a sua proposição original apresentada por Vapnik em 1995 (VAPNIK, 1995).

Desde então, a comunidade científica tem dedicado uma atenção extremamente expressiva ao tema. Esse grande interesse sobre o problema SVM decorre do mais amplo escopo de suas aplicações práticas, podendo ser aplicado em diversas áreas na solução de problemas distintos, como visto anteriormente.

A formulação matemática do problema SVM apresenta algumas particularidades como a não linearidade, não convexidade e não diferenciabilidade. Essas características impedem a solução desse problema pelos métodos padrões de solução de problemas com restrição.

No presente trabalho, a técnica de suavização hiperbólica será adotada como a alternativa para a solução do problema SVM. Essa técnica corresponde a um desdobramento direto do método da penalização hiperbólica, destinado à solução do problema geral de programação não linear com restrições, originalmente apresentado por Xavier em 1982 (XAVIER, 1982).

A suavização hiperbólica tem sido usada para a solução de diversos problemas de programação matemática não diferenciável. Primeiramente foi utilizado para a resolução de um problema de calibração automática de modelos hidrológicos, como apresentado em (DIB, 1994) e em (SILVA *et al.*, 1990). Logo a seguir, foi adotada para resolver um problema de controle elétrico, conforme apresentado em (MOTA *et al.*, 1992), e para a minimização de funções definidas por mais de uma cláusula, conforme em (XAVIER, 1993).

Como os resultados obtidos inicialmente foram de fato satisfatórios, essa abordagem foi a seguir utilizada para a resolução de um problema de grande importância teórica e prática, o problema *min-max*. A descrição detalhada desse problema, bem como sua resolução através da técnica de suavização hiperbólica são encontrados em (CHAVES, 1987) e em (CHAVES *et al.*, 1998).

Uma síntese desse conjunto de aplicações é apresentada em (SANTOS, 1997). Mais recentemente, essa técnica tem sido utilizada na resolução de problemas de recobrimento, simples e múltiplo, de uma região plana por círculos, conforme apresentado em (XAVIER, 2000), (XAVIER *et al.*, 2003), (XAVIER *et al.*, 2005) e (BRITO, 2004). As aplicações mais recentes dessa abordagem consideram a resolução de problemas de agrupamento (*Clustering*), em (XAVIER, 2005), e o de classificação segundo critério de máquina de vetor suporte, em (XAVIER *et al.*, 2006).

Na técnica de suavização hiperbólica, em todos os problemas acima citados, a solução é obtida através da resolução de uma seqüência infinita de problemas continuamente diferenciáveis, classe C^{∞}, que gradativamente se aproximam do problema original. Registra-se que o desempenho computacional dessa técnica, frente a todos esses problemas, sempre obteve êxito.

Neste trabalho de dissertação, considera-se, em particular, a extensão e o aprimoramento do uso da abordagem da suavização hiperbólica para a resolução do problema SVM, primeiramente utilizada no artigo (XAVIER *et al.*, 2006). Para melhor descrever essa técnica este trabalho está organizado de acordo com a seguinte seqüência. No capítulo 2 é dada a definição de classificação, bem como a descrição de algumas técnicas utilizadas na solução desses problemas de classificação. Além disso, é feita uma revisão bibliográfica, de forma resumida, sobre diversos métodos de classificação, onde estão definidos de maneira formal os problemas SVM e agrupamento (*Clustering*).

No capítulo 3 é feita uma descrição formal do problema matemático associado ao SVM, bem como suas principais características e a metodologia de solução utilizada na sua resolução.

No capítulo 4 são apresentados os primeiros experimentos computacionais desenvolvidos e os correspondentes resultados obtidos comparando-os com os resultados apresentados pelo *SVMLight*, que é um software desenvolvido há cerca de dez anos, sendo amplamente utilizado na solução do problema SVM. O *SVMLight* também é descrito com maiores detalhes nesse capítulo.

No capítulo 5 é desenvolvida uma técnica de redução do conjunto de treinamento, utilizada para melhorar a eficiência do software desenvolvido. Os novos resultados obtidos após a aplicação dessa nova técnica também são exibidos nesse capítulo, comparando-os, novamente, com o *SVMLight*.

Finamente, no capítulo 6 são apresentadas as conclusões e sugestões para futuros trabalhos.

Capítulo 2 – Técnicas de Organização de Dados

2.1 – Introdução

O crescimento da quantidade de informações disponibilizadas pelos meios de comunicação, como Internet, é uma das motivações para o estudo de técnicas de classificação. Esses dados são disponibilizados de forma desordenada, tornando difícil obter informações relevantes a respeito de um assunto específico.

Existem dois tipos de enfoques básicos e naturais para a organização desses dados: *agrupamento* e *classificação*. A técnica de *agrupamento* consiste em agrupar os dados através de uma métrica qualquer que informe a similaridade entre eles, de forma que os dados mais semelhantes fiquem no mesmo grupo. Já na *classificação* tem-se um conjunto de dados em que a informação de pertinência a uma particular classe de um conjunto de classes é conhecida. A problemática se destina a classificar um dado em que a classe é desconhecida. Em princípio, esse dado deve ser atribuído à classe na qual o seu perfil melhor se encaixa.

A diferença principal entre classificação e agrupamento está na forma como os dados de entrada são fornecidos. Nos problemas de classificação cada dado está rotulado com alguma informação que especifica a qual classe ele pertence. Já nos problemas de grupamento os dados não são rotulados. Através do uso de uma métrica é que se determina o grau de semelhança entre eles. Assim, os dados mais semelhantes são agrupados em grupos comuns, de forma que os dados com menor similaridade ficam em grupos diferentes.

2.2 – Técnicas de Classificação

Existem várias metodologias diferentes utilizadas para solucionar o problema de classificação. Essas metodologias são classificadas basicamente em quatro áreas: análise exploratória, predição de modelos, otimização e análise de decisão (ISAAC, 2003). As quatro áreas são descritas sucintamente a seguir:

- **Análise Exploratória**: as metodologias desse grupo não utilizam informações externas, elas extraem dos dados as informações necessárias para o processo de classificação. Um exemplo de uma técnica deste grupo é a análise de *cluster* (agrupamento), onde os dados mais semelhantes são agrupados no mesmo grupo.
- **Predição de Modelos**: nesse grupo os dados são identificados e representados matematicamente. As técnicas deste grupo utilizam informações externas no processo de classificação e geralmente são utilizadas para fazer predições ou classificações sobre um novo dado desconhecido. Essas técnicas são, em sua maioria, aplicadas a problemas com grandes quantidades de dados de entrada. Alguns exemplos deste grupo são *boosting*, análise discriminante, redes neurais, reconhecimento de padrões, regressão, máquinas de vetores suporte e análise supervisionada.
- **Técnicas de otimização**: nessas técnicas se busca um conjunto de possíveis soluções para um problema restrito ou irrestrito com o objetivo de maximizar ou minimizar uma função matemática particular. Dentre as técnicas deste grupo destacam-se os algoritmos genéticos, programação linear e programação não linear. Várias técnicas de predição de modelo e análise de decisão utilizam técnicas de otimização para obter suas soluções.
- **Análise de decisão**: a proposta dessas técnicas é de avaliar as alternativas de tomada de decisão, fazendo a melhor opção em situações complexas, usualmente sobre incertezas. Alguns exemplos são modelos gráficos de decisão, análise de decisão de objetivos múltiplos e decisão seqüencial.

2.3 – *Boosting*

Boosting é uma técnica utilizada para melhorar a performance de uma metodologia de predição já existente. Essa técnica utiliza uma seqüência de algoritmos de classificação para atualizar os pesos dos dados dos conjuntos de treinamento. A técnica de *Boosting* produz bons resultados quando os algoritmos utilizados para atualizar os pesos dos dados do conjunto de treinamento são desenvolvidos a partir de metodologias simples de classificação.

O objetivo dessa metodologia é articular um conjunto de *classificadores simples*, que não são capazes de apresentarem bons resultados sozinhos, e, por essa articulação, produzir um classificador final eficiente. Essa técnica é útil na solução de problemas onde a obtenção de um único classificador eficiente é lenta e complexa. Assim, outras técnicas são utilizadas para produzirem classificadores que, embora não sejam capazes de solucionar o problema original, representam uma estrutura simplificada do problema.

Boosting organiza esses diversos classificadores simples e produz um classificador geral e eficiente para solucionar o problema. A solução de um problema de classificação complexo, por exemplo, utilizando a tecnologia de *árvore* produzirá uma árvore complicada e com grande profundidade[1]. A árvore gerada será um bom classificador, porém muito complexo. Para solucionar o mesmo problema de classificação pode ser produzida uma *árvore rasa* que embora seja simples não é um classificador eficiente. Como esses classificadores não são eficientes eles são chamados de *classificadores fracos*. A idéia da metodologia de *Bossting* é agrupar vários classificadores fracos e produzir um classificador que tenha a precisão dos classificadores complexos e a simplicidade dos classificadores fracos.

Como outro exemplo, seja o problema de estimar *log odds*, que é uma função definida no espaço de entrada das variáveis de predição. A técnica de *boosting* irá produzir essa estimativa da seguinte forma

$$LO(x) = \sum_{i=1}^{I} \lambda_i g_i(x), \tag{2.1}$$

onde $g_i(x)$ são os resultados produzidos pelas tecnologias de predição utilizadas na obtenção dos resultados parciais e λ_i são os pesos associados aos resultados de cada classificador. Se árvore é a tecnologia utilizada para obter os resultados parciais, então $g_i(x)$ é uma *árvore rasa*, como descrito no exemplo anterior. Assim, será obtida uma estimativa completa a partir da junção de vários resultados parciais, onde esses resultados são produzidos por classificadores simples.

---

[1] A profundidade de um nó de uma árvore é definida pelo maior número de nós existentes entre a raiz e o nó. A profundidade da árvore é dada pelo maior número de nós existentes entre a raiz a as folhas da árvore. Assim, uma árvore é dita profunda quando é grande o número de nós existentes entre a raiz e as folhas, e é dita rasa quando existem poucos nós entre a raiz e as folhas.

2.3.1 – AdaBoost

AdaBoost (*ADAptive BOOSting*) foi uma das primeiras técnicas de *boosting* a ser desenvolvida e é, também, uma das mais utilizadas (REYZIN, 2005). Essa técnica trabalha da seguinte maneira: seja D_t o vetor de pesos d_t que representa os pesos de cada uma das observações do conjunto de treinamento, onde $d_t(i)$ é o peso da observação i na iteração t. Para cada observação do conjunto de treinamento faça inicialmente $D_1(i) = 1/N$, $i = 1, ..., N$, onde N é o número de elementos do conjunto de treinamento. Seja, também, α_t o peso do resultado produzido (hipótese h_t) pelos classificadores fracos na iteração t. A cada iteração encontre h_t com o menor erro em relação ao vetor de pesos D_t e então atualize α_t, o peso da hipótese, da regra:

$$\alpha_t = \frac{1}{2}\log\frac{1-\mu_t}{\mu_t}; \tag{2.2}$$

onde μ_t é o somatório de todos os pesos pela hipótese h_t das observações classificadas erroneamente na iteração t:

$$\mu_t = \sum_{n=1}^{N} d_n^t I(y_n, h_t(x_n)); \tag{2.3}$$

tal que:

$$I(x,y) = \begin{cases} 1, & x \neq y \\ 0, & x = y. \end{cases} \tag{2.4}$$

Para cada elemento do conjunto de treinamento atualize D para a próxima iteração da seguinte maneira:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}; \tag{2.5}$$

onde Z_t é a norma de D_t.

O processo iterativo continua até que termine a etapa de treinamento. O classificador final será definido por:

$$H(x) = \text{sinal}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{2.6}$$

onde T é o número de ciclos do *boosting*.

2.4 – Análise *Cluster* (Agrupamento)

Agrupamento ou análise de agrupamento é uma técnica que busca identificar subconjuntos de dados de acordo com o grau de similaridade entre esses dados. As observações pertencentes a um mesmo subconjunto apresentam um grau de similaridade maior entre si que em relação aos dados dos demais subconjuntos. A técnica de agrupamento é distinta das técnicas de classificação. O agrupamento é conduzido de forma não supervisionada, ou seja, os dados do conjunto de treinamento não possuem rótulos indicando a qual classe eles pertencem e, além disso, não há, em alguns casos, informações sobre o número de classes do problema. A similaridade entre os dados é extraída de sua própria estrutura. Elementos pertencentes a uma determinada classe são mais homogêneos, mantendo grande similaridade entre si, enquanto que elementos pertencentes a classes distintas apresentam maiores diferenças.

O objetivo da técnica de análise de agrupamento é descobrir grupos naturais de observações de forma que as observações de um dado grupo tendem a serem similares entre si e dissimilares às observações dos outros grupos. Os modelos de agrupamento têm sido utilizados em aplicações de biologia, em estatística, em *marketing* para caracterizar grupos de consumidores similares, dentre outros. Assim, com o lançamento de um novo produto, por exemplo, se tem a informação de qual grupo de consumidores irá se interessar mais ou menos pelo novo produto.

Existem, basicamente, duas técnicas de agrupamento, que serão descritas nos tópicos abaixo: agrupamento hierárquico e agrupamento via particionamento.

2.4.1 – Algoritmos Hierárquicos

A técnica de agrupamento hierárquico é um procedimento que constrói uma "árvore" onde em cada nível dessa árvore representa uma alteração na estrutura dos grupos. Existem dois tipos de agrupamento hierárquico. O primeiro é chamado de agrupamento hierárquico *aglomerativo* que adiciona os dados aos grupos em cada nível da árvore. O segundo é denominado agrupamento hierárquico *particionado* que realiza sucessivas divisões nos grupos.

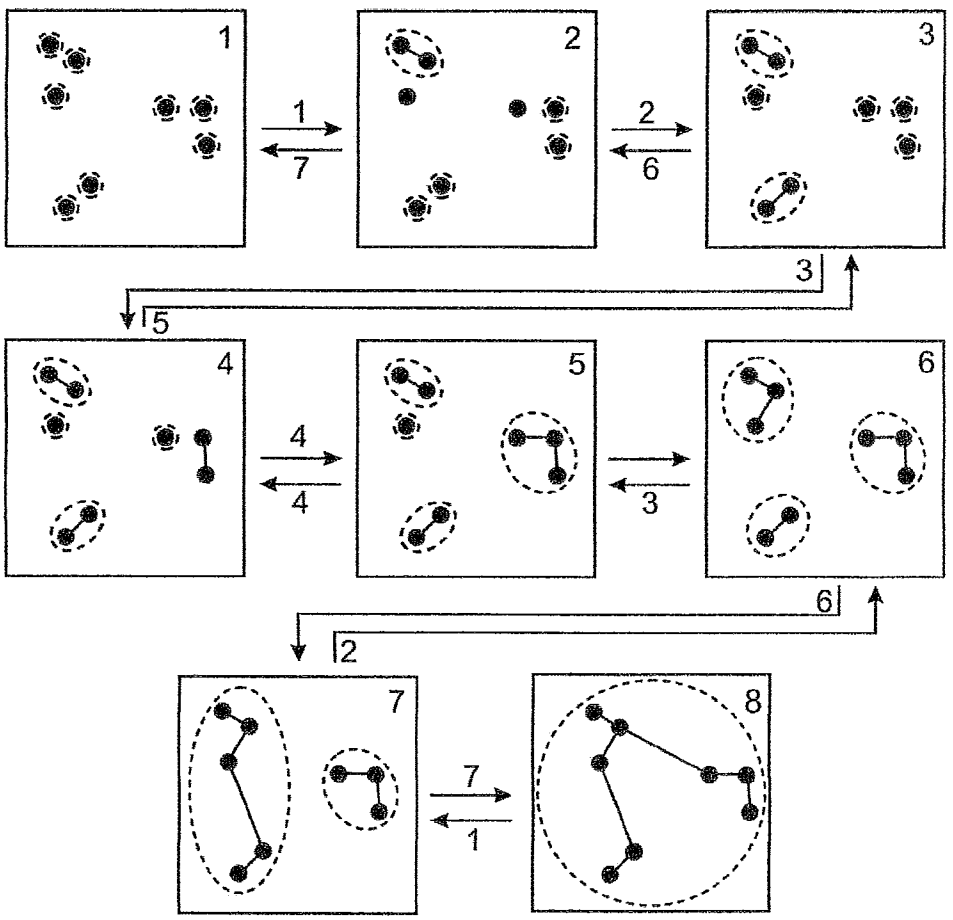


Figura 2.1 - Agrupamento Hierárquico Aglomerativo e Particionado

Seja *N* o número de elementos de um problema de grupamento. Os métodos hierárquicos aglomerativos começam com *N* grupos, onde cada um dos grupos possui uma única observação. O número de grupos é então reduzido para *N-1* pela junção de duas observações em um único grupo baseado em algum critério de similaridade entre elas. O processo continua até que o número desejado de grupos seja atingido ou até que

seja obtido um único grupo que contenha todas as observações. Nessa técnica a árvore é construída das folhas para a raiz, onde cada folha representa uma observação.

Um exemplo do funcionamento dos métodos aglomerativos é ilustrado na Figura 2.1. O processo inicia de cima para baixo e pode ser observado no primeiro quadro que cada observação representa um grupo, ou seja, no começo do procedimento têm-se oito grupos distintos. Em cada etapa do procedimento aglomerativo os dois grupos mais próximos são unificados em um único grupo e o processo continua até que todos as observações façam parte do mesmo grupo como ilustrado no último quadro. Entretanto, não se deseja ter como resultado todas as observações em um mesmo grupo. Pode ser observado que o sexto quadro apresenta a melhor formação, onde as observações mais próximas estão nos mesmos grupos.

Os métodos hierárquicos particionados trabalham no sentido oposto dos métodos aglomerativos. Esses métodos começam com um grupo de N elementos e divide esse grupo em grupos menores usando um critério de dissimilaridade entre os dados. Um critério bastante utilizado de dissimilaridade é a distância entre os dados. Várias funções de distância podem ser utilizadas, sendo a mais tradicional a distância *Euclidiana*. O processo de particionamento continua até que algum critério seja atingido ou até que sejam obtidos N grupos com um elemento cada um. O processo dos métodos hierárquicos particionados também é ilustrado na Figura 2.1, pela ordem inversa, começando do oitavo quadro até o primeiro quadro.

2.4.2 – Algoritmos não Hierárquicos

As técnicas de agrupamento não hierárquico buscam minimizar a distância intragrupos, ou seja, minimizar a distância entre as observações pertencentes ao mesmo grupo. Essas técnicas exigem que o número K de grupos seja definido no começo do processo. Após ser definido o número de grupos as observações são separadas em K grupos. Em cada iteração, a distância entre as observações e os centros dos grupos são calculadas. Cada observação será então associada ao grupo do centro mais próximo. Assim, a observação pode permanecer no mesmo grupo ou ser associada a um outro grupo. Os centros dos grupos são recalculados e as distâncias são novamente obtidas. As iterações continuam até que não haja nenhuma troca entre os grupos. As técnicas de agrupamento não hierárquico podem trabalhar com conjuntos de dados muito maiores que as técnicas de agrupamento hierárquico.

Vários algoritmos foram desenvolvidos para solucionar o problema de grupamento. O algoritmo *k-Means*, embora não seja o mais eficiente, é o mais utilizado. Este algoritmo tem como entrada o número de grupos k e em primeiro lugar são determinados k centróides $c^j, j=1,...,k$. Após a determinação inicial dos centróides em cada iteração i as observações $x^i, i=1,...,m$, são associadas ao grupo $l(i)$ seguindo o critério que o centróide $c^{l(i)}$ é o mais próximo da observação x^i. Os centróides são novamente determinados e as observações são novamente associadas aos grupos. Esse processo continua até que não haja alterações entre os centróides.

2.4.3 – Formulação Matemática

Agrupamento é definido como sendo um problema de associar m pontos $\{x^1, x^2, ..., x^m\}$, definidos em um espaço real de dimensão n ($\Re^n$), a k grupos $\{c^1, c^2, ..., c^k\}$ de tal forma que a soma das distâncias de cada ponto ao centróide mais próximo seja mínima. Centróide é um ponto que melhor representa o grupo ao qual ele pertence. Este ponto terá as principais propriedades do grupo e este grupo passa a ser representado por ele. Assumindo $\|.\|$ uma norma arbitrária no $\Re^n$, o problema de agrupamento é então definido na forma

$$\min_{c^1,...,c^k} \sum_{i=1}^{m} \min_{l=1,...,k} \left\| x^i - c^l \right\| \tag{2.7}$$

A equação (2.7) possui um somatório do mínimo de um conjunto de funções convexas que, na maioria dos casos, não é nem convexo e nem côncavo. Assim, esse problema torna-se bastante complexo, sendo de difícil resolução.

O critério mais natural, intuitivo e freqüentemente adotado para (2.7) é de minimizar a soma de quadrados das distâncias das observações ao seu centróide mais próximo. Esse problema é conhecido na literatura como *minimum sum-of-squares clustering* (MSSC). Este critério corresponde à minimização da variância intragrupos.

É um critério que contempla simultaneamente os objetivos de homogeneidade e de separação, pois de acordo com o Teorema de Huygens, minimizar a variança intra (homogeneidade dentro dos grupos) é equivalente a maximizar a variança entre grupos (separação entre os grupos).

Trata-se de um problema não-convexo e não-diferenciável. Xavier (XAVIER, 2005), apresenta uma metodologia de resolução baseada em suavização das não diferenciabilidades.

2.5 – Máquina de Vetor suporte (SVM)

SVM é um problema particular de classificação relativamente novo na literatura, pois foi proposta por Vapnik em 1995 (VAPNIK, 1995). Esta técnica será discutida com mais detalhes no capítulo 3.

De forma simplificada, pode-se dizer que essa técnica é utilizada para separar dois conjuntos e esses conjuntos são separáveis linearmente por um hiperplano. O objetivo é produzir um classificador que irá trabalhar bem para exemplos não conhecidos. Para tal o SVM determina esse hiperplano separador de forma que a distância do hiperplano às classes seja minimizada.

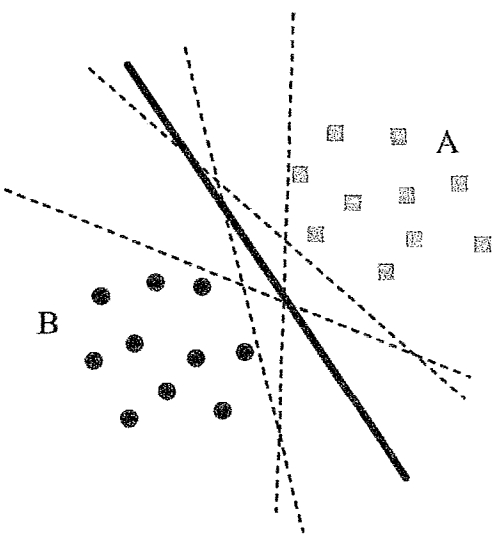


Figura 2.2 - Hiperplanos separadores para dois conjuntos de dados

No caso simples tem-se dois conjuntos de dados linearmente separáveis *A* e *B*. Os dados dessas duas classes são representados por um par (s_i, y_i), onde s_i representa as coordenadas da observação *i* e y_i é o rótulo da observação. O rótulo indicará a qual classe s_i pertence e, geralmente é dado por $y_i = \pm 1$. Como mostra na Figura 2.2, existem infinitos hiperplanos capazes de separar as classes *A* e *B* linearmente. Entretanto, existe somente um hiperplano que maximize a margem separando as duas

classes. Esse é o objetivo do SVM, determinar o hiperplano que gere a maior margem entre as classes. A margem é definida como sendo a distância entre o hiperplano e o ponto mais próximo da classe *A* somado à distância do hiperplano ao ponto mais próximo da classe *B*. Ao maximizar a margem, o poder de generalização do classificador aumenta, ou seja, a chance do classificador acertar ao definir a qual classe um dado desconhecido pertence aumenta. Assim, o número de erros produzidos pelo classificador diminui.

2.5.1 – Descrição

Vapnik propôs em 1995 (VAPNIK, 1995) uma nova técnica para classificação de dados denominada *Support Vector Machine* (SVM). Esta técnica é utilizada, na maioria dos casos, para classificar dados em classes diferentes. A complexidade do SVM está relacionada à forma pela qual os dados estão distribuídos. O caso mais simples ocorre quando os dados podem ser separados linearmente por um hiperplano (Figura 2.3).

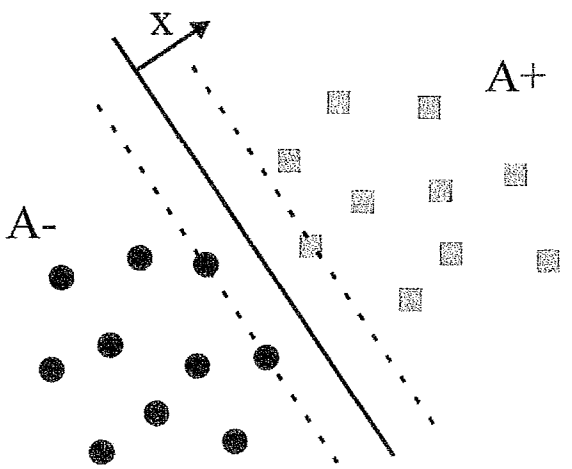


Figura 2.3 - SVM Linear

A entrada do SVM consiste em um conjunto de dados, onde cada um é rotulado com um valor indicando a qual classe ele pertence. Estes dados de entrada são denominados como *conjunto de treinamento*. Em geral, os problemas SVM separam, por um hiperplano, dois conjuntos de observações, onde essas observações são definidas da seguinte forma:

$$D=\left\{\left(s^1,y^1\right),...,\left(s^l,y^l\right)\right\}, s\in\Re^n, y\in\{-1,1\}. \tag{2.8}$$

O conjunto de observações será separado de forma ótima pelo hiperplano quando não existirem erros na separação e a soma das distâncias entre os pontos mais próximos ao hiperplano seja máxima. Assim, o SVM fornece como solução um hiperplano que separa os dados com a máxima margem. Esta margem é definida, no caso linear, pela soma da distância do ponto mais próximo ao hiperplano da classe positiva e da distância do ponto mais próximo ao hiperplano da classe negativa, como mostra a Figura 2.3. Estes pontos que fornecem a distância mínima da classe ao hiperplano são denominados *Vetores Suporte*.

2.5.2 – Formulação

O SVM linear apresenta uma formulação bastante simples, dada como segue:

$$u = x.s - \gamma \tag{2.9}$$

onde x é o vetor normal ao hiperplano separador, s é o vetor do conjunto de pontos de entrada e γ determina o deslocamento do hiperplano em relação a origem. Os pontos mais próximos ao hiperplano estão sobre os planos $u = 1$, para os pontos da classe positiva, e $u = -1$ para os pontos da classe negativa. Assim, tem-se a seguinte relação:

$$\begin{cases} x.s_i - \gamma \geq 1 & s_i \in Classe\,1 \\ x.s_j - \gamma \leq -1 & s_j \in Classe\,-1. \end{cases} \tag{2.10}$$

onde s_i representa o i-ésimo elemento do conjunto de treinamento. Em particular, como exibido na Figura 2.4, $x.s_i - \gamma = +1$ para todos os pontos suporte da classe 1 e $x.s_j - \gamma = -1$ para os pontos suporte da classe 2. A margem então pode ser calculada pela soma destas equações:

$$\begin{aligned} m &= (x.s_i - \gamma) + (x.s_j - \gamma) \\ &= \frac{|\langle x, s_i \rangle - \gamma|}{\|x\|} + \frac{|\langle x, s_j \rangle - \gamma|}{\|x\|} \\ &= \frac{1}{\|x\|}\left(|\langle x, s_i \rangle - \gamma| + |\langle x, s_j \rangle - \gamma|\right). \end{aligned} \tag{2.11}$$

Como na solução ótima $|\langle x, s_i \rangle - \gamma| = |\langle x, s_j \rangle - \gamma| = 1$, no final do processo de otimização a margem será dada por:

$$m = \frac{2}{\|x\|}. \tag{2.12}$$

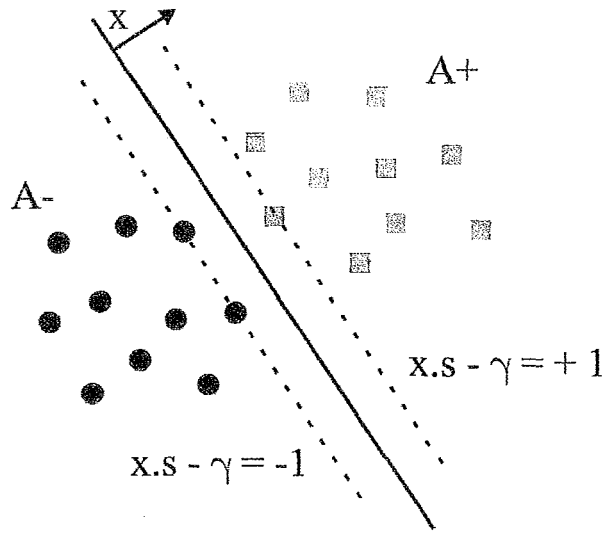


Figura 2.4 - Definição da máxima margem

A margem pode ser maximizada através do seguinte problema de otimização (BURGES, 1998):

$$\begin{aligned} &\underset{x,\gamma}{\textbf{minimizar}} \ \frac{1}{2}\|x\|^2 \\ &s.a \\ &\quad y_i(x.s_i - \gamma) \geq 1, \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{2.13}$$

onde l representa o número de elementos do conjunto de treinamento e y_i é o rótulo que indica a qual classe o elemento s_i está associado. Tem-se $y_i = +1$ para os exemplos positivos e $y_i = -1$ para os exemplos negativos.

Pode ser observado que esta metodologia representa um problema de otimização quadrática. Usando a função *Lagrangeana*, esse problema de otimização pode ser

transformado em uma formulação dual, onde a função objetivo depende unicamente do conjunto de multiplicadores de Lagrange. Esta formulação é dada por (PLATT, 1998):

$$\begin{aligned} &\underset{\alpha}{\textbf{maximizar}}\ X(\alpha) = -\sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (s_i, s_j) \\ &s.a. \\ &\qquad \sum_{i=1}^{l} y_i \alpha_i = 0 \\ &\qquad \alpha_i \geq 0, \quad \forall i = 1, ..., l \end{aligned} \tag{2.14}$$

Nesta formulação α é um vetor com l componentes, onde cada uma representa um elemento do conjunto de treinamento, ou seja, a componente α_i corresponde ao elemento (s_i, y_i).

Existe uma relação um para um entre cada multiplicador de Lagrange e cada elemento do conjunto de treinamento. Uma vez que os multiplicadores de Lagrange são calculados, o vetor normal x e γ podem ser determinados a partir da seguinte relação:

$$\begin{aligned} &x = \sum_{i=1}^{l} y_i \alpha_i s_i, \\ &\gamma = x.s_k - y_k \quad \textit{para algum } \alpha_k > 0. \end{aligned} \tag{2.15}$$

Como x pode ser calculado, pela equação (2.14), a partir do conjunto de treinamento, o custo computacional para calcular um SVM linear é constante e está diretamente relacionado ao número de vetores suporte.

É claro que o SVM não é utilizado somente para solucionar problemas linearmente separáveis. Muito pelo contrário, a maioria dos problemas estudados apresenta a característica de serem não linearmente separáveis e para estes casos a formulação anterior fornecerá uma solução infinita.

Em 1995, Cortes e Vapnik (CORTES *et al.*, 1995) sugeriram uma modificação na formulação do problema de otimização anterior (2.13) permitindo que um ponto seja classificado de forma errada, mas penalizando essas ocorrências. Dentro desse referencial, o novo problema é dado por:

$$\begin{aligned} &\underset{w,b}{\textbf{minimizar}} \frac{1}{2}\|x\|^2 + C\sum_{i=1}^{l}\xi_i \\ &s.a \\ &\quad y_i(x.s_i - \gamma) \geq 1 - \xi_i, \quad i \in \{1,2,...,l\} \\ &\quad \xi_i \geq 0 \end{aligned} \tag{2.16}$$

onde ξ_i é uma variável de folga que permite a observação i violar a margem e C é o fator de penalidade, que estabelece um compromisso entre o erro de treinamento e a margem. Esse parâmetro C pode ser visto também como um parâmetro que penaliza os pontos que violam a margem.

Quando este novo problema é transformado em sua forma dual, somente a restrição de desigualdade do problema, em relação ao problema (2.14), é modificada e a variável ξ_i, que possibilita que uma observação viole a margem, não aparece nessa formulação. O novo problema é dado por (PLATT, 1998):

$$\begin{aligned} &\underset{\alpha}{\textbf{maximizar}}\ X(\alpha) = -\sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (s_i, s_j) \\ &s.a. \\ &\quad \sum_{i=1}^{l} y_i \alpha_i = 0 \\ &\quad 0 \leq \alpha_i \leq C, \quad \forall i = 1,...,l. \end{aligned} \tag{2.17}$$

A formulação SVM pode ser generalizada também para utilizar separadores não lineares (BURGES, 1998). A classificação de uma nova observação desconhecida s, através de um classificador não linear, é feita explicitamente pelos multiplicadores de Lagrange, como segue:

$$u = \sum_{i=1}^{l} y_i \alpha_i K(s_i, s) - \gamma \tag{2.18}$$

onde K, denominada *função kernel*, indica a similaridade ou distância entre o vetor de entrada s e o vetor do conjunto de treinamento s_i. Dessa forma, caso $u \geq 1$ sabe-se que s pertence à classe 1, por outro lado, caso $u \leq -1$ a observação s será classificada como sendo da classe –1.

Alguns exemplos de funções *kernel* incluem Gaussianas, Polinomiais, funções de ativação do tipo sigmoidal. Se K é linear, então se tem o SVM linear original (2.9).

Os multiplicadores de Lagrange α_i continuam sendo calculados por um problema de otimização quadrática. Apesar da não-linearidade alterar a forma quadrática, a função objetivo dual continua sendo quadrática em α:

$$\begin{aligned} &\underset{\alpha}{\textbf{maximizar}}\ X(\alpha) = -\sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(s_i, s_j) \\ &s.a. \\ &\qquad \sum_{i=1}^{l} y_i \alpha_i = 0 \\ &\qquad 0 \le \alpha_i \le C, \quad \forall i = 1, ..., l. \end{aligned} \tag{2.19}$$

Para que o problema anterior seja positivo definido, a função *kernel* deverá obedecer às condições de Mercer (BURGES, 1998). As condições de existência de uma função *kernel* serão definidas no próximo tópico.

As condições de *Karush-Kuhn-Tucker* (KKT) são condições necessárias e suficientes para um ponto de um problema quadrático positivo definido, pois o problema é convexo. As condições de KKT para o problema (2.17) são particularmente simples. O problema é solucionado quando, para todo i:

$$\begin{aligned} &\alpha_i = 0 \Leftrightarrow y_i u_i \ge 1, \\ &0 < \alpha_i < C \Leftrightarrow y_i u_i = 1, \\ &\alpha_i = C \Leftrightarrow y_i u_i \le 1. \end{aligned} \tag{2.20}$$

onde $u_i = \pm 1$ é a saída do SVM para o i-ésimo exemplo do conjunto de treinamento.

2.5.3 – Condições de existência das funções *Kernel*

Em geral, as funções *kernel* devem satisfazer a algumas condições básicas para garantir a correta formulação de um *kernel*. A propriedade mais simples de ser verificada, em função da simetria do produto interno, é a que determina que um *kernel* deve ser simétrico

$$k(x, y) = k(y, x), \tag{2.21}$$

e então satisfazer a desigualdade de Cauchy-Schwartz

$$k^2(x, y) \leq k(x, x)\, k(y, y). \tag{2.22}$$

Para garantir a existência de um espaço de características é necessário que a simetria da função *kernel* seja definida positiva. Tem-se, assim, as condições de necessidade e de suficiência (MERCER, 1909). Definição positiva, significa que para qualquer conjunto de exemplos $s_1, ..., s_l$ e qualquer conjunto de números reais positivos $\lambda_1, ..., \lambda_l$ a função deve satisfazer a desigualdade

$$\sum_{i=1}^{l} \sum_{j=1}^{l} \lambda_i\, \lambda_j\, k\left(s_i, s_j\right) \geq 0. \tag{2.23}$$

Dessa forma, essas condições sendo obedecidas em uma função *kernel*, tem-se a garantia da existência de um espaço de características e, além disso, a garantia do bom funcionamento dessa função *kernel*.

Capítulo 3 – Suavização Hiperbólica

3.1 – Introdução

No capítulo anterior foram apresentadas diversas técnicas de classificação e agrupamento para categorização dos dados. Dentre os problemas apresentados, pode-se observar que a grande maioria de suas formulações é de natureza não diferenciável e não convexa.

No presente trabalho, será apresentada uma alternativa ao problema SVM apresentado no capítulo 2. Neste capítulo, será desenvolvida uma metodologia que transforma a formulação do problema SVM original em uma formulação mais simples, em um espaço de menor dimensão, que seja diferenciável C^{∞} e convexa. Na nova formulação uma seqüência de subproblemas diferenciáveis são solucionados aproximando-se do problema original.

Cada subproblema possui a forma padrão dos problemas de otimização, ou seja, possui uma função objetivo e um conjunto de m restrições de igualdade e p restrições de desigualdade, problema (3.1). Esses problemas são transformados intrinsecamente em problemas irrestritos através da técnica de penalização hiperbólica (XAVIER, 1982). Assim, tem-se um problema de otimização irrestrito e completamente diferenciável. A vantagem dessa metodologia é que cada um dos subproblemas pode ser solucionado, de forma mais simples, por métodos poderosos que utilizam informações da derivada, como gradiente conjugado, quase-Newton, dentre outros.

$$\begin{aligned} &\textbf{minimizar } f(x) \\ &s.a \quad g_i(x) \geq 0, \quad i = 1, ..., m \\ &\qquad h_j(x) = 0, \quad j = 1, ..., p \end{aligned} \tag{3.1}$$

3.2 – Definição do Problema

Existem várias formulações para os problemas de classificação. No capítulo anterior foi definido o problema de máquina de vetores suporte descrito por Vapnik, que foi o pioneiro na proposição desse problema. Para o desenvolvimento da nova metodologia que será apresentada neste capítulo será adotada como base uma outra

formulação do problema SVM definida por Mangasarian (MANGASARIAN *et al.*, 1999).

3.2.1 – Formulação SVM de Mangasarian

Como descrito no capítulo 2, a margem definida pelo hiperplano separador é igual a $2/\|w\|$. O objetivo do problema SVM é maximizar essa margem gerando um classificador com maior poder de generalização. Como maximizar a margem é o mesmo que minimizar o seu inverso, o problema SVM minimiza a função objetivo $\|w\|^2/2$.

A formulação do problema SVM de Mangasarian é equivalente a desenvolvida por Vapnik descrita na equação (2.13). Para os problemas de classificação, nos quais o conjunto de treinamento possui m pontos definidos no $\Re^n$ e esses são linearmente separáveis, Mangasarian propôs a seguinte formulação:

$$\begin{aligned} &\underset{(w,\gamma)\in\Re^{n+1+m}}{\textbf{minimizar}} \quad \tfrac{1}{2}w'w \\ &\quad s.a \quad y_i(w.s_i-\gamma)\geq 1, \quad i\in\{1,2,...,m\}. \end{aligned} \tag{3.2}$$

Sendo $s_i, i=1,...,m,$ o conjunto de observações de treinamento, y_i é o rótulo da observação, de forma que $y_i=\pm 1$. Caso a observação s_j pertença a classe +1 então $y_i=+1$, caso contrário $y_i=-1$. A variável w é a normal ao hiperplano separador e γ é o deslocamento deste hiperplano em relação à origem.

A superfície de separação linear é dada pelo hiperplano

$$s'w=\gamma \tag{3.3}$$

e os hiperplanos que limitam as classes, ou seja, os hiperplanos que estão nos limites das classes +1 e −1 são definidos, respectivamente, por:

$$\begin{aligned} s'w-\gamma&=+1 \\ s'w-\gamma&=-1, \end{aligned} \tag{3.4}$$

onde a distância entre esses hiperplanos determina a margem entre as duas classes.

Como a maior parte dos problemas de classificação são não linearmente separáveis, adiciona-se uma variável de folga ξ no problema (3.2), de forma que seja permitido que uma observação viole a margem produzida pelo classificador. Assim, com o uso da variável de folga, as observações responsáveis por tornar o problema não separável ficam classificadas de forma errônea, como pode ser observado na Figura 3.1. O novo problema fica então definido por:

$$\begin{aligned} \underset{(w,\gamma)\in\Re^{n+1+m}}{\textbf{minimizar}} \quad & C\sum_{i=1}^{m}\xi_i + \tfrac{1}{2}w'w \\ s.a \quad & y_i\left(w.s_i - \gamma\right) + \xi_i \geq e \\ & \xi_i \geq 0, \quad i \in \{1,2,...,m\}. \end{aligned} \tag{3.5}$$

Para o problema anterior, no qual os dados não são linearmente separáveis, os hiperplanos da equação (3.6) determinam os limites das classes definindo uma *margem suave* (*soft margin*). Essa margem está diretamente relacionada à não negatividade da variável de folga y.

$$\begin{aligned} s_i'w - \gamma + \xi_i \geq +1, \quad & para \;\; y_i = +1, \\ s_i'w - \gamma - \xi_i \leq -1, \quad & para \;\; y_i = -1. \end{aligned} \tag{3.6}$$

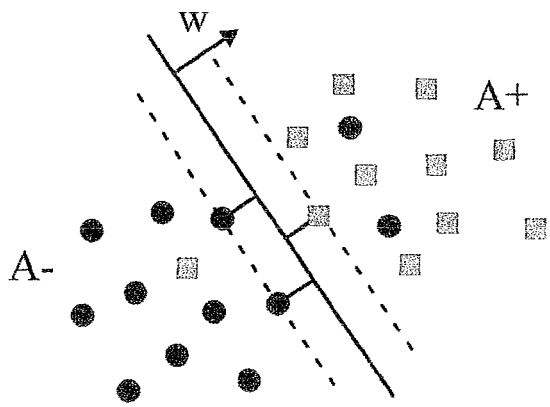


Figura 3.1 - Problema não linearmente separável

3.2.2 – Formulação SVM adotada

Neste trabalho, será adotada a formulação a seguir especificada. Seja $S = \{s_1, s_2, ..., s_m\}$ um conjunto de m observações, onde cada observação genérica $s_j \in \Re^n$, $j = 1, ..., m$. Seja um conjunto $J = \{1, 2, ..., m\}$ tal que $J = J_1 \cup J_2$, onde J_1 é o conjunto das observações pertencentes à classe 1 e J_2 é o conjunto de observações pertencentes à classe -1. Ou seja, se $s_j \in J_1$ a observação pertence a classe 1 e caso $s_j \in J_2$ será uma observação pertencente à classe 2. Assim, $J_1 \cap J_2 = \emptyset$.

O objetivo deste problema é encontrar um hiperplano $x \in \Re^n$ que separe os dados das duas classes da melhor forma possível segundo um critério particular. Especificamente, pretende-se obter um hiperplano tal que a margem de separação entre os dados seja a maior possível.

Seja

$$Z_i = \underset{j \in J_i}{\text{minimo}} \quad x s_j + \gamma, \qquad i = 1, 2 \tag{3.7}$$

a distância mínima entre a classe i e o hiperplano definido por x. Como deseja-se maximizar a margem, um novo problema de otimização pode ser definido da seguinte forma

$$\begin{aligned} &\text{minimizar} - (Z_1 + Z_2) \\ &s.a \\ &\quad Z_1 = \underset{j \in J_1}{\text{mínimo}} (x s_j + \gamma) \\ &\quad Z_2 = \underset{j \in J_2}{\text{mínimo}} (-(x s_j + \gamma)) \\ &\quad \|x\|_2 = 1 \end{aligned} \tag{3.8}$$

As restrições do problema (3.8), após uma simples manipulação, podem ser reescritas como

$$\begin{aligned} Z_1 &= \left[\underset{j \in J_1}{\text{mínimo}} (x s_j) \right] + \gamma \\ Z_2 &= \left[\underset{j \in J_2}{\text{mínimo}} (-x s_j) \right] - \gamma. \end{aligned} \tag{3.9}$$

Pode ser observado que a variável γ é anulada pelas soma das restrições. Assim, desde que a variável γ entra na função objetivo com sinais contrários, o resultado é indiferente ao seu valor assumido. Como o papel de γ é completamente inócuo, o mesmo será retirado do problema (3.8) sem qualquer prejuízo. A partir da retirada da variável γ, chega-se a seguinte formulação equivalente, com dimensão $(n+2)$:

$$\begin{aligned} &\underset{x,Z_1,Z_2}{\text{minimizar}} -(Z_1+Z_2) \\ &s.a \\ &\quad Z_1 = \underset{j\in J_1}{\text{mínimo}}\left(x\, s_j\right) \\ &\quad Z_2 = \underset{j\in J_2}{\text{mínimo}}\left(-x\, s_j\right) \\ &\quad \|x\|_2 = 1 \end{aligned} \tag{3.10}$$

Para os problemas cujos dados do conjunto de treinamento são linearmente separáveis pode ser definida a seguinte relação entre a formulação proposta (3.8) e a formulação de Mangasarian (3.5)

$$Z_1 + Z_2 = \frac{2}{\|w\|_2} \tag{3.11}$$

$$\gamma = \frac{Z_1 - Z_2}{2} \tag{3.12}$$

$$x = \frac{Z_1 + Z_2}{2} w \tag{3.13}$$

onde w é o vetor normal ao hiperplano separador na formulação do Mangasarian.

As racionalidades dessas relações acima podem ser obtidas facilmente. Como $Z_1 + Z_2$ define a margem entre as classes, a relação (3.11) é dada diretamente. Tem-se que γ é o deslocamento do hiperplano separador à origem, portanto, com o auxílio da Figura 3.2, a relação (3.12) pode ser inferida. Pode ser observado que o vetor x, da formulação proposta, e o vetor w, da formulação de Mangasarian, são vetores normais ao hiperplano. Entretanto, w não está normalizado, enquanto que $\|x\|_2 = 1$. Assim, pode-se afirmar que

$$x = \frac{w}{\|w\|_2} \tag{3.14}$$

e substituindo (3.11) em (3.14) tem-se a relação (3.13).

Assim, a partir das relações definidas acima, para o caso onde as classes são linearmente separáveis, obtém-se que o problema (3.8) é equivalente ao problema (3.5) e, conseqüentemente, suas soluções são iguais.

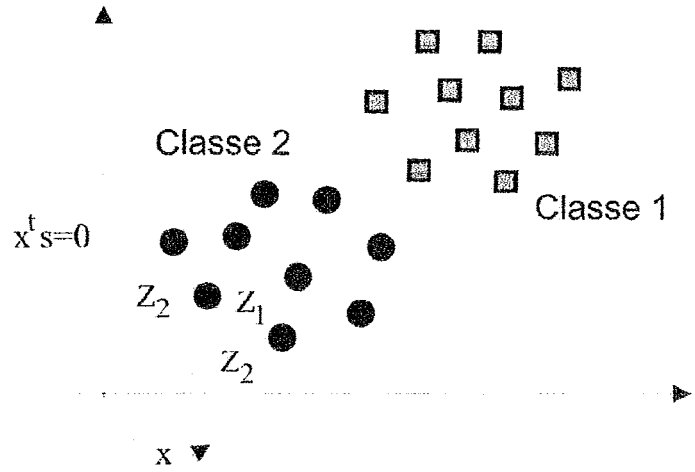


Figura 3.2 - Distância mínima entre a classe e o hiperplano

Para os problemas nos quais o conjunto de treinamento é formado por dados não linearmente separáveis a igualdade das soluções está diretamente relacionada ao valor do parâmetro C, definido nas formulações de Mangasarian e Vapnik .

Para essas duas formulações, quando o parâmetro C decresce, o termo de maximização da margem tem sua importância gradualmente crescente. Assim, conforme C decresce, o tamanho da margem aumenta (GUNN, 1998). Quando o parâmetro $C = 0$, toda a ênfase é dada ao termo de maximização da margem, e a solução produzida possui a máxima margem. Dessa forma, com $C = 0$, as soluções de Mangasarian e Vapnik e da formulação proposta continuam sendo iguais.

A Figura 3.3 tem o objetivo de ilustrar essa situação. Os três pontos extremos estão envolvidos com linhas tracejadas. Quando $C = 0$, somente esses três pontos influenciam as soluções de Mangasarian e Vapnik e da formulação proposta.

Por outro lado, no caso em que $C \neq 0$ as soluções obtidas são diferentes. Essa diferença é produzida pela contribuição diferenciada dos pontos não-separáveis nos dois critérios. No critério de Mangasarian e Vapnik todos os pontos não separáveis contribuem na obtenção do hiperplano separador. Já na formulação proposta, somente os pontos mais não linearmente separáveis contribuem, ou seja, somente os pontos que mais violam a margem participam da definição do novo hiperplano.

A Figura 3.3 igualmente pode ser usada para ilustrar esse caso. A solução produzida pelas formulações de Mangasarian e Vapnik com $C \neq 0$ seriam influenciadas por todos os pontos com violação. Esses pontos são mostrados na figura por 4 círculos cheios e 5 quadrados cheios.

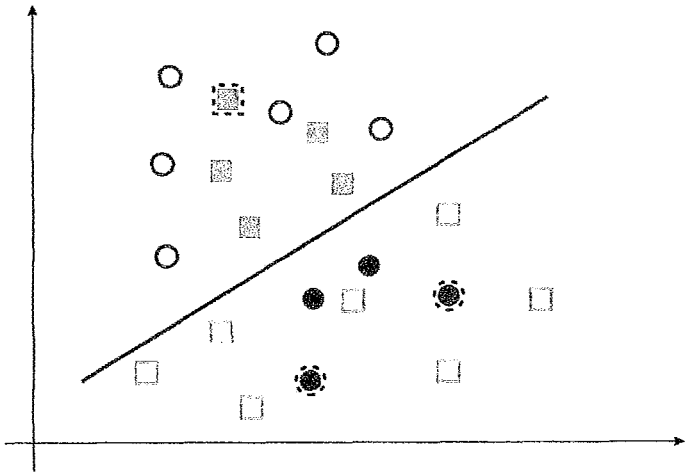


Figura 3.3 - Solução do problema não linearmente separável ($C = 10^{-8}$)

3.3 – Transformação do Problema

O problema anterior apresenta uma dificuldade intrínseca em suas restrições, já que estas apresentam uma particularidade de serem não diferenciáveis. Buscando eliminar esta dificuldade será utilizada a técnica de suavização hiperbólica (XAVIER, 1993). Essa técnica tem por objetivo transformar equações não diferenciáveis em diferenciáveis. Assim, aplicando essa técnica no problema anterior, esse poderá ser resolvido por qualquer método de otimização que faça uso de derivadas primeiras ou segundas. Esses métodos, como são amplamente difundido na literatura, têm desempenhos computacionais superiores, tanto analisando o critério de robustez

(capacidade de produzir soluções corretas) como de eficiência (capacidade de produzir soluções rapidamente).

Entretanto, a transformação do problema (3.10) em um problema diferenciável requer algumas mudanças em sua definição. Neste tópico, o problema será preparado para posteriormente ser aplicada a técnica de suavização.

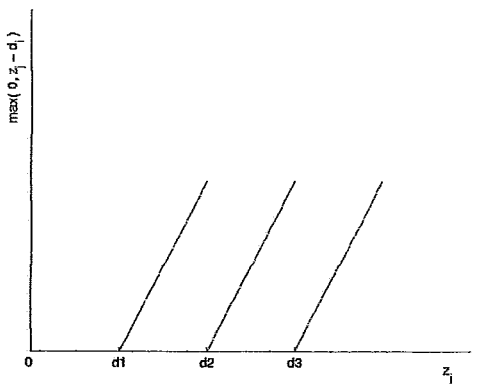


Figura 3.4 - Somatório da lado direito das restrições (3.17).

Primeiramente, considere a função φ definida da seguinte forma:

$$\varphi(y) = \max(0, y). \tag{3.15}$$

Observe que pelas restrições 1 e 2, Z_i, $i = 1,2$ representa a menor distância entre a respectiva classe e o hiperplano separador, como mostra a Figura 3.2. Assim, estas restrições podem ser reescritas na forma de desigualdades, transformando o problema original como segue:

$$\begin{aligned} &\underset{x, Z_1, Z_2}{\text{minimizar}} -(Z_1 + Z_2) \\ &s.a \\ &\quad Z_1 - s_j^t x \le 0, \quad j \in J_1 \\ &\quad Z_2 + s_j^t x \le 0, \quad j \in J_2 \\ &\quad \|x\|_2 = 1 \end{aligned} \tag{3.16}$$

Com o auxílio da função φ as restrições do problema (3.16) podem ser escritas na forma:

$$\sum_{j\in J_1}\varphi\left(Z_1-s_j^t x\right)=0$$
$$\sum_{j\in J_2}\varphi\left(Z_2+s_j^t x\right)=0 \tag{3.17}$$

Considerando a primeira restrição do problema (3.16) e assumindo $J_1=\{1,2,3,...,m_1\}$ e $d_1<d_2<\ldots<d_{m_1}$ com $d_j=s_j^t x,\ j\in J_1$, a Figura 3.4 ilustra os três primeiros somatórios do lado esquerdo dessa restrição como uma função de Z_1, como na figura anterior.

Assim, substituindo as restrições do problema (3.16) definidas em (3.17), respectivamente, obtém-se o problema transformado abaixo:

$$\begin{aligned}
&\underset{x,Z_1,Z_2}{\text{minimizar}}\ -\left(Z_1+Z_2\right)\\
&s.a\\
&\sum_{j\in J_1}\varphi\left(Z_1-s_j^t x\right)=0\\
&\sum_{j\in J_2}\varphi\left(Z_2+s_j^t x\right)=0\\
&\|x\|_2=1
\end{aligned} \tag{3.18}$$

3.4 – Suavização do Problema

Embora o problema (3.18) seja de dimensão muito menor, quando comparado ao problema (3.2) de Mangasarian, todas as suas restrições continuam sendo não diferenciáveis, o que torna a obtenção de sua solução, como dito anteriormente, altamente difícil. Neste tópico, o problema será suavizado de forma a obter um problema completamente diferenciável (XAVIER *et al.*, 2006).

Sobre esse ponto de vista, considere a seguinte função:

$$\phi(y,\tau)=\left(y+\sqrt{y^2+\tau^2}\right)\Big/2 \tag{3.19}$$

para $y\in\Re$ e $\tau>0$.

Esta função possui as seguintes propriedades:

1. $\phi(y,\tau) > \varphi(y), \quad \forall \tau > 0$;
2. $\lim_{\tau \to 0} \phi(y,\tau) = \varphi(y)$;
3. $\phi(.,\tau)$ é uma função C^{∞} convexa e crescente.

Pela segunda propriedade da equação (3.19), pode ser observado que para $\tau = 0$ a função $\phi(y,\tau)$ é equivalente à função $\varphi(y)$, visto que para $y \leq 0$ tem-se $\phi(y,0) = 0$ e para $y > 0$ obtém-se $\phi(y,0) = y$. Entretanto, para $\tau > 0$, como descrito na terceira propriedade da função, a função $\phi(.,\tau)$ é C^{∞}, enquanto que a função $\varphi(y)$ diferenciadamente é não diferenciável.

A Figura 3.5 exibe simultaneamente o gráfico dos três primeiros somatórios da primeira restrição do problema (3.18) e suas aproximações suavizadas dadas pela função $\phi(y,\tau)$ da equação (3.19).

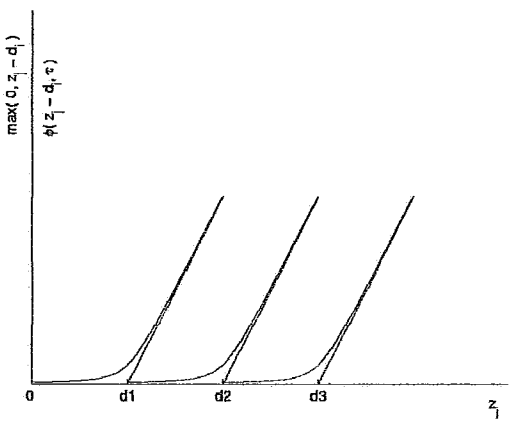


Figura 3.5 - Somatório original e suavizado das restrições do problema.

Substituindo a função $\varphi(y)$, não diferenciável, pela função $\phi(y,\tau)$ e incluindo uma perturbação $\phi(.,\tau)$ no lado direito das restrições é obtido um problema completamente diferenciável, abaixo definido:

$$
\begin{aligned}
&\underset{x, Z_1, Z_2}{\text{minimizar}} -(Z_1 + Z_2) \\
&s.a \\
&\sum_{j \in J_1} \phi(Z_1 - s_j^t x,\ \tau) \leq \varepsilon \\
&\sum_{j \in J_2} \phi(Z_2 + s_j^t x,\ \tau) \leq \varepsilon \\
&\|x\|_2 = 1
\end{aligned}
\tag{3.20}
$$

3.5 – Resolução do problema

Definindo os parâmetros τ e ε no problema (3.20) obtém-se o problema (3.18). Assim, a solução do problema (3.18) pode ser obtida resolvendo várias formulações do problema (3.20) decrescendo os parâmetros $\tau \to 0$ e $\varepsilon \to 0$. Definindo-se um processo iterativo no qual em cada iteração os parâmetros τ e ε são reduzidos, tendendo a zero, tem-se a garantia de obter o problema original.

Sendo a função $\phi(.,\tau)$ crescente e convexa (propriedade 3), as restrições do problema (3.20) serão certamente ativas para qualquer vetor x. Assim, o problema anterior será equivalente ao problema de dimensão $(n+2)$ descrito abaixo.

$$
\begin{aligned}
&\underset{x, Z_1, Z_2}{\text{minimizar}} -(Z_1 + Z_2) \\
&s.a \\
&h_1(Z_1, x) = \sum_{j \in J_1} \phi(Z_1 - s_j^t x,\ \tau) - \varepsilon = 0 \\
&h_2(Z_2, x) = \sum_{j \in J_2} \phi(Z_2 + s_j^t x,\ \tau) - \varepsilon = 0 \\
&\|x\|_2 = 1
\end{aligned}
\tag{3.21}
$$

Entretanto, o problema (3.21) possui uma estrutura separável, pois cada variável Z_1 e Z_2 aparece em somente uma restrição de igualdade. Desta forma, já que as derivadas parciais de cada restrição $h_1(Z_1, x)$ e $h_2(Z_2, x)$ em relação a Z_1 e Z_2, respectivamente, não é igual a zero, é possível usar o Teorema da Função Implícita (Apêndice A) para calcular Z_1 e Z_2 como função de x.

Assim, o problema

$$\begin{aligned} &\min f(x) = -\left(Z_1(x) + Z_2(x)\right) \\ &s.a \\ &\quad \|x\|_2 = 1 \end{aligned} \tag{3.22}$$

é obtido, onde $Z_1(x)$ e $Z_2(x)$ resultam do cálculo das raízes das equações a seguir.

$$h_i(Z_i, x) = \sum_{j \in J_1} \phi\left(Z_i - s_j^t x,\ \tau\right) - \varepsilon = 0 \qquad i = 1, 2. \tag{3.23}$$

Observando a terceira propriedade da função de suavização hiperbólica, cada termo ϕ é intrinsecamente crescente com a variável Z_i. Assim, a equação tem somente uma raiz.

Novamente, pelo Teorema da Função Implícita, as funções $Z_i(x)$, $i = 1, 2$, têm todas as derivadas em relação ao vetor x. Logo, é possível calcular o gradiente da função objetivo do problema (3.22) como segue:

$$\nabla f(x) = \sum_{i=1}^{2} \nabla Z_i(x) \tag{3.24}$$

onde

$$\nabla Z_i(x) = -\nabla h_i(Z_i, x) \Big/ \frac{\partial h_i(Z_i, x)}{\partial Z_i}. \tag{3.25}$$

Primeiramente, deve ser observado que a função objetivo do problema (3.22) é uma função homogênea de grau um do vetor x, com norma Euclidiana $\|x\|_2$. Então, é possível substituir a restrição de igualdade do problema (3.22) por duas restrições de desigualdade:

$$\begin{aligned} g_1(x) &= u - \|x\|_2 \geq 0, \\ g_2(x) &= \|x\|_2 - l \geq 0, \end{aligned} \tag{3.26}$$

para todo $0 < l < u$. Com essa substituição acima, obtém-se o problema:

$$\begin{aligned} &\underset{x}{\text{minimizar}}\ f(x) = -\left(Z_1(x) + Z_2(x)\right) \\ &s.a \\ &\quad g_1(x) = u - \|x\|_2 \geq 0, \\ &\quad g_2(x) = \|x\|_2 - l \geq 0. \end{aligned} \tag{3.27}$$

Se o problema for linearmente separável, a primeira desigualdade será ativa, e, já que a segunda é não ativa, desprezando-a, tem-se assim um problema essencialmente convexo. Para os problemas não linearmente separáveis a segunda desigualdade é que estará ativa, obtendo-se um problema de natureza não convexa.

Finalmente, o problema (3.27) definido pela suavização hiperbólica é um problema de programação não-linear com restrições de desigualdade, portanto, poderia ser solucionado por qualquer método existente na literatura de otimização. Será definida no próximo tópico a metodologia de penalização hiperbólica, que será a utilizada na resolução do problema definido anteriormente.

A vantagem dessa escolha é que há uma natural articulação entre os parâmetros de suavização e de penalização, como será apresentado.

3.6 – Penalização Hiperbólica

O Método de Penalização Hiperbólica, definido de maneira mais completa em (XAVIER, 1982), é utilizado para solucionar problemas gerais de otimização sujeito a restrições de desigualdade. Esse método é uma alternativa aos demais métodos difundidos na literatura para a solução de problemas de programação não linear. Em geral esses problemas estão na forma:

$$\begin{aligned} &\textit{minimizar} \quad f(x) \\ &s.t. \quad g_i(x) \geq 0,\ i = 1, \ldots, m \end{aligned} \tag{3.28}$$

onde $f : \Re^n \to \Re$ e $g_i : \Re^n \to \Re$, $i = 1, \ldots, m$.

Essa metodologia incorpora as restrições do problema de otimização à função objetivo utilizando uma função penalidade. Assim, é gerado um problema irrestrito que tem uma função objetivo modificada ou penalizada, como segue:

$$\text{minimizar} \quad f(x)+\sum_{i=1}^{m}\overline{P}(g_i(x)) \tag{3.29}$$

onde o segundo termo é o termo de penalização.

A função de penalização é pertencente a classe de funções C^{∞} e é definida por:

$$\overline{P}(y,\alpha,\rho)=-(\frac{1}{2}\tan\alpha)y+\sqrt{(\frac{1}{2}\tan\alpha)^2y^2+\rho^2}\,, \tag{3.30}$$

onde $\alpha\in[0,\pi/2)$ e $\rho>0$. Essa função, como mostra a Figura 3.6, possui uma assíntota horizontal e uma inclinada com ângulo α, interceptando em ρ com o eixo das ordenadas.

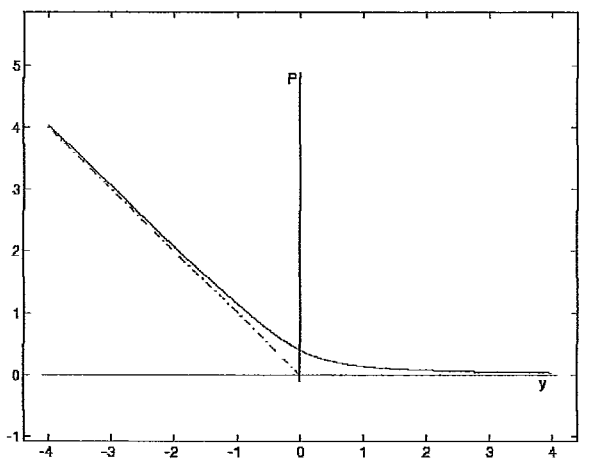


Figura 3.6 – Função de Penalização Hiperbólica

De forma alternativa, a função de penalização hiperbólica pode ser definida por:

$$\overline{P}(y,\lambda,\rho)=-\lambda y+\sqrt{\lambda^2y^2+\rho^2} \tag{3.31}$$

com $\lambda\geq 0$ e $\rho\geq 0$.

O novo problema que será otimizado possui a nova função objetivo modificada, sendo formulado como segue:

$$F(x, \lambda^k, \rho^k) = f(x) + \sum_{i=1}^{m} P(g_i(x), \lambda^k, \rho^k). \tag{3.32}$$

É importante observar que fixando o parâmetro $\rho = 0$, a resolução do problema (3.32) é equivalente à resolução do problema original (3.28), conforme demonstrado em (XAVIER, 1982). Assim, para solucionar o problema (3.32) é gerada uma seqüência de subproblemas intermediários, $k = 1, 2, 3, \ldots$, de forma que os parâmetros λ e ρ decresçam a cada iteração k.

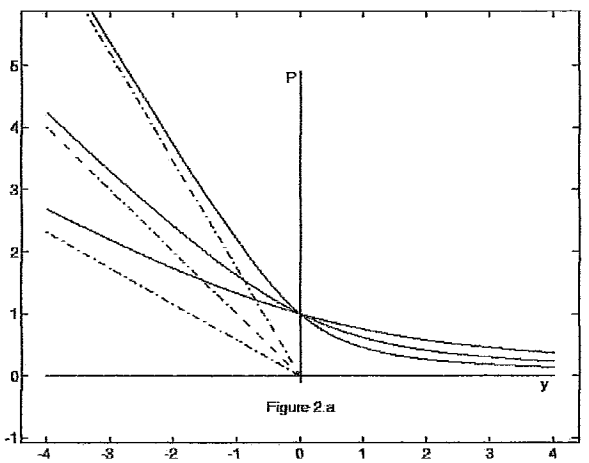


Figura 3.7 - Variação do parâmetro λ.

Cada um dos parâmetros da função penalidade trabalha em uma região do espaço onde o problema está definido. O parâmetro λ trabalha sobre a região inviável, de forma que o seu acréscimo eleva a penalidade sobre os pontos inviáveis, levando-os para a região viável, Figura 3.7. Por outro lado, o decréscimo do parâmetro τ faz com que o valor da penalidade decresça assintoticamente a zero, conforme na Figura 3.8.

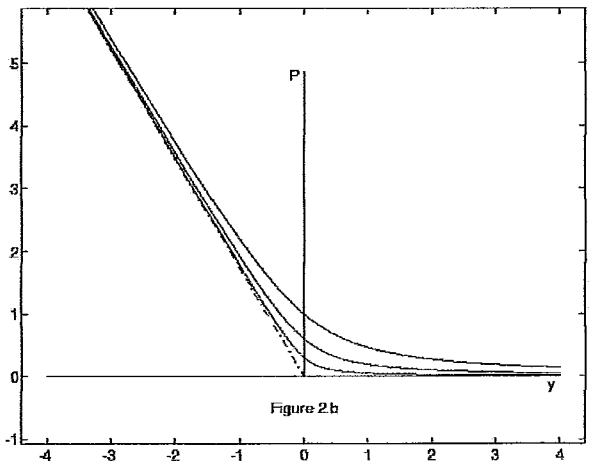


Figura 3.8 - Variação do parâmetro ρ.

A maior parte dos métodos de penalização utilizam apenas um parâmetro. Já a penalização hiperbólica, como descrito anteriormente, trabalha com dois parâmetros de penalização. Assim, o algoritmo da penalização hiperbólica, descrito a seguir, deve, naturalmente, manipular esses dois parâmetros.

Algoritmo de Penalização Hiperbólica

1: Faça $k = 0$ e x^0, $\lambda^1 > 0$, $\rho^1 > 0$.

2: k := k + 1 e resolva o problema: **minimizar**$_x$ $F(x, \lambda^k, \rho^k)$

3: Se x^k é um ponto inviável então faça

$$\lambda_i^{k+1} := r\lambda_i^k, \quad para\ r > 1$$

e retorne ao passo 2.

4: Senão faça

$$\rho_i^{k+1} = q\rho_i^k, \quad para\ 0 < q < 1$$

e retorne ao passo 2.

De acordo com a publicação original do método de penalização hiperbólica (XAVIER, 1982) o algoritmo funciona da seguinte maneira:

"A seqüência de subproblemas é obtida pela variação controlada dos dois parâmetros, λ e ρ, em duas diferentes fases do algoritmo. Inicialmente, se aumenta o parâmetro λ, causando um aumento significativo na penalização fora da região viável e, ao mesmo tempo, uma redução significativa na penalização para os pontos

dentro da região viável. Esse processo continua até que se obtenha um ponto viável. Daí em diante, mantém-se λ constante e se diminui ρ seqüencialmente. Dessa maneira, a penalização interna fica cada vez mais irrelevante, mantendo o mesmo nível de proibição na região externa."

Articulando a suavização hiperbólica com a técnica de penalização hiperbólica descrita anteriormente, o problema de otimização (3.27) pode ser redefinido completamente irrestrito como segue:

$$\min F(x,\lambda,\rho) = -\left(Z_1(x)+Z_2(x)\right)+\sum_{i=1}^{2} P\left(g_i(x),\lambda,\rho\right) \tag{3.33}$$

onde $P(y,\lambda,\rho) = -y\lambda+\left((y\lambda)^2+\rho^2\right)^{1/2}$.

Desta maneira, o problema (3.33) é facilmente resolvido utilizando-se qualquer método que se baseie em informações das derivadas de primeira ordem da função ou segunda ordem, que são mais robustas, como registrado amplamente na literatura, por exemplo (MINOUX, 1986). É importante observar que o problema (3.27) é definido somente no espaço de dimensão n, dessa forma, o conjunto de transformações efetuadas gerou um problema de otimização com a mais completa independência do número de observações m.

3.7 – Algoritmo SHSVM simplificado

Primeiro passo:

- Escolha os valores iniciais para x^0, τ^1, ρ;
- Escolha um valor fixo para o parâmetro λ da penalização hiperbólica suficientemente alto;
- Escolha os valores: $0<q_1<1$, $0<q_2<1$, $0<q_3<1$. Faça $k=0$.

Passo principal: Repita até obter o Critério de Parada

- Resolva o problema (3.33) com $\tau=\tau^k$ e $\varepsilon=\varepsilon^k$, começando no ponto inicial x^{k-1}, sendo x^k a solução obtida.
- Faça $\tau^{k+1}=q_1\tau^k$, $\varepsilon^{k+1}=q_2\varepsilon^k$, $\rho^{k+1}=q_3\rho^k$. Faça $k=k+1$.

A solução do problema SVM é obtida resolvendo uma seqüência infinita de problemas de otimização (passo principal), no presente caso, uma seqüência de subproblemas de minimização irrestritos.

Note que o algoritmo faz com que τ e ε se aproxime de 0, tornando o problema (3.33) equivalente ao problema original.

Capítulo 4 – Implementação e Resultados Computacionais

Neste capítulo serão descritos alguns problemas utilizados para validar e testar a confiabilidade e a eficiência da nova metodologia apresentada no capítulo 3. Além disso, serão feitos testes comparativos com os resultados produzidos pelo *SVMLight*, que é, atualmente, um dos softwares mais consagrados e eficientes na solução de problemas de classificação. Os experimentos numéricos com conjuntos de dados de pequeno, médio e grande portes foram realizados em um PC, AMD Sempron 1.67 GHz, 1,00 Gb de memória RAM.

No algoritmo proposto, conforme descrito no capítulo 3, a solução é obtida através da resolução de uma seqüência de subproblemas irrestritos. As minimizações irrestritas foram realizadas por meio de um algoritmo Quase-Newton, com fórmula de atualização BFGS, disponibilizado pela Harwell Library (MURPHY *et al.*, 1992). Toda a implementação foi codificada na linguagem Fortran 77, e compilado utilizando-se o compilador DIGITAL Visual Fortran versão 6.0A.

4.1 – Implementação

O primeiro passo do algoritmo é determinar o ponto inicial e a os parâmetros iniciais para o processo de otimização. Existem vários critérios para a escolha do ponto inicial, dentre eles, o mais utilizado é determinar o ponto inicial através dos centros de gravidade das duas classes. Assim, o hiperplano inicial é obtido calculando-se a metade da distância entre os centros de gravidade de cada umas das classes, como na Figura 4.1.

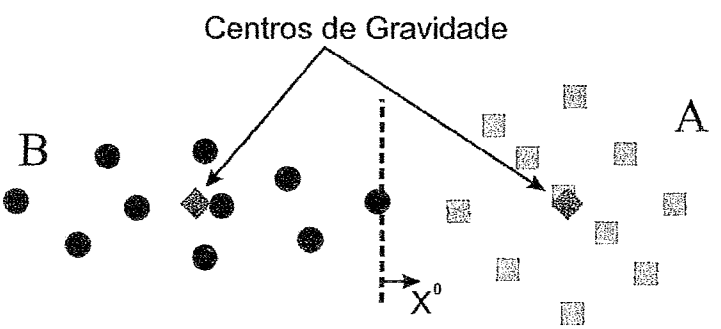


Figura 4.1 - Centros de gravidade e hiperplano inicial

Como observado no capítulo 3, que descreve a metodologia utilizada para a solução do problema de classificação, existe um conjunto de sete parâmetros associados ao algoritmo SHSVM. O parâmetros τ e ε são do algoritmo de suavização hiperbólica, os parâmetros λ e ρ do algoritmo de penalização hiperbólica e os parâmetros q_1, q_2, q_3 são fatores de redução vinculados a cada um dos parâmetros anteriores. É natural se imaginar que esses sete parâmetros devam ser especificados de maneira harmônica para a obtenção de bons resultados.

A harmonização dos parâmetros é o principal problema encontrado durante a implementação computacional. Trata-se de um trabalho árduo de tentativa e erro, na busca por intervalos numéricos que definam a melhor conformação dos parâmetros.

Em virtude da complexidade na definição dos parâmetros, será feita a seguir uma explanação sobre o papel e o funcionamento de cada um deles.

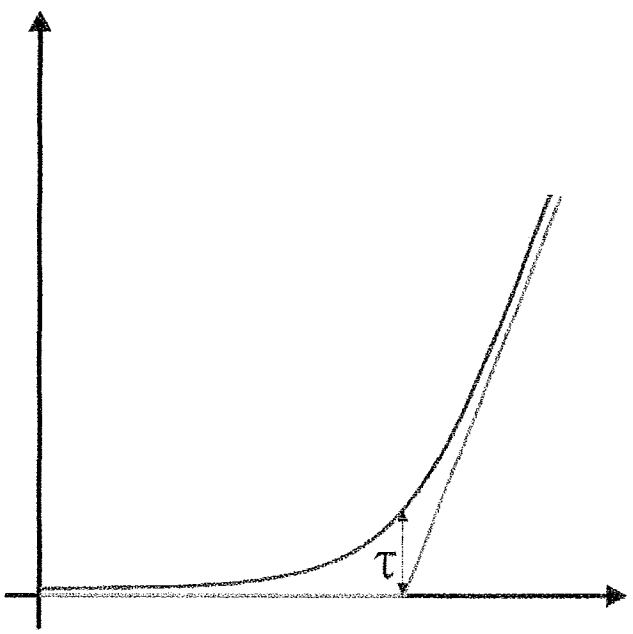


Figura 4.2 – Parâmetro τ.

O parâmetro τ está diretamente relacionado ao nível de suavização do problema e sua variação é exibida, de forma ilustrativa, na Figura 4.1. Como pode ser observado, quanto menor for esse parâmetro, melhor será a aproximação da função suavizada à função original. Assim, para valores muito pequenos, o problema suavizado tem um comportamento nervoso próximo ao ponto de não diferenciabilidade, associado à variação muito forte da derivada primeira, ou seja, aos valores muito grandes da

derivada segunda. Por outro lado, a escolha de valores grandes implica em um distanciamento excessivo do problema original.

O parâmetro de tolerância ε, relacionado à suavização hiperbólica, está naturalmente ligado ao parâmetro τ, vide equação (3.20). A manipulação desse parâmetro reflete na rigidez do hiperplano, de forma que o acréscimo de ε proporciona uma certa elasticidade ao hiperplano, pois um número maior de pontos interferirá de forma mais significativa na sua definição. A análise da Figura 4.3 possibilita o entendimento esse comportamento.

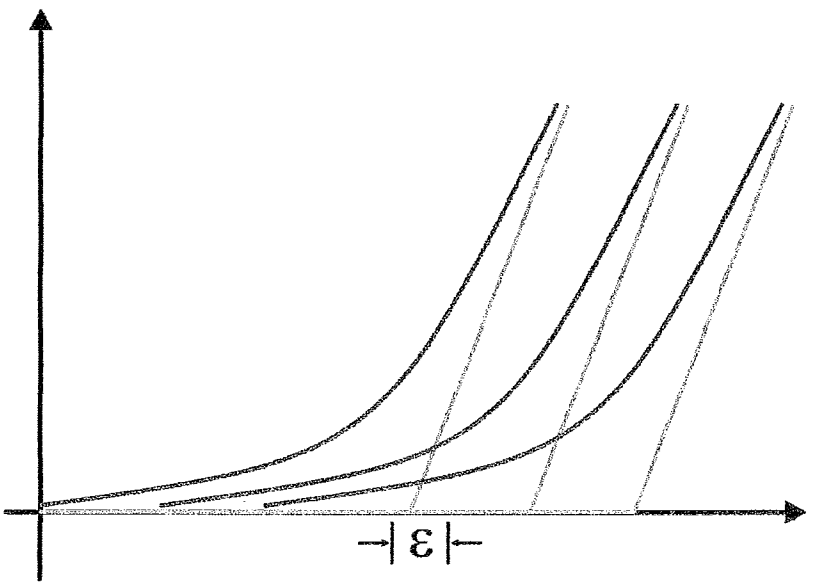


Figura 4.3 – Parâmetro ε.

O parâmetro λ da penalização hiperbólica está mais associado à penalização da violação das restrições, ou seja, produz mais efeito fora da região viável.

O parâmetro ρ da penalização hiperbólica está associado à relaxação das restrições e produz um maior efeito dentro da região viável. Assim, quando $\rho = 0$, as duas restrições, definidas em (3.26), são obedecidas rigorosamente.

Como dito anteriormente, a definição dos parâmetros do algoritmo é específica para cada problema, onde a quantidade de observações, a dimensão do espaço de entrada e a distribuição das observações nesse espaço são fatores que influenciam fortemente essa definição. Cada problema apresenta uma melhor performance para um determinado conjunto específico de parâmetros, ou seja, não é possível determinar *a priori* uma conformação ideal que produza o melhor resultado.

Os experimentos mostraram que o parâmetro τ pode ser definido de acordo com a dimensão do problema, sendo adequado se tomar um valor inicial de τ no intervalo $\|x\|_2/\sqrt{n} \leq \tau \leq \|x\|_2/n$.

O parâmetro de tolerância ε tem uma forte associação com o parâmetro de suavização τ, conforme equação (3.20). Uma relação que se mostrou apropriada foi tomar $\tau/2 \leq \varepsilon \leq 10\tau$.

O valor do λ foi escolhido simplesmente para produzir ponto viável já na primeira iteração. Em termos práticos, implica em o algoritmo de penalização hiperbólica só executar a segunda fase do mesmo. O valor suficientemente alto para tal propósito foi $\lambda = 1$.

Como dito anteriormente o parâmetro ρ da penalização hiperbólica também está diretamente ligado à τ. Os valores iniciais mais adequados para esse parâmetro da penalização hiperbólica foram obtidos fixando $\rho = \tau^2$.

A suavização hiperbólica por ser uma técnica que implica na resolução de uma seqüência de subproblemas, carece dos parâmetros q_1 e q_2, denominados por *fatores de redução*, responsáveis por aproximar, a cada iteração, o subproblema suavizado solucionado do problema original. Como visto no capítulo 3, a redução dos parâmetros τ e ε aproxima o problema suavizado ao problema original.

Analogamente, o método da penalização hiperbólica carece de um fator q_3 para redução do parâmetro ρ, para que seja provida a obediência rigorosa às restrições.

Nos experimentos computacionais, demonstrou-se eficaz a escolha $q_1 = q_2 = q_3$. A manipulação dos fatores de redução tem dois compromissos. Um valor excessivo para esses fatores prejudica a continuidade harmoniosa dos processos aproximativos. Essa falta de harmonia será refletida em iterações intermediárias mais demoradas. Por outro lado, valores pequenos aumentam desnecessariamente o número total de iterações.

A Tabela 4.1 apresenta, para cada um dos problemas testados, os parâmetros e os fatores de redução utilizados para cada um deles. A escolha dos valores dos fatores de redução mostrou-se bem sensível na geração dos resultados. Os valores mais adequados para os problemas teste estão tabulados, dentro da faixa $1.2 \leq q \leq 2.0$.

A análise dos resultados permitiu se observar que problemas com margem menor necessitam de um fator de redução menor, enquanto que problemas que possuem margem maior permitem um fator de redução maior. Isso se deve ao fato dessa técnica solucionar uma seqüência de subproblemas que são aproximações do problema original.

Tabela 4.1 - Variação dos parâmetros

Problemas Padrão			
Base de Dados **m x n**	**Fator de Redução**	τ	ε
Iris 150 x 4	1.2	$\|x\|_2/n$	10τ
Letras 1555 x 16	1.5	$\|x\|_2/\sqrt{n}$	10τ
Leukemia 38 x 50	1.2	$\|x\|_2/n$	10τ
Ann-Thyroid 284 x 21	1.2	$\|x\|_2/n$	$\tau/2$
Problemas Aleatórios			
Base 200 200 x 10	2.0	$\|x\|_2/n$	$10\tau/2$
Base 2000 2000 x 10	2.0	$\|x\|_2/n$	10τ
Base 20000 20000 x 10	1.5	$\|x\|_2/n$	10τ
Base 200000 200000 x 10	2.0	$\|x\|_2/n$	10τ

Para problemas de difícil solução, com margem pequena, os subproblemas intermediários podem tender a uma solução rasa, ou seja, um mínimo local. Entretanto, caso o fator de redução seja pequeno, a obtenção da solução final é feita de forma lenta, diminuindo a possibilidade de se obter um mínimo local raso durante a resolução da seqüência de subproblemas. Entende-se como mínimo local raso um ponto em que o valor da função objetivo assume um valor relativamente alto. Assim, quanto menor a margem maior a dificuldade de se obter a solução e o processo de convergência é mais lento.

4.2 – SVMLight

SVMLigth é um software desenvolvido há cerca de dez anos, para solucionar problemas de classificação SVM. A formulação básica utilizada por esse software baseia-se na metodologia desenvolvida por Vapnik, descrito no capítulo 2.

O problema de otimização quadrática definido por Vapnik, equação 2.13, pode ser reescrito, de maneira equivalente, utilizando-se notação matricial. Assim, seja Q uma matriz, tal que $(Q)_{ij} = y_i y_j k(x_i, x_j)$, i,j = 1, ..., m e seja m o número de observações. A formulação matricial é dada por:

$$\begin{aligned} \text{minimizar} \quad & W(\alpha) = -\alpha^T 1 + \frac{1}{2}\alpha^T Q\alpha \\ s.a \quad & \alpha^T y = 0, \\ & 0 \le \alpha \le C1, \end{aligned} \tag{4.1}$$

sendo k, como dito no capítulo 2, uma função *kernel*, que indica o grau de similaridade ou de dissimilaridade entre os dados do conjunto de treinamento e o vetor de entrada. C, que corresponde a um fator de tolerância que proporciona uma maior flexibilidade ao classificador, permitindo que algumas observações do conjunto de treinamento possam violar a margem.

Com o auxílio da formulação (4.1) pode-se observar que a solução do problema para grandes conjuntos de treinamento é impraticável, dado ao seu grande porte. Isso se deve ao fato da matriz Q não poder ser armazenada na memória para os problemas grandes.

Atualmente, muitas implementações necessitam armazenar toda a matriz Q na memória durante o treinamento do classificador, tornando-se necessário limitar o tamanho do conjunto de treinamento para que seja possível armazenar toda a matriz Q.

Tentando fugir da limitação física do armazenamento de toda a matriz Q, algumas aplicações têm como alternativa calcular a matriz Q em tempo de execução, de forma que esta não precisaria ficar armazenada na memória. Embora possa em princípio funcionar, essa alternativa de solução torna-se muito lenta com o aumento da matriz Q.

Outra alternativa para problemas com grande conjunto de treinamento é decompor o problema original em uma série de subproblemas menores. O SVMLight utiliza esse estratégia, baseado na idéia de decomposição dada por Osuna (JOACHIMS,

1998). O objetivo dessa decomposição é dividir o conjunto de treinamento em dois conjuntos: um conjunto inativo e outro ativo, que é denominado *conjunto de trabalho*. A principal vantagem é que o problema torna-se linear em memória, para as observações do conjunto de treinamento, e linear no número de vetores suporte (SV). Uma desvantagem é que, caso sejam selecionados conjuntos de trabalho inadequados, o algoritmo irá convergir muito lentamente para a solução ótima.

Visando solucionar os problemas da decomposição, o algoritmo desenvolvido no SVMLight incorpora as seguintes alternativas:

1. Uma técnica eficiente e correta para a definição do conjunto de trabalho que será utilizado pelo algoritmo;
2. Sucessivas reduções no conjunto de treinamento a fim de facilitar o encontro do conjunto de trabalho. Essa redução é feita baseando-se em dois conceitos:
    a. O número de SVs é bem menor que o número de observações;
    b. Vários SVs têm α_i limitado superiormente por C.
3. Melhorias computacionais como atualizações incrementais do gradiente, critérios de parada e uso de *caching*.

Serão descritas nesse trabalho algumas dessas alternativas que tornam o *SVMLight* um classificador capaz de solucionar problemas de grande escala com relativa rapidez. Maiores detalhes e outras alternativas do *SVMLight* podem ser encontrados no trabalho do Joachims (JOACHIMS, 1998).

4.2.1 – Decomposição e Redução do Conjunto de Treinamento

A decomposição do conjunto de treinamento é feita a cada iteração. Em cada iteração as variáveis duais α_i são explicitadas em um conjunto de variáveis livres B, que podem ser atualizadas durante a iteração, e em um conjunto de variáveis fixas N, que não pode ser atualizado. O conjunto de variáveis livres B é denominado por conjunto de trabalho.

O algoritmo soluciona, a cada iteração, o problema (4.1) utilizando um dado conjunto de trabalho, em princípio de pequena dimensão. Já que o problema (4.1) tem *Hessiana* da matriz Q semi-definida positiva e todas as restrições lineares, o problema é convexo. Dessa forma, as condições de *Karush-Kuhn-Tucker* (KKT) são condições

necessárias e suficientes para garantir a otimalidade da solução encontrada. Assim, esse problema é solucionado com o conjunto de trabalho e, a cada iteração, as condições de KKT são analisadas para avaliar a solução encontrada. Caso a solução atenda às condições de KKT, o processo termina e a solução ótima foi encontrada. Caso contrário, é definido um novo conjunto de trabalho e uma solução é novamente obtida.

Deve ser ressalvado que o processo de convergência pode ser extremamente lento, caso os conjuntos de trabalho sejam escolhidos de forma ineficiente. Logo, é necessário definir critérios para selecionar bons conjuntos de trabalho, que na iteração corrente produzam maiores progressos em direção ao mínimo de $W(\alpha)$. A idéia é encontrar uma direção de descida $\mathbf{d}$, viável, que tenha q elementos não nulos. As variáveis α_i que correspondem a esses elementos definem o conjunto de trabalho. Essa técnica é baseada em Zoutendijk's (JOACHIMS, 1998).

A direção de descida $\mathbf{d}$ é obtida através do seguinte problema de otimização:

$$\begin{aligned} \text{minimizar} \quad & V(\mathbf{d}) = g(\alpha^t)^T \mathbf{d} \\ s.a \quad & y^T \mathbf{d} = 0 \\ & d_i \geq 0 \quad para \ i : \alpha_i = 0 \\ & d_i \leq 0 \quad para \ i : \alpha_i = C \\ & -1 \leq \mathbf{d} \leq 1 \\ & \left| \{ d_i : d_i \neq 0 \} \right| = q \end{aligned} \tag{4.2}$$

A função objetivo define que se deseja uma direção de descida. A direção de descida tem um produto interno negativo com o vetor das derivadas parciais $g(\alpha^t)$ no ponto α^t. As três primeiras restrições garantem que a direção de descida é projetada ao longo da direção da restrição de igualdade do problema (4.1). A quarta restrição normaliza a direção de descida tornando o problema bem definido. Já a quinta restrição garante que a direção de descida terá somente q elementos não nulos. E esses elementos não nulos de $\mathbf{d}$ são adicionados ao conjunto de trabalho.

Como na grande maioria dos casos, o número de vetores suporte é menor que o número de observações do conjunto de treinamento, o algoritmo do *SVMLight* incorporou mais uma alternativa para reduzir o conjunto de trabalho e com isso aumentar sua eficiência. Caso se conhecessem a princípio quais são os vetores suporte de um determinado problema, o conjunto de trabalho poderia ser reduzido a um

conjunto contendo somente os vetores suporte, que o resultado produzido seria o mesmo encontrado utilizando-se todo o conjunto de treinamento.

Além do número de vetores suporte ser menor que a quantidade de observações do conjunto de treinamento, os problemas, em geral, possuem ruídos, que são observações que violam a margem. Caso essas observações também fossem conhecidas, o valor de α_i correspondente a cada uma dessas observações poderia ser fixado por C. Assim, o problema teria muito menos observações que o problema original e a solução também seria encontrada muito mais facilmente.

Durante o processo de otimização é fácil definir se uma observação tem alguma possibilidade de ser um vetor suporte ou não. Caso perceba-se que essa nunca será um vetor suporte, ela pode ser excluída do conjunto de treinamento (JOACHIMS, 1998). Por utilizar todas essas heurísticas, o *SVMLight* torna-se um software eficiente e robusto, solucionando problemas com grandes conjuntos de treinamento. Em função dessas características, esse software é um dos mais utilizados para solucionar problemas SVM.

4.3 – Resultados Computacionais

Para validar os resultados obtidos pela implementação da metodologia descrita no capítulo 3 foram utilizados vários problemas teste. Primeiramente, foi construído um conjunto de treinamento $\{s_j \mid s_j \in \Re^2\}$ que permitisse uma observação visual da solução obtida, Figura 4.4. Os vetores suporte calculados pelas alternativas SHSVM e *SVMLight*, foram os mesmos para os dois classificadores e são representados pelas observações circuladas.

Para se ter uma comparação mais exata entre as soluções fornecidas pelos dois softwares foram feitas duas comparações numéricas. A primeira foi entre os vetores normais ao hiperplano, que pode ser observada na Tabela 4.2. Para permitir essa verificação, o resultado do *SVMLight* foi normalizado. A segunda avalia a margem produzida pelos dois classificadores e é exibida na Tabela 4.3.

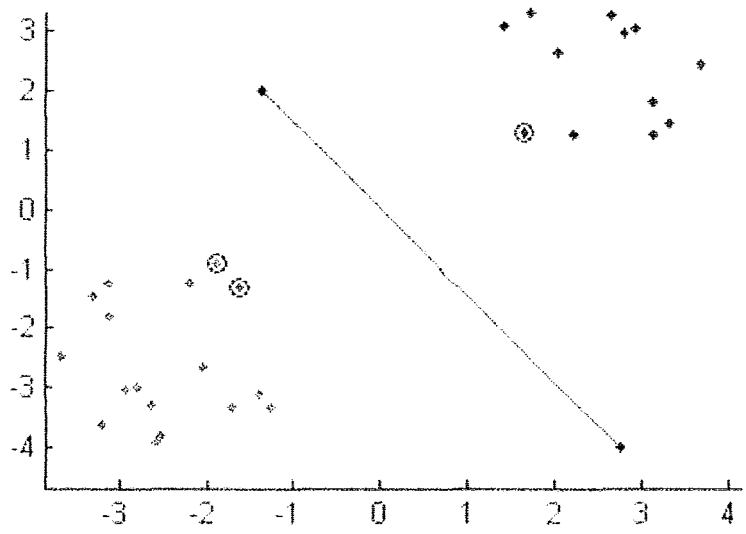


Figura 4.4 - Solução gráfica SHSVM e SVMLight

Embora a Figura 4.4 ilustre um único hiperplano separador, os resultados numéricos apresentados nas tabelas apresentam diferenças na segunda e na terceira casa decimal. Existem dois motivos que explicam essa diferença nos resultados. Primeiro, o SHSVM é uma aproximação do problema original e por isso apresenta um resultado próximo ao resultado real. Além disso, o *SVMLight* tem uma série de parâmetros que são utilizados durante o processo de otimização, sendo um deles o parâmetro C, definido como sendo um compromisso entre o erro de treinamento e a margem. Esse parâmetro permite que algumas observações violem a margem para aumentar o poder de generalização do classificador, fazendo com que a margem calculada pelo *SVMLight* seja maior que a margem exata entre as duas classes[2]. Ou seja, como a margem do SHSVM é exatamente a distância entre as duas classes, essa torna-se menor que a margem mais flexível do *SVMLight*.

Tabela 4.2 - Comparação dos resultados entre SVMLight e SHSVM

Software	x_1	x_2
SHSVM	0,825532378	0,564354759
SVMLight	0, 828623359	0,559806619

[2] O conceito de margem adotado pelo SHSVM pode ser visto no capítulo 2.

Tabela 4.3 - Comparação entre as margens

Software	Margem
SHSVM	4,1534042030
SVMLight	4,1549807832

Alguns problemas teste disponíveis na *web* foram utilizados a fim de comparar os resultados obtidos entre as duas metodologias. As bases de dados foram obtidas de *UCI Machine Learning Repository* (MURPHY *et al.*, 1992).

4.3.1 – Problemas não linearmente separáveis

A fim de analisar o comportamento do SHSVM para problemas não linearmente separáveis, selecionou-se a base de dados de diagnóstico de câncer de mama. Essa base de dados possui duas classes não linearmente separáveis entre si, uma constituída por amostras de nódulos benignos e a outra por malignos. As duas classes possuem juntas 198 observações, cada uma com 32 características.

O software SHSVM comportou-se de maneira estável buscando sempre maximizar a margem. Entretanto, há uma incompatibilidade de comparação completa dos resultados, em função da diferença entre os critérios utilizados pelos softwares SHSVM e *SVMLight*. Segundo o algoritmo proposto, o valor da margem assume um valor negativo, enquanto o *SVMLight* fornece um valor extremamente alto.

Mesmo com a impossibilidade de uma real comparação entre os resultados apresentados pelo *SVMLight* e pelo SHSVM, está sendo apresentado na Tabela 4.4 os resultados obtidos pelos dois softwares.

Tabela 4.4 - Comparação dos resultados para base não linearmente separável

Base de Dados m x n	**Algoritmo**	**Exatidão fase de treinamento (%)**	**Tempo CPU (segundos)**
wpbc 198 x 32	SHSVM	76.26	1.17
	SVMLight	76.26	11.49

Embora tanto o SHSVM, como o *SVMLight* tenham obtido a mesma exatidão na fase de treinamento do classificador, somente cerca de 62,5 % dos vetores suporte apresentados pelo SHSVM são iguais aos vetores suporte do *SVMLight*.

O tempo de processamento gasto pelo algoritmo proposto SHSVM, como pode ser visto na Tabela 4.4, na resolução desse problema não linearmente separável foi da ordem de dez vezes inferior ao tempo de execução do *SVMLight*.

4.3.2 – Problemas linearmente separáveis

4.3.2.1 – Bases de teste

Selecionou-se quatro bases que apresentam a propriedade de serem linearmente separáveis. Foram escolhidas bases cujo conjunto de treinamento fosse constituído por duas ou mais classes. As bases de dados com mais de duas classes, como a base de reconhecimento de letras, são tradicionalmente utilizadas em problemas de classificação múltipla. Para realizar os testes sobre um classificador de duas classes, foram selecionados subconjuntos desses pontos formando conjuntos de dados com duas classes apenas. Em geral, para as bases que possuem três classes foi feita uma combinação dois a dois entre as classes gerando três problemas distintos linearmente separáveis, como sugerido por Mangasarian (O.L.MANGASARIAN *et al.*, 2001).

O problema da tiróide é tradicionalmente um problema de classificação múltipla que possui três classes, sendo que as classes 1 e 2 são linearmente separável entre si, enquanto a classe 3 não é das outras duas. Assim, foi construindo um problema teste no qual o conjunto de treinamento contém somente dados da classe 1 e da classe 2. Esse problema é utilizado para determinar o estado clínico de um paciente informando se ele possui hipotireoidismo ou não. A classe 1 é referente a pessoas com hipotireoidismo e a classe 2 a pessoas com funcionamento anormal da tiróide, porém que não apresentam sintomas de hipotireoidismo. Essas duas classes possuem juntas um total de 284 observações, cada uma com 21 características.

O mesmo procedimento adotado para o problema da tiróide foi adotado para o problema de reconhecimento de letras. A base de dados de reconhecimento de letras é, na maioria das vezes, utilizada em classificadores de múltiplas classes, por se tratar de um problema no qual cada letra está associada a uma classe distinta. Assim, como foi feito por Mangasarian (O.L.MANGASARIAN *et al.*, 2001), foram selecionados dois subconjuntos, um deles com as observações referentes à letra A e outro com as observações referentes à letra B. Esses dois subconjuntos formaram um conjunto de treinamento com 1555 observações, onde cada uma delas possui 16 características.

A base de dados Fisher Íris é provavelmente a mais utilizada na literatura para problemas de reconhecimento de padrões. Essa base possui três classes, onde uma classe é sempre linearmente separável das outras duas. Cada classe é referente a uma espécie de planta e possuem um total de 150 observações e cada observação possui quatro características. Os resultados apresentados são relativos a separação da classe 1 em relação às outras duas classes. Assim, no conjunto de treinamento, a classe 1 possui os dados da classe 1 da base de dados e a classe 2 agrupa os dados das classes 2 e 3.

A base de dados Leukemia possui informações de um dos cânceres mais freqüentes e estudados, que é o câncer de mama humano. Essa base é dividida em duas classes, uma com características miomas benignos e a outra com as características de miomas malignos. No total tem-se um conjunto de treinamento com 38 observações e 50 características.

Os resultados obtidos para os quatro problemas descritos acima são exibidos na Tabela 4.5. Uma comparação exata entre as margens produzidas pelo SHSVM e o *SVMLight* não é possível, pela diferença de critérios. Entretanto, os pontos suporte obtidos pelos algoritmos são exatamente os mesmos, resultado que é indicador da validade da metodologia proposta.

Como os vetores suporte são iguais nos resultados obtidos tanto pelo algoritmo proposto SHSVM, como pelo *SVMLight*, pode-se concluir que os critérios para a definição da margem são diferentes nos dois softwares.

Tabela 4.5 - SHSVM comparado ao SVMLight

Base de Dados m x n	**Algoritmo**	**Margem**	**Exatidão fase de teste (%)**	**Tempo CPU (segundos)**
Ann-Thyroid 284 x 21	SHSVM	0. 00243384581	97.89	0.55
	SVMLight	0.002435179390	97.83	0.20
Letras 1555 x 16	SHSVM	0.116839728245	99.04	2.2
	SVMLight	0.116888042879	98.74	0.11
Iris 150 x 4	SHSVM	1.635107122655	100	0.09
	SVMLight	1.635109061774	100	0.02
Leukemia 38 x 50	SHSVM	5568.139424290	94.74	0.08
	SVMLight	5555.555555556	93.10	0.01

No que concerne à margem, a Tabela 4.5 apresenta um resultado inesperado para a base de dados Leukemia, no qual o valor fornecido pelo *SVMLight* é bem menor que o apresentado pelo SHSVM. Como esse problema, em particular, possui uma margem muito grande, a norma de x será muito pequena, já que a margem é dada por $2/\|x\|$. O *SVMLight* arredonda o seu resultado na quinta casa decimal fornecendo $\|x\| = 0.00036$. A partir do resultado do SHSVM pode-se inferir que a norma de x será aproximadamente dado por $\|x\| = 3.591864081669 * 10^{-4}$ e caso esse valor seja arredondado na quinta casa decimal será obtido exatamente o resultado do *SVMLight*. Assim, pode-se inferir que o arredondamento feito pelo *SVMLight* é refletido com um erro no valor real da margem. Ademais esse mesmo raciocínio pode explicar adicionalmente as diferenças para os demais problemas.

A Tabela 4.5 exibe também os valores obtidos para a exatidão da fase de teste, medida pelo percentual de acertos, que demonstram um desempenho superior do SHSVM em todos os quatro problemas.

Os testes foram realizados utilizando-se a técnica de *leave-one-out*, que, além de ser muito utilizada para obter a exatidão dos métodos, é a técnica utilizada pelo software *SVMLight*. Essa técnica de teste consiste em retirar uma observação do conjunto de treinamento e realizar a etapa de aprendizado sobre todos os outros $m-1$ elementos do conjunto. A observação que foi retirada é classificada sobre o hiperplano encontrado. Esse processo é repetido para cada uma das m observações. No final tem-se o número de observações que foram classificadas corretamente sobre o hiperplano criado. As tabelas deste capítulo apresentam o percentual de acertos para cada um dos problemas executados.

4.3.2.2 – Bases Aleatórias

Como pode ser observado na Tabela 4.5 o tempo de processamento do SHSVM comparativamente ao *SVMLight* aumenta com o crescimento do número de observações. Para analisar a eficiência da metodologia foram criados vários conjuntos de treinamento aleatórios com tamanhos diferentes, variando de 200 observações a 200 mil observações. Esses problemas fora criados para testar a robustez e a eficiência da nova metodologia na solução de problemas de pequeno, médio e grande porte.

Todos esses problemas testes foram criados usando o software *NDC Data Generator*, produzido por Musicant (MUSICANT, 1998). No primeiro conjunto de testes, as observações foram geradas aleatoriamente segundo uma distribuição uniforme no intervalo $]-6,8]$ gerando duas classes linearmente separáveis.

Tabela 4.6 - Comparação dos resultados para bases aleatórias

Base de Dados m x n	**Algoritmo**	**Margem**	**Exatidão fase de teste (%)**	**Tempo CPU (segundos)**
Base 200	SHSVM	15.65636759185	100	0.03
200 x 10	*SVMLight*	15.65680288085	100	0.48
Base 2000	SHSVM	3.106290251622	100	1.30
2000 x 10	*SVMLight*	3.106361829025	100	0.03
Base 20000	SHSVM	3.101742156230	100	55.90
20000 x 10	*SVMLight*	3.102843756302	100	0.50
Base 200000	SHSVM	0.025655465151	100	193.77
200000 x 10	*SVMLight*	0.025922450397	100	9.07

4.4 – Síntese dos Resultados

Os resultados apresentados anteriormente mostram que a metodologia apresentada nesse trabalho é eficaz. As margens fornecidas pelo *SVMLight* e o SHSVM são praticamente iguais para todos os problemas exceto para a base de dados Leukemia (Tabela 4.5), cuja explicação encontra-se logo abaixo da tabela.

Os resultados da Tabela 4.4 para o problema não-separável de câncer de mama, mostram a superioridade em termos de tempo do SHSVM. Entretanto, não é possível obter uma análise concreta dos resultados obtidos em função da incompatibilidade dos critérios dos dois softwares.

Os resultados exibidos na Tabela 4.6 mostram um domínio bem superior do *SVMLight* no quesito tempo. Essa diferença em relação ao tempo de processamento, está diretamente relacionada às técnicas de redução do conjunto de treinamento utilizadas pelo software *SVMLight*, que, assim, demonstram sua grande eficiência.

As margens obtidas pelos dois métodos em todos os problemas teste, a despeito das diferenças de critérios, são praticamente iguais. Além disso, em todos os testes os vetores suporte obtidos pelos dois softwares são exatamente os mesmos, indicando a validade do algoritmo proposto SHSVM e confirmando a diferença nos critérios de definição da margem.

Nos experimentos computacionais descritos neste capítulo foi realizado um trabalho de harmonização dos sete parâmetros definidos anteriormente. Embora os resultados mostrem que a combinação de valores para os parâmetros tenha sido adequada, nem todas as opções foram exauridas, ou seja, uma harmonização melhor ainda pode ser obtida.

Finalmente, os erros da fase de teste obtidos pelo SHSVM são um pouco menores do que aqueles do *SVMLight*.

Capítulo 5 – Poda

Os resultados apresentados no capítulo anterior mostraram que embora a metodologia SHSVM seja robusta, perde em eficiência para o *SVMLight*. As técnicas de redução do conjunto de treinamento utilizadas pelo software *SVMLight* produzem uma diminuição substantiva no tempo de processamento utilizado pelo software na solução dos problemas.

Buscando reduzir a diferença existente no tempo de processamento entre os dois métodos, será desenvolvida neste capítulo técnicas de redução do conjunto de treinamento. Esses critérios de poda dos dados de entrada procuram excluir os pontos que certamente não influenciam na solução do problema. Assim, o tempo de processamento é reduzido e a solução do problema não é alterada, ou seja, busca-se uma identificação precoce dos pontos não-suporte para sua imediata exclusão.

5.1 – Critério para seleção de dados

O objetivo dessa técnica é eliminar o maior número possível de pontos para que o processo de otimização seja mais eficiente. Como foi visto em capítulos anteriores, a solução do problema *SVM* só se fundamenta em informações dos pontos suporte. Desta forma, todos os outros pontos podem ser descartados sem que a solução ótima seja alterada, ou seja, o problema reduzido tem a mesma solução do original.

O algoritmo de Suavização Hiperbólica resolve a cada iteração um problema mais simples obtendo um novo hiperplano que melhor classifica os dados. Como a cada iteração tem-se um novo problema *SVM* suavizado, todos os pontos não suporte, a princípio, poderiam ser excluídos para o cálculo do novo hiperplano separador. Entretanto, não é possível saber *a priori* quais serão os pontos suporte deste novo hiperplano.

A idéia desta metodologia de seleção de dados é, em uma dada iteração k, utilizar um ângulo α^k, correspondente a variação, em princípio máxima, entre os vetores x^{k-1} e x^k de duas soluções intermediárias, como elemento norteador do procedimento de poda. A partir de uma estimativa de variação acumulada máxima desses ângulos de uma iteração qualquer até a iteração final, pode-se precisar o conjunto

de possíveis pontos candidatos a vetores suporte. A partir desta informação, os pontos pertencentes ao conjunto complementar podem ser podados.

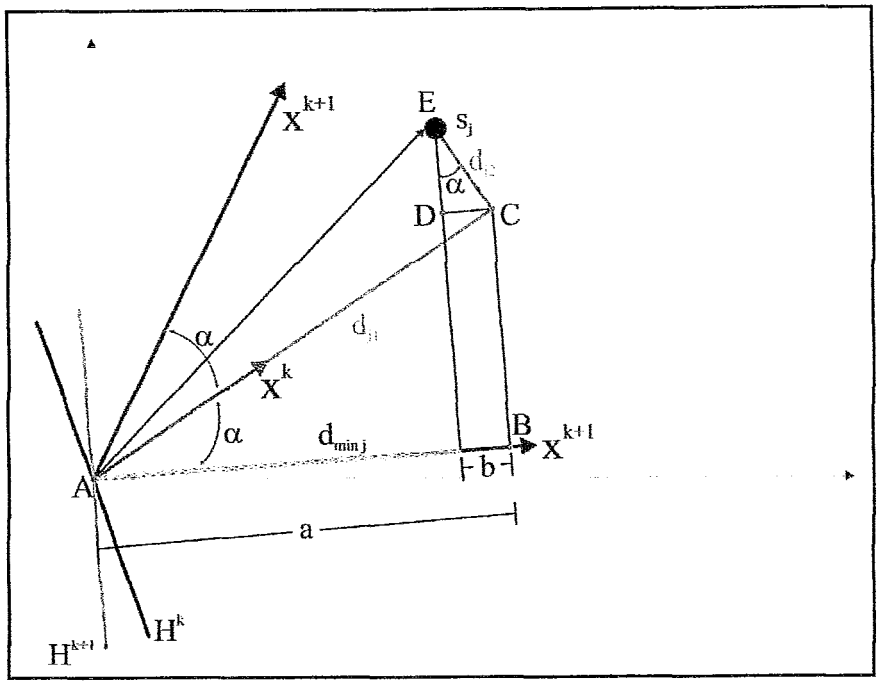


Figura 5.1 - Menor distância possível para $x_j, j \notin J^*$

5.1.1 – Cálculo da Variação Mínima para pontos não suporte

Seja J^* o conjunto dos pontos suporte associados à solução intermediária obtida em uma iteração e s_j um ponto do conjunto de treinamento tal que $j \notin J^*$. O objetivo deste método é comparar o pior caso para os pontos $s_i, i \in J^*$, com o melhor caso para os pontos $s_j, j \notin J^*$. Ou seja, deseja-se comparar a menor distância possível para todo $s_j, j \notin J^*$, com a maior distância futura para os pontos suporte. Como ilustrado pela Figura 5.1, ao se girar x^k no ângulo dado α são obtidas duas situações para o novo hiperplano H^{k+1}, já que o vetor x^k pode ser girado tanto no sentido horário como no sentido anti-horário. Dependendo do sentido de rotação de x^k o ponto s_j estará mais próximo ou mais distante de H^{k+1}. Como s_j não é um ponto suporte é analisada a opção onde o ponto fique mais próximo do novo hiperplano.

Para calcular a distância de s_j à H^{k+1}, decompõe-se esse ponto em duas direções ortogonais. A primeira é segundo a direção do vetor x^k, que é o vetor normal ao hiperplano H^k e, conseqüentemente, representa a distância entre s_j e H^k, dada pelo valor d_{j1}. A segunda é a componente residual, correspondente à diferença de s_j e a primeira direção, que, assim, é normal ao vetor x^k e é denominada por d_{j2}. Estas duas direções estão ilustradas em azul claro na figura acima. Segundo esse critério, o vetor s_j fica decomposto na forma

$$s_j = d_{j1} + d_{j2} \tag{5.1}$$

Pela ortogonalidade entre as duas componentes da decomposição acima, tem-se:

$$\|s_j\|_2^2 = \|d_{j1}\|_2^2 + \|d_{j2}\|_2^2. \tag{5.2}$$

Dessa forma, os valores da norma euclidiana da segunda componente $\|d_{j2}\|$ pode ser facilmente calculado através de:

$$d_{j2} = \sqrt{\|s_j\|_2^2 - d_{j1}^2}. \tag{5.3}$$

Com essa decomposição de s_j, são obtidos dois triângulos retângulos importantes que serão utilizados para o cálculo de uma distância entre s_j e qualquer novo hiperplano H^{k+1} dentro de um cone de abertura 2α. Analisando a Figura 5.1, essa distância é dada por $d_{\min j} = a - b$. A partir do triângulo ABC, reto em B, tem-se que $a = d_{j1} \cos \alpha$ e pelo triângulo EDC, reto em D, obtém-se $b = d_{j2} \operatorname{sen} \alpha$. Logo, a menor distância possível entre s_j e o novo hiperplano H^{k+1} é dada por:

$$d_{\min j} = d_{j1} \cos \alpha - d_{j2} \operatorname{sen} \alpha. \tag{5.4}$$

5.1.2 – Cálculo da Variação Máxima para Pontos Suporte

A mesma análise feita para os pontos não-suporte foi realizada para os pontos suporte. Como dito anteriormente, a distância entre um ponto suporte, s_i, na iteração k e o novo hiperplano H^{k+1} depende do sentido de rotação do vetor x^k. Como s_i é um ponto suporte é analisada a opção onde o ponto fique mais distante do novo hiperplano.

Assim como foi feito para $s_j, j \notin J^*$, os vetores suporte $s_i, i \in J^*$ devem ser decompostos em duas direções: na direção de x^k e em uma direção normal a x^k. Essas direções formam triângulos retângulos que podem utilizados para calcular a maior distância entre s_i e o H^{k+1}. A Figura 5.2 mostra que essa distância máxima corresponde a $d_{max\,j} = a + b$. Por trigonometria, o triângulo ACB, reto em C, fornece que $a = d_{j1} \cos \alpha$ e, a partir do triângulo BDE, reto em D, tem-se $b = d_{j2}\,\text{sen}\,\alpha$. Assim, a máxima variação sofrida, dentro de um cone de abertura 2α, por cada um dos pontos suporte é dada por

$$d_{\max j} = d_{j1} \cos \alpha + d_{j2}\,\text{sen}\,\alpha. \tag{5.5}$$

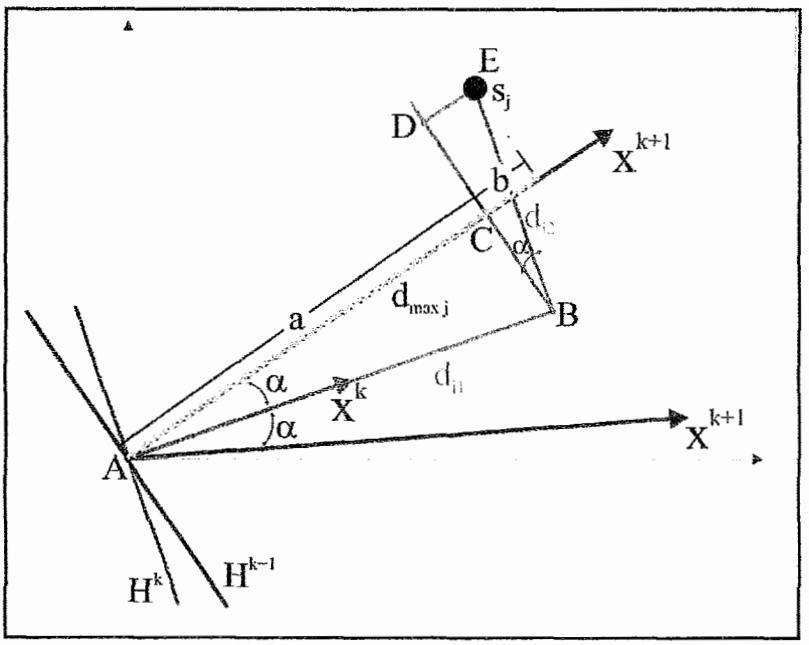


Figura 5.2 - Menor distância possível para $x_i, i \in J^*$

Após se obter as variações máximas das distâncias dos pontos suporte, calcula-se a menor variação dentre as obtidas. Essa seleção é feita, pois caso uma observação não-suporte tenha sua variação, definida em (5.4), maior que a variação mínima dos pontos suporte, ela poderá ser podada.

$$\overline{d} = \min_{i \in J^*} d_{\max i} \tag{5.6}$$

Os valores $d_{\min j}, j \notin J^*$ são calculados para todos os pontos que não são suporte e $\overline{d}$ é obtido calculando-se a menor variação dentre as máximas variações dos pontos suporte. Caso $d_{\min j} > \overline{d}$, pode ser deduzido que esse ponto s_j não será um ponto suporte de qualquer H^{k+1} dentro do cone de abertura 2α, já que a sua distância ao hiperplano, no melhor caso, ainda será maior que a distância dos pontos suporte ao novo hiperplano, no pior caso. Logo, s_j pode ser podado, pois se tem a garantia de que ele não será um ponto suporte nas próximas iterações, desde que a variação máxima do vetor x esteja dentro do cone 2α.

5.2 – Especificação inicial do ângulo de poda α

Considerando o objetivo de podar o maior número de observações possível, deve ser especificado o menor valor possível para o parâmetro α, associado ao ângulo da poda. Todavia, caso essa estratégia for por demasia gulosa, existe o risco real de ser eliminado incorretamente um número elevado de observações, candidatas reais a serem pontos suporte.

Foi observado nos experimentos computacionais, que valores de α iniciais muito pequenos podem inclusive levar à não convergência, devido à variância errática do conjunto de observações de trabalho. De outro lado, a especificação de um valor alto para o parâmetro α, embora ofereça uma garantia empírica de convergência, implica na inoperância dos procedimentos de poda. Dessa forma, a especificação do valor inicial do parâmetro α deve contemplar esses dois fenômenos polares.

A partir da segunda iteração, o valor do parâmetro α é especificado de acordo com a variação observada do ângulo do vetor x da equação do hiperplano separador, determinada pelos seus dois últimos valores.

Esta estratégia mostrou-se adequada por oferecer os requisitos necessários de robustez, uma vez que a variação desses ângulos apresenta, em geral, um comportamento decrescente no decorrer do processo de otimização. Dessa forma, a especificação desse critério acima definido enseja que nas iterações iniciais o ângulo α seja maior que nas iterações finais. Assim, tem-se a garantia de que, no início do processo, quando há uma maior variação no ângulo de *x*, sejam podados poucos pontos, enquanto que no final do procedimento, quando se está próximo da solução e a variação de α é menor, o número de pontos podados seja muito maior.

Esta relação do número de pontos podados com a variação de α pode ser facilmente demonstrada. Sem perda de generalidade, considere $\alpha \in \left[0, \pi/2\right]$. Pode ser observado que o número de elementos excluídos do processo de otimização está diretamente ligado ao valor de α, de forma que quanto maior o valor de α menor será o número de pontos "podados" e quanto menor o valor de α maior o número de elementos podados.

Como exemplo, no caso limite para $\alpha = \pi/2$ o valor de $d_{\max j}$ será positivo e $d_{\min j}$ será negativo, já que d_{j2} é positivo. Assim, $d_{\min j} < d_{\max i}, \forall i, j$ e nenhum dos pontos poderá ser podado.

Por outro lado, no caso extremo para $\alpha = 0$ tem-se $d_{\max j} < 0$ e $d_{\min j} > 0$, logo $d_{\min j}$ será sempre maior que $d_{\max i}$, $\forall i, j$, e todos os pontos não-suporte na iteração serão excluídos do conjunto de trabalho.

Em virtude dessa observação, pode ser concluído que a escolha do ângulo α deve ser feita cuidadosamente para que o processo convirja corretamente. Nos casos intermediários, onde $\alpha \in \left]0, \pi/2\right[$, as observações que estão mais distantes do hiperplano terão $d_{\min j} < d_{\max i}, \forall i, j$ e serão conseqüentemente excluídas do conjunto de trabalho. Já as observações que estiverem próximas ao hiperplano possuirão $d_{\min j} > d_{\max i}, \forall i, j$, sendo essas mantidas no conjunto de trabalho.

5.3 – Critérios Adicionais do processo de Poda

Alguns critérios para melhorar a eficiência do procedimento responsável por realizar a poda sobre o conjunto de treinamento foram adotados e serão descritos nesta

seção. Foi observado durante os testes do sistema que a variação entre o vetor x_k, de uma dada iteração k, e o vetor x_{k+1}, da iteração $k+1$, era, na maioria das vezes, menor que a variação observada entre as vetores x_k e x_{k-1}. De certa forma, o ângulo formado entre vetores consecutivos reduz a cada iteração, ou seja, embora a variação não seja estritamente monótona decrescente, ela é dominante decrescente. O gráfico da Figura 5.3 ilustra essa situação.

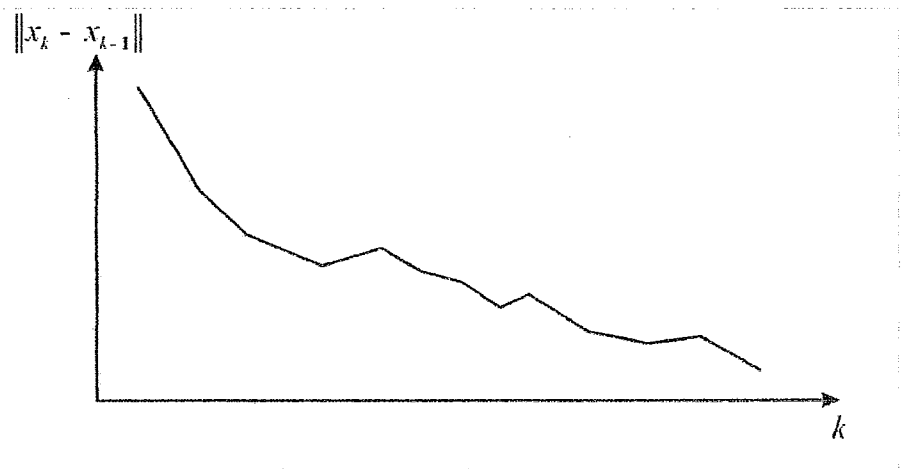


Figura 5.3 - Variação do ângulo α entre duas direções consecutivas.

Considere que o procedimento de poda sobre as observações do conjunto de treinamento tenha sido executado com um ângulo α definindo um conjunto de trabalho. Ao executar esse procedimento novamente, caso a variação entre o vetor x_{k-1} da poda anterior e o novo vetor x_k, seja menor que o ângulo α utilizado na última poda, tem-se a garantia de que todos os pontos retirados do conjunto de treinamento continuarão fora do conjunto. Ou seja, caso a poda seja feita sobre todo o conjunto, todos os pontos que haviam sido retirados do conjunto serão podados novamente.

A fim de melhorar a eficiência do método é definido um cone, Figura 5.4, de forma que enquanto os vetores x estiverem dentro do cone o procedimento de poda será executado somente sobre os pontos do conjunto de trabalho, já que os pontos anteriormente retirados do conjunto de treinamento permanecerão fora dele.

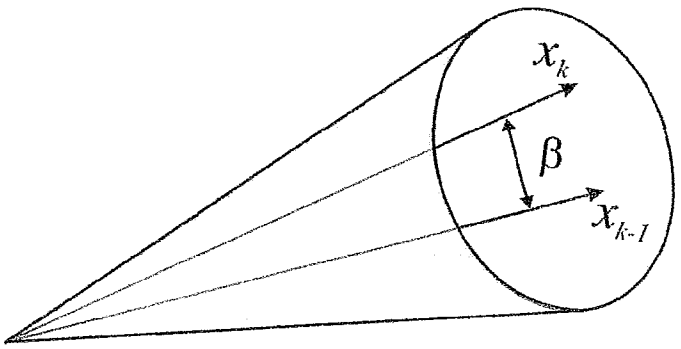


Figura 5.4 - Variação angular dentro do cone

Seja β_k o ângulo de variação entre as direções x_k e x_{k-1}, conforme ilustra a Figura 5.4. Para se ter um melhor procedimento de poda o ângulo de poda α deve ser idealmente o menor possível. A Figura 5.3 indica uma tendência de decréscimo dos valores de β. Entretanto, a escolha gulosa $\alpha_{k+1} = \beta_k$ pode se mostrar errônea. Um critério que concilia efeitos de redução e de adequação é tomar:

$$\alpha_{k+1} = \frac{(\beta_k + \beta_{k-1})}{2} \tag{5.7}$$

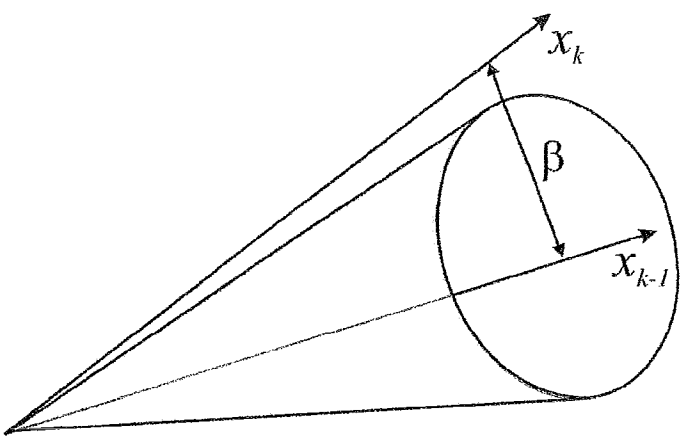


Figura 5.5 - Variação angular fora do cone.

Por outro lado, caso o novo vetor x saia do cone, Figura 5.5, não pode ser garantido que as observações retiradas do conjunto de treinamento não influenciarão na solução. Assim, deve ser efetuado um recomeço do processo em que o procedimento de poda deve ser realizado sobre todas as observações do conjunto de treinamento. Uma escolha adequada para o ângulo α utilizado para recomeçar o processo de poda pode

ser definido pelo ângulo formado entre o vetor x_{k-1} da poda anterior e o novo vetor x_k da poda atual, como na seguinte equação:

$$\alpha_{k+1} = \hat{a}ngulo(x_k, x_{k-1}), \tag{5.8}$$

onde $\hat{a}ngulo(a,b)$ é uma função que define o ângulo formato pelos vetores a e b. Sempre que há uma redefinição do ângulo de poda o cone também é redefinido.

Durante os testes realizados, observou-se que há, afortunada e tipicamente, uma seqüência de cones que são contidos nos cones anteriores. Como, normalmente, a variação de β é decrescente, o novo cone acaba por ser definido dentro dos cones definidos anteriormente, conforma ilustrado pela Figura 5.6 (a). Assim, a cada nova definição do cone é menor o conjunto de trabalho gerado.

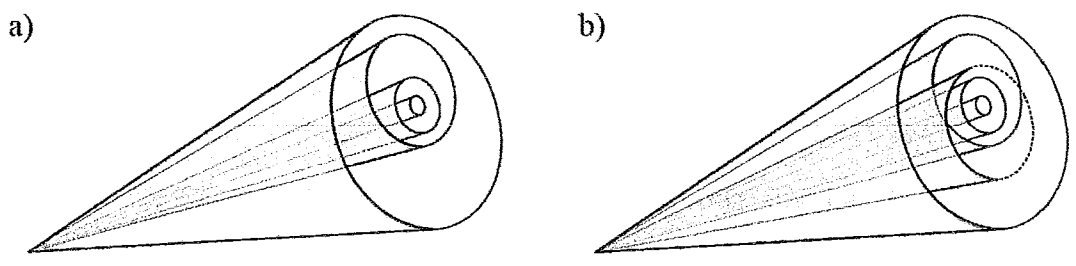


Figura 5.6 - Comportamento típico da seqüência de cones (a); Comportamento atípico com vetor x definido fora do cone (b).

Como dito anteriormente, caso o novo vetor tenha uma variação maior que o cone, um novo cone será definido. Nesse contexto, todos os pontos pertencentes ao que se pode chamar de "cone continente" devem ser considerados no processo de poda.

5.4 – Poda por similaridade

Após a aplicação do procedimento de poda descrito anteriormente, foi observado que o mesmo era muito eficiente para alguns conjuntos de treinamento, enquanto para outros conjuntos os resultados ficavam aquém do esperado. De forma que, em um determinado problema, eram eliminados mais de 95% das observações do conjunto de treinamento e para outros problemas somente cerca de 50% das observações eram eliminadas.

Para o primeiro caso, era gerado um conjunto de trabalho pequeno e com poucas observações a mais que o número de pontos suporte, produzindo, assim, um grande aumento da velocidade de convergência da metodologia de classificação. Já no segundo caso, havia uma melhora na eficiência da metodologia, porém essa não era muito significante quando comparada com os resultados obtidos pelo *SVMLight*.

Após uma análise minuciosa sobre os conjuntos de treinamento que estavam sendo utilizados no processo de otimização, concluiu-se que, embora todas as observações fossem completamente diferentes entre si, havia vários grupos com observações muito semelhantes. Assim, embora as observações fossem distintas, elas estavam muito próximas umas das outras em aglomerados. A Figura 5.7 esboça de maneira simplificada o efeito da aglomeração de várias observações.

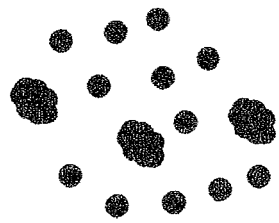


Figura 5.7 - Observações aglomeradas

Com a ocorrência desse fenômeno o processo de poda torna-se pouco eficiente, já que quando um ponto não era eliminado pelo processo, um grupo de pontos distintos, próximos a ele, também não era eliminado. Como exemplo, considere um conjunto de treinamento com 20000 observações no qual existem 5000 observações produzidas por pequenos ruídos em torno de um ponto suporte. Já que essas observações estão praticamente sobrepostas umas às outras, uma pequena variação no ângulo α irá excluir todas ou, num sentido diametralmente oposto, nenhuma dessas observações. Como somente os pontos suporte influenciam na solução dos problemas *SVM*, pode-se concluir que, nessa última situação, o conjunto de trabalho desse problema terá certamente mais de 5000 observações.

Na Figura 5.8 é ilustrado o caso onde várias observações estão aglomeradas sobre um ponto suporte. A região definida em tons mais claros é relativa aos pontos que foram podados durante o procedimento de poda para uma certa variação do vetor x. Observe que nenhuma observação que está ao redor do ponto suporte é eliminada do

conjunto de treinamento, produzindo um conjunto de trabalho com número de observações maior que o necessário.

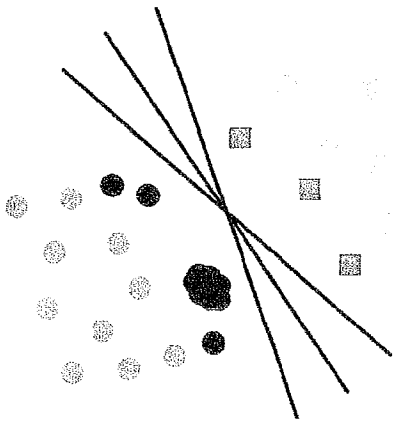


Figura 5.8 - Observações aglomeradas não podadas

Com o auxílio das figuras anteriores, pode ser observado que os pontos que estão muito próximos têm suas distâncias, em relação ao hiperplano, praticamente iguais. Assim, na primeira iteração, os pontos que possuem distâncias relativas ao hiperplano "iguais" são excluídos do conjunto de treinamento. Dentre todos os pontos que têm distâncias iguais, apenas um continuará no conjunto de trabalho.

Essas considerações induzem à idéia de se efetuar uma "poda de pontos aglomerados" que aqui é denominada *poda por similaridade*.

A poda por similaridade é realizada somente na primeira iteração, a cada iteração seguinte é feito um procedimento de repescagem para garantir que nenhuma observação excluída por esse critério estará violando a solução obtida.

O algoritmo do procedimento de poda por similaridade é definido a seguir, de forma que D é um conjunto de intervalos de distâncias das observações ao hiperplano, J é o conjunto de trabalho, $\overline{J}$ é o conjunto de observações excluídas do conjunto de trabalho por esse processo de poda por similaridade e d_j é a distância da observação j ao hiperplano. É importante lembrar que esse algoritmo é executado somente na primeira iteração, portanto não há um grande acréscimo no custo computacional do processo como um todo.

Algoritmo de Poda por Similaridade

1: Faça $j = 0$, $D = \emptyset$, $\overline{J} = \emptyset$ e defina x^0.

2: Para cada observação j faça:

2.1: Se $d_j \notin D$ faça

- Adicione o intervalo $\left[d_j - \delta,\ d_j + \delta\right]$ ao conjunto D, onde δ é um parâmetro de tolerância;

- Adicione j ao conjunto de trabalho.

2.2: Senão faça

- Adicione j ao conjunto $\overline{J}$;

- Exclua j do conjunto de trabalho.

3: Fim-Procedimento.

Embora a idéia seja muito simples, existem alguns casos nos quais esse procedimento de redução do conjunto de treinamento pela distância entre os pontos e o hiperplano pode proporcionar uma pequena oscilação na solução final. Esse comportamento dar-se-á quando as observações estiverem, por exemplo, aglomeradas ao redor do ponto suporte, já que o verdadeiro ponto suporte pode ter sido removido, como ilustrado na Figura 5.9.

Esses casos de variação da solução final ocorrem em função do critério de poda por similaridade ser um critério heurístico.

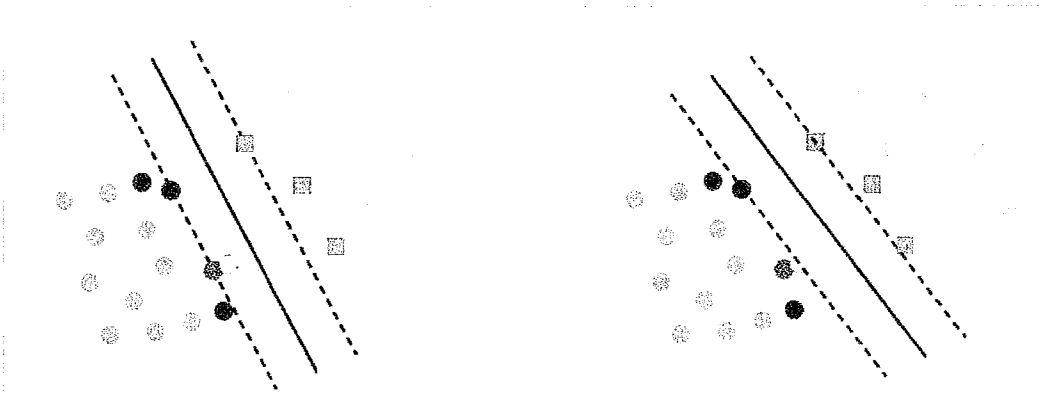


Figura 5.9 - Solução do problema sem as observações aglomeradas (esquerda) e solução do problema com a observação que viola a margem (direita).

Visando solucionar essa dificuldade, durante o processo de otimização todos os pontos $s_j, j \in \overline{J}$ são analisados e os que estiverem violando a margem voltam para o conjunto de trabalho. A cada nova iteração, quando um novo hiperplano é definido, todas as observações $s_j, j \in \overline{J}$ são reavaliadas, de forma que as que violarem a margem voltarão para o conjunto de trabalho.

Note que as observações são excluídas por similaridade somente na primeira iteração, em todas as outras é feito um trabalho de repescagem para que nenhuma observação viole a margem definida pelo classificador. Assim, no final do processo de otimização tem-se a garantia de que nenhuma observação viola a margem.

Na Figura 5.9 são exibidas as duas situações descritas anteriormente. Na figura da esquerda, a solução foi obtida selecionando-se aleatoriamente um dos pontos que estavam aglomerados sobre o ponto suporte. Pode ser observado que, para a solução obtida, um dos pontos que foram descartados violam a margem, justificando a necessidade do processo de repescagem sobre os pontos podados pelo critério de similaridade das distâncias.

Unindo a poda por variação angular com a poda por similaridade, a redução do conjunto de treinamento tornou-se bastante significativa, aumentando, consideravelmente, a eficiência do processo de otimização.

Já a situação mostrada à direita da Figura 5.9 retrata a correção da solução adicionando o ponto que violava a margem no conjunto de trabalho. Como a observação é adicionada ao conjunto de trabalho durante o processo de otimização, a solução é corrigida pelo próprio processo iterativo, de forma que a solução obtida no final é a solução correta.

Deve ser destacado que a poda por variação angular e a poda por similaridade são procedimentos diferentes e que recebem tratamentos específicos.

Somente a poda por similaridade pode produzir variação da margem, já que não há restrição de poda sobre pontos suporte e pontos não suporte. Assim, um ponto suporte pode ser retirado do conjunto de trabalho, já que qualquer ponto pode ser excluído. Por isso, enfatiza-se ser necessário fazer uma repescagem sobre os pontos que foram excluídos pelo critério de similaridade, a fim de que nenhum deles viole a margem definida pelo hiperplano.

Por outro lado, o critério de poda por variação angular é conservador ao remover observações do conjunto de treinamento. Somente os pontos que não são suporte serão retirados do conjunto de treinamento, tendo-se a garantia de que nenhuma observação irá violar a margem no final do processo de otimização. Logo, não é necessário fazer nenhuma repescagem sobre os pontos removidos por esse critério de poda, desde que a variação da solução esteja dentro do cone pré-definido.

5.4.1 – Análise da Poda por Similaridade

A fim de realizar uma análise mais criteriosa sobre o procedimento de poda por similaridade, considere o problema teste ilustrado na Figura 5.10. No primeiro quadro, tem-se a representação da primeira iteração e a definição do ponto inicial dado por x^0. Observa-se que embora as observações s_1 e s_2 da classe 1 e as observações s_3 e s_4 da classe 2 não estejam aglomeradas, elas estão à mesma distância do hiperplano definido pelo vetor x^0.

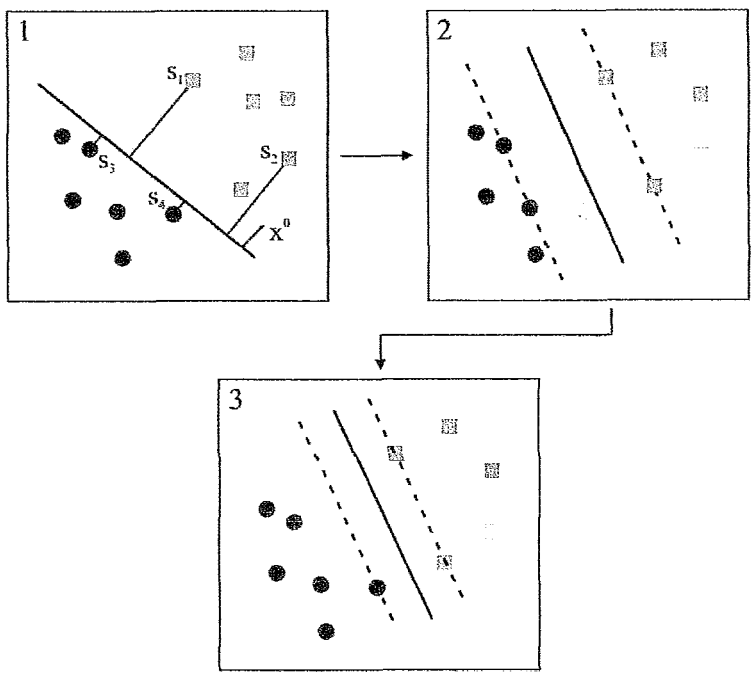


Figura 5.10 - Análise do Comportamento da Poda por Similaridade

De acordo com o funcionamento do procedimento de poda por similaridade, as observações s_2 e s_4 foram excluídas do conjunto de trabalho, já na primeira iteração,

conforme primeiro quadro da Figura 5.10. No segundo quadro, é obtida a solução para o problema desconsiderando as observações excluídas na primeira iteração.

Após a segunda iteração o procedimento de repescagem é executado. Observa-se que uma das observações que foram eliminadas do conjunto está violando a margem definida na segunda iteração. Assim, o processo de repescagem localiza essa observação e coloca-a novamente no conjunto de trabalho.

Na terceira iteração, um novo hiperplano separador é calculado utilizando a observação que estava violando a margem na iteração anterior. Após obter a solução para o novo conjunto o procedimento de repescagem é executado novamente. Como nenhuma observação viola a solução obtida, tem-se a garantia de que a solução está correta.

Para todos os conjuntos de treinamento utilizados como teste, os resultados obtidos aplicando-se o procedimento de poda por similaridade foram bem sucedidos. Entretanto, carece-se fazer mais experimentos computacionais para uma avaliação mais precisa da efetividade do processo de poda por similaridade.

Ademais, como esses resultados demonstraram sucesso, é de se esperar que haja a possibilidade de implementação de outros critérios de poda igualmente eficientes, como é feito no *SVMLight*.

5.5 – Resultados

Aplicando os critérios de poda acima descritos, foi efetuado um novo conjunto de experimentos computacionais. Novamente foram utilizadas bases de dados tradicionais, descritas no capítulo anterior, bem como bases de dados aleatórias, a fim de se estabelecer uma comparação entre os resultados anteriores e os resultados atuais.

Em função das dificuldades de comparação para conjuntos de treinamento não linearmente separáveis, os resultados obtidos a partir da redução do conjunto de treinamento foram analisados apenas sobre as bases de dados linearmente separáveis. Além disso, ao aplicar os critérios de poda sobre o conjunto de treinamento do problema de câncer de mama, os resultados obtidos foram os mesmos que os apresentados no capítulo anterior, com exceção do tempo de processamento que foi ligeiramente maior que o gasto anteriormente, como pode ser visto na Tabela 5.1.

Tabela 5.1 - Comparação de resultados entre os classificadores SHSVM, SVMLight, SHSVMPoda para o problema de câncer de mama

Base de Dados m x n	Algoritmo	Exatidão fase de treinamento (%)	Tempo CPU (segundos)
wpbc 198 x 32	SHSVM	76.26	1.17
	SVMLight	76.26	11.49
	SHSVMPoda	76.26	1.32

Primeiramente, o classificador SHSVMPoda foi aplicado aos problemas tradicionais, como o problema da Tiróide, reconhecimento de Letras, problema da Íris e para o problema de reconhecimento de câncer Leukêmia, gerando os resultados apresentados na Tabela 5.2. Os resultados dessa tabela mostram que para bases de dados pequenas, ou seja, com poucas observações, o uso do classificador SHSVMPoda não é vantajoso. Isso se deve ao custo computacional exigido pelo procedimento de poda, ou seja, o número de observação removidas do conjunto de treinamento será certamente pequeno e, assim sendo, o tempo gasto para a construção do conjunto de trabalho não é compensado pelo tempo gasto durante o processo de otimização.

Tabela 5.2 - Comparação de resultados entre os classificadores SHSVM, SVMLight, SHSVMPoda

Base de Dados m x n	Algoritmo	Margem	Exatidão fase de teste (%)	Tempo CPU (segundos)
Ann-Thyroid 284 x 21	SHSVM	0. 00243384581	97.89	0.55
	SVMLight	0.002435179390	97.83	0.20
	SHSVMPoda	0.002435030973	97.89	1.14
Letras 1555 x 16	SHSVM	0.116839728245	99.04	2.2
	SVMLight	0.116888042879	98.74	0.11
	SHSVMPoda	0.116842005797	99.04	0.8
Iris 150 x 4	SHSVM	1.635107122655	100	0.09
	SVMLight	1.635109061774	100	0.02
	SHSVMPoda	1.635087515546	100	0.12
Leukemia 38 x 50	SHSVM	5568.139424290	94.74	0.08
	SVMLight	5555.555555556	93.10	0.01
	SHSVMPoda	5568.084400123	94.74	0.16

Os problemas da Tiróide, da Íris e de câncer Leukemia, que possuem conjuntos de treinamento pequenos, demonstram exatamente o que foi descrito anteriormente. Comparando o SHSVM com o SHSVMPoda, para esses casos, percebe-se que o tempo de processamento gasto pelo SHSVMPoda chega a ser duas vezes maior que o tempo gasto pelo SHSVM. Já no caso do problema de reconhecimento de letras, que tem um conjunto de treinamento maior, o tempo de processamento reduziu substancialmente, aproximando-se o do tempo de CPU gasto pelo *SVMLight*.

A fim de testar o funcionamento do SHSVMPoda para bases de dados maiores, foram utilizadas as mesmas bases de dados aleatórias, descritas no capítulo anterior, geradas usando o software *NDC Data Generator*. Dois tipos de conjuntos de treinamento foram desenvolvidos. O primeiro, tem a característica de possuir dados aglomerados, ou seja, existem várias observações muito próximas umas das outras, estando praticamente sobrepostas. O segundo tipo de conjunto de treinamento foi gerado de forma que os dados estivessem espalhados, de forma que nenhuma observação esteja sobreposta à outra.

Tabela 5.3 - Comparação de resultados para bases aleatórias com dados aglomerados

Base de Dados m x n	**Algoritmo**	**Margem**	**Exatidão fase de teste (%)**	**Tempo CPU (segundos)**
Base 200 200 x 10	SHSVM	15.65636759185	100	0.03
	SVMLight	15.65680288085	100	0.48
	SHSVMPoda	15.65637038808	100	0.04
Base 2000 2000 x 10	SHSVM	3.106290251622	100	1.30
	SVMLight	3.106361829025	100	0.03
	SHSVMPoda	3.106394517789	100	0.31
Base 20000 20000 x 10	SHSVM	3.101742156230	100	55.90
	SVMLight	3.102843756302	100	0.50
	SHSVMPoda	3.101140464835	100	5.33
Base 200000 200000 x 10	SHSVM	0.025655465151	100	193.77
	SVMLight	0.025922450397	100	9.07
	SHSVMPoda	0.025913804190	100	38.78

Para o primeiro tipo de conjunto de treinamento gerado, base com 200 observações e 10 características, como era esperado, o resultado obtido para a primeira base não apresentou melhora em relação ao SHSVM, já que se trata de uma base com poucas observações, como mostra a Tabela 5.3. Entretanto, o resultado para esse problema continuou sendo melhor, em tempo, que o resultado do *SVMLight*.

Por outro lado, para todas as outras bases que possuem um número maior de observações, embora o resultado não seja melhor que o apresentado pelo SVMLight, o tempo de processamento do SHSVMPoda foi muito menor do que o do SHSVM, reduzindo em mais de $\frac{1}{4}$ o tempo de CPU gasto em todos os três casos.

Tabela 5.4 - Comparação de resultados para bases aleatórias com dados esparsos

Base de Dados m x n	**Algoritmo**	**Margem**	**Exatidão fase de teste (%)**	**Tempo CPU (segundos)**
Base 200 200 x 10	SHSVM	1.098027249260	100	0.09
	SVMLight	1.098032326071	100	0.60
	SHSVMPoda	1.098029569068	100	0.06
Base 2000 2000 x 10	SHSVM	0.054207810229	100	4.17
	SVMLight	0.054346748012	100	2.56
	SHSVMPoda	0.054295314836	100	0.92
Base 20000 20000 x 10	SHSVM	0.983238264857	100	16.45
	SVMLight	0.983874300219	100	8.95
	SHSVMPoda	0.983226212314	100	2.40
Base 200000 200000 x 10	SHSVM	0.810413968764	100	182.20
	SVMLight	0.811003698176	100	27.47
	SHSVMPoda	0.810404575781	100	22.34

Embora os resultados tenham tido ganhos significativos, o critério de construção do conjunto de trabalho do *SVMLight* é mais eficiente para esse tipo de distribuição do conjunto de dados. Isso se deve ao fato do *SVMLight* não utilizar todo o conjunto de treinamento para gerar o conjunto de trabalho. Ele seleciona um pequeno grupo de observações, obtém uma solução para as observações selecionadas e as que violarem a solução são colocadas no conjunto de trabalho, retirando outras que não violam.

Dessa forma, quando o conjunto de treinamento possui dados muito próximos uns dos outros, o *SVMLight* tem uma vantagem em relação as outras técnicas de redução do conjunto de treinamento. Isso se deve ao fato de que ao selecionar uma observação, todas as outras que estão próximas a ela não serão mais avaliadas, já que a observação selecionada representará todas as outras e essas outras não violarão o classificador. Já no SHSVMPoda, todo o conjunto de treinamento é analisado de forma a verificar quais são as restrições que podem ser excluídas. Logo, enquanto o *SVMLight* iniciará o processo de otimização com um conjunto pequeno, o SHSVMPoda começará com um conjunto de trabalho grande para depois reduzi-lo.

Os resultados obtidos na resolução de bases de dados esparsas, apresentados na Tabela 5.4, demonstram o sucesso dos procedimentos de poda implementados no algoritmo SHSVMPoda. Não só os tempos foram melhores do que aqueles obtidos na versão original SHSVM, bem como foram também bem melhores do que os do *SVMLight*.

Os resultados obtidos para os conjuntos de treinamento sem dados aglomerados, Tabela 5.4, comprovam a conjectura descrita anteriormente. Em todos os casos o SHSVMPoda teve um tempo de processamento menor que o *SVMLight*. Para esse tipo de distribuição do conjunto de treinamento, onde as observações estão completamente espalhadas o *SVMLight*, ao selecionar uma observação para ser incluída no conjunto de trabalho, todas as outras continuarão tendo que ser analisadas, já que nenhuma observação será igual à selecionada, ou seja, cada observação é única. Assim, o *SVMLight*, em princípio, requer um número de iterações maior para que ele tenha um conjunto de trabalho ótimo e obtenha a solução ótima. Já o critério de poda por variação angular do SHSVM, descrito neste capítulo, irá eliminar mais facilmente os pontos que estiverem mais afastados do hiperplano e por isso apresenta um desempenho melhor que o *SVMLight* para conjuntos de dados esparsos.

5.6 – Síntese dos Resultados dos Procedimentos de Poda

Os resultados deste capítulo mostram a eficiência do critério de poda, principalmente, em bases de dados de grande porte. O ganho da nova versão SHSVMPoda em relação ao SHSVM original foi bastante significativo em todos os casos, reduzindo consideravelmente o tempo de processamento do classificador, exceto para conjuntos de treinamento pequenos.

O SHSVMPoda não tem problemas com bases de dados esparsos. Pelo contrário, conjuntos de dados esparsos proporcionam um melhor funcionamento do critério de poda por variação angular, tornando o classificador mais eficiente. Os dados que estiverem muito afastados do hiperplano em geral são eliminados na primeira iteração e não voltam mais para o conjunto de trabalho.

O SVMLight apresentou melhor desempenho em problemas que possuem conjuntos de treinamento com dados aglomerados. De outro lado, não produziu bons resultados, em termos de tempo de processamento, para bases de dados de grande porte com dados esparsos.

Geralmente, os problemas reais não apresentam muitas observações aglomeradas como nos problemas com dados aglomerados gerados neste capítulo. Muito pelo contrário, os conjuntos de treinamento de problemas reais apresentam observações mais esparsas, como é o caso das bases de dados obtidas na literatura (Tiróide, Íris, Leukemia e reconhecimento de letras). Assim, a princípio, a poda por similaridade terá pouca influência na resolução de problemas reais, enquanto que a poda por variação angular terá maior influência nesses casos.

Capítulo 6 – Conclusão

Neste trabalho, foi abordado o tópico: problemas de classificação. O enfoque foi dado à classificação via SVM. Foi apresentada e discutida uma nova metodologia para resolução do problema SVM, constituída pela articulação das técnicas de Suavização Hiperbólica e Penalização Hiperbólica. Os resultados produzidos apresentam boa precisão numérica e o tempo computacional exigido pelo método foi, primeiramente, aceitável.

Pode-se acreditar que a proposta deste trabalho obteve êxito, já que a articulação das técnicas de suavização hiperbólica com a penalização hiperbólica e a harmonização do conjunto de parâmetros, proporcionou resultados com as propriedades idealizadas *a priori*.

A harmonização do conjunto de parâmetros demonstrou-se adequada, entretanto nem todas as alternativas foram exauridas. Assim, provavelmente, pode ser obtida uma melhor combinação de valores para esses parâmetros, de forma que os resultados, no quesito tempo, melhorem ainda mais.

O desenvolvimento da uma nova metodologia para redução do conjunto de treinamento também apresentou resultados satisfatórios, pois possibilitou uma grande melhora na eficiência do algoritmo, tornando-o, em alguns casos, mais eficiente que o *SVMLight*. Assim, o software SHSVMPoda alcançou os requisitos de robustez e de eficiência.

O problema específico contemplado foi o de classificação, por máquina de vetor suporte, utilizando sempre um hiperplano como separador. Vale a pena ressaltar que essa metodologia pode ser expandida a fim de utilizar funções não lineares como separadores, através do uso de *kernel*.

Uma outra abordagem interessante, presente nas formulações de Cortes e Vapnik, problema (2.16), e SVMLight, definido em (4.1), é permitir que alguns dados violem a margem, penalizando essas violações. Essa abordagem permite a resolução de uma alternativa do SVM denominada SVM de *soft* margem.

Em suma, por todos os resultados obtidos, pode-se afirmar que a abordagem metodológica foi plenamente exitosa, pois permitiu a obtenção de boas soluções, dentro do nível de precisão dado pela máquina e pelos valores finais dos parâmetros τ, ε e

ρ. Além disso, os resultados foram obtidos com um baixo custo computacional, dada a eficiência da técnica de poda sobre o conjunto de treinamento.

6.1 – Propostas para trabalhos futuros

Como uma extensão natural do presente trabalho, pode-se cogitar o emprego de uma abordagem mais ampla do problema SVM utilizando-o na resolução de problemas mais complexos, como os problemas não linearmente separáveis.

Uma boa alternativa é utilizar as funções *kernel* na formulação matemática do problema SVM, a fim de obter separadores não lineares na resolução de problemas de classificação. O uso de *kernel* possibilita que os dados do conjunto de treinamento sejam separados por uma superfície, não linear, que terá um maior poder de generalização que uma superfície linear.

Uma outra alternativa é aprimorar os critérios de poda para tentar reduzir ainda mais o tempo de processamento, utilizando-se estruturas de dados específicas, mais rápidas e que ocupem menos espaço em memória.

Apêndice A

A.1 - Teorema da Função Implícita

Considere o conjunto de m equações e n variáveis

$$h_i(x)=0, \qquad i=1,2,...,m. \tag{7.1}$$

O teorema da função implícita afirma que caso $n-m$ das variáveis são fixas, as equações podem ser resolvidas pelas m variáveis restantes (LUENBERGE, 1937). Assim, selecionando m variáveis, $x_1, x_2, ..., x_m$, deseja-se determinar se estas m variáveis podem ser obtidas através das variáveis restantes na forma

$$x_i=\phi_i(x_{m+1}, x_{m+2}, ..., x_n), \qquad i=1,2,...,m. \tag{7.2}$$

As funções ϕ_i, caso existam, são chamadas funções *implícitas*.

Teorema. Seja $x^0=(x_1^0, x_2^0, ..., x_n^0)$ um ponto no $\Re^n$ que satisfaça as seguintes propriedades:

i) As funções $h_i \in C^p, i=1,2,...,m$ em alguma vizinhança de x^0, para algum $p \geq 1$.

ii) $h_i(x^0)=0, \qquad i=1,2,...,m.$

iii) A Matriz Jacobiana m x m

$$\mathbf{J}=\begin{pmatrix} \dfrac{\partial h_1(x^0)}{\partial x_1} & \cdots & \dfrac{\partial h_1(x^0)}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial h_m(x^0)}{\partial x_1} & \cdots & \dfrac{\partial h_m(x^0)}{\partial x_m} \end{pmatrix}$$

é não singular.

Então existe uma vizinhança de $\hat{x}^0=(x_{m+1}^0, x_{m+2}^0, ..., x_n^0) \in \Re^{n-m}$, tal que para $\hat{x}=(x_{m+1}, x_{m+2}, ..., x_n)$ nesta vizinhança, existem funções $\phi_i(\hat{x}), i=1,2,...,m$ tais que

i) $\phi_i \in C^p$.

ii) $x_i^0 = \phi_i(\hat{x}^0),\ i = 1, 2, ..., m.$

iii) $h_i\left(\phi_1(\hat{x}), \phi_2(\hat{x}), ..., \phi_m(\hat{x}), \hat{x}\right) = 0,\ i = 1, 2, ..., m.$ A Matriz Jacobiana $m\ x\ m$

Exemplo 1. Seja a equação $x_1^2 + x_2 = 0$. A solução é $x_1 = 0$, $x_2 = 0$. Entretanto, na vizinhança desta solução existe uma função ϕ tal que $x_1 = \phi(x_2)$. Nesta solução a condição (iii) do teorema da função implícita é violada. Em qualquer outra solução, entretanto, ϕ existe.

Exemplo 2. Seja **A** uma matriz $m\ x\ n$ com $m < n$ e seja o sistema de equações lineares $\mathbf{Ax = b}$. Se **A** é particianada como $\mathbf{A = [B, C]}$ onde **B** é $m\ x\ m$, então a condição (iii) é satisfeita, se e somente se **B** é não singular. Esta condição corresponde exatamente com o que a teoria de programação linear consagra. Em vista deste exemplo, o teorema da função implícita pode ser aplicado obtendo uma generalização não linear da teoria desenvolvida no contexto linear.

Referências

BENNETT, K.P., MANGASARIAN, O.L., 1992, "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", *Optimization Methods and Softwares 1*, pp. 23-34.

BRADLEY, P.S., FAYYAD, U.M., MANGASARIAN, O.L., 1998, "Mathematical Programming for Data Mining: Formulations and Challenges", *Journal on Computing*, v. 11 (1999), pp. 217-238.

BRITO, J.A.M., 2004, *Suavização Hiperbólica Aplicada no Problema de Localização de Estações de Rádio Base*, Tese D. Sc., COPPE, UFRJ, Rio de Janeiro, RJ, Brasil.

BURGES, C.J.C., 1998, "A Tutorial on Support Vector Machine for Pattern Recognition", *Data Mining and Knowledge Discovery*, v. 2, pp. 121-167.

CHAVES, A.M.V., 1987, *Resolução de Problemas Minmax Via Suavizações*, Dissertação de M. Sc., COPPE, UFRJ, Rio de Janeiro, RJ, Brasil.

CHAVES, A.M.V., XAVIER, A.E., 1998, *Problemas Minimax: Uma Alternativa de Resolução via suavização*, Relatório Técnico, COPPE, UFRJ, Rio de Janeiro, RJ, Brasil.

CORTES, C., VAPNIK, V., 1995, "Support Vector Networks", *Machine Learning*, v. 20, pp. 273-297.

DIB, K.R., 1994, *Utilização de Função de Penalização Hiperbólica na Suavização e Otimização de um Modelo Chuva-Vazão: Modelo SWMS*, Tese de D.Sc., COPPE, UFRJ, Rio de Janeiro, RJ, Brasil.

GUNN, S.R., 1998, *Support Vector Machines for Classification and Regression*, Faculty of Engineering, Science and Mathematics School of Eletronics and Computer Science, University of Southhampton.

ISAAC, F., 2003, "A Discussion of Data Analysis Prediction and Decision Techniques", *Fair Isaac*.

JOACHIMS, T., 1998, *Making large-Scale SVM Learning Practical.*, Computer Science Department of the University of Dortmund, 24.

LUENBERGE, D.G., 1937, *Linear and nonlinear programming*, Second ed., Stanford University.

MANGASARIAN, O.L., MUSICANT, D.R., 1999, *Data Discrimination Via Nonlinear Generalized Support Vector Machines*, Computer Sciences Department, University of Wisconsin, 96-05.

MERCER, J., 1909, "Functions of positive and negative type and their connection with the theory of integral equations." *Philos. Trans. Roy. Soc. London*, n. A 209, pp. 415-446.

MINOUX, M., 1986, "Mathematical Programming: Theory and Algorithms", *John Wiley and Sons, Chichester.*

MOTA, F.C., BHAYA, A., KASKUREWICZ, E., 1992, *"Robust Stabilization of Time-Varying Discrete Internal Systems", Proceedings of Congress of Decision and Control (CDC-92)*, Tucson, Arizona.

MURPHY, P.M., AHA, D.W., 1992, "UCI Repository of Machine Learning Datasets", *www.ics.uci.edu/mlearn/MLRepository.html.*

MUSICANT, D.R., 1998, "NDC: Normally Distributed Datasets", *www.cs.wisc.edu/~musicant/data/ndc/.*

O.L.MANGASARIAN, MUSICANT, D.R., 2001, "Langrangian Support Vector Machines", *Journal of Machine Learning Research*, pp. 161-177.

PLATT, J.C., 1998, *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.*, Technical Report MSR-TR-98-14.

REYZIN, L., 2005, *Analyzing Margins in Boosting*, Princeton University Class of 2005, BSE Senior Independent Work.

SANTOS, A.B.A., 1997, *Problemas de Programação Não-Diferenciável: Uma Metodologia de Suavização*, Dissertação de M. Sc., COPPE, UFRJ, Rio de Janeiro, RJ, Brasil.

SILVA, L.P., XAVIER, A.E., CANEDO, M.P., 1990, *Calibração Automática de Modelos Chuva-Vazão: Um método Assintótico*, Tese M. Sc., COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil.

VAPNIK, V.N., 1995, *The nature of statistical learning theory*, Springer-Verlag New York, Inc.

XAVIER, A.E., 1982, *Penalização Hiperbólica: Um Novo Método para Resolução de Problemas de Otimização*, Tese M.Sc., COPPE, UFRJ, Rio de Janeiro.

XAVIER, A.E., 1993, *Solução de Problemas de Programação Não-Diferenciáveis via Suavização*, Relatório Técnico ES-290/93, PESC/COPPE, UFRJ, Rio de Janeiro.

XAVIER, A.E., 2000, "Optimum Covering of Plane Domains by Circles". In: *17th International Symposium on Mathematical Programming*, Atlanta, USA, 7-11 August.

XAVIER, A.E., 2005, *The Hyperbolic Smoothing Clustering Method*, Relatório Técnico 674/05, PESC/COPPE, UFRJ.

XAVIER, A.E., CURVELO, P., STRÖELE, V., *et al.*, 2006, "The Hyperbolic Smoothing Approach for Solving the Support Vector Machine Problem". In: *19 th International Symposium on Mathematical Programming*, Rio de Janeiro.

XAVIER, A.E., OLIVEIRA, A.A.F., 2003, "Optimum Order p Covering of Plane Domains by Circles Via Hyperbolic Smoothing Method". In: *International Symposium on Mathematical Programming*, Kopenhagen, August.

XAVIER, A.E., OLIVEIRA, A.A.F., 2005, "Optimum Covering of Plane Domais by Circles via Hyperbolic Smoothing Method", *Journal of Global Optimization*, v. 31, n. 3 (March), pp. 493-504.