Nota final
9,5 (nove e cinco)

UNIVERSIDADE DE SÃO PAULO
Escola Politécnica

DEPARTAMENTO DE ENGENHARIA MECATRÔNICA

22/01/04

PROJETO DE CONCLUSÃO DE CURSO

Classificação Automática de Dados com *Support Vector Machines (SVMs)* a Partir de Bases de Dados Incompletas

Clayton Silva Oliveira
Eduardo Takashi Inowe

Orientador: Prof. Dr. Fábio G. Cozman

São Paulo
2003

UNIVERSIDADE DE SÃO PAULO
Escola Politécnica

DEPARTAMENTO DE ENGENHARIA MECATRÔNICA

PROJETO DE CONCLUSÃO DE CURSO

Classificação Automática de Dados com *Support Vector Machines (SVMs)* a Partir de Bases de Dados Incompletas

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico

Clayton Silva Oliveira
Eduardo Takashi Inowe

Orientador: Prof. Dr. Fábio G. Cozman

São Paulo
2003

Resumo

O presente Projeto de Conclusão de Curso apresenta uma fusão de dois temas atualmente muito importantes no estudo de classificação automática de dados (parte da área de estudos de Inteligência Artificial): *Support Vector Machines* e *treinamento semi-supervisionado*. O classificador apresentado no primeiro tema vem substituindo as rede neurais em várias aplicações de automação de tomada de decisão; o segundo tema, vem sendo estudado por vários pesquisadores, numa tentativa de se utilizar bases incompletas para o treinamento de classificadores de dados. Através do programa MatLab, implementou-se uma SVM (a qual recebeu a denominação de *SVMsst*) de treinamento semi-supervisionado por *transdução*. Para medir a performance do classificador *SVMsst* após a execução do treinamento, utilizou-se um método de testes que fornecia uma medida estatística do desempenho da *SVMsst*. Em uma comparação da diferença de desempenho de *SVMsst*'s treinadas de forma supervisionada e semi-supervisionada, não se obteve melhora significativa quando do uso de transdução. Porém, quando se utilizou da transdução para o treinamento semi-supervisionado da *SVMsst*, seguido de re-treinamento do classificador, diferenças positivas de desempenho foram alcançadas, demonstrando a validade da implementação da SVM de treinamento semi-supervisionado. Como principal conclusão, a metodologia de treinamento semi-supervisionado, a partir de transdução, aplicada em SVMs pode permitir ao possível usuário deste tipo de classificador automático de dados a utilização de base de dados *incompletas* (ou *não completamente classificadas*) para o treinamento destes, fazendo com que este usuário aproveite o máximo de informação disponível, o que não seria possível com um treinamento inteiramente supervisionado.

Às nossas famílias, pelo carinho e apoio.

Aos nossos amigos, inesquecíveis companheiros.

À Deus, pela vida.

Agradecimentos

Em especial, ao professor e orientador Dr. Fábio Gagliardi Cozman pela orientação na condução de nosso trabalho, e pela confiança em nossas idéias. No geral, agradecemos a todo corpo docente responsável por nossa formação como Engenheiro Mecatrônico.

Agradeço aos meus pais, Kiyoshi e Dirce Inowe, pelo apoio e suporte durante o meu desenvolvimento tanto na Poli como na vida e pela minha formação pessoal. Obrigado pai, obrigado mãe! À minha irmã, Érica, pela confiança. Aos meus queridos amigos que fiz durante esses cinco anos, obrigado pelo apoio, carinho, e amizade, em especial aos *βetafriends* e aos amigos mecatrônicos, todos eternos amigos. E a Deus que me proporcionou a maior dádiva de todas.

Eduardo Takashi Inowe

Eu, Clayton, agradeço aos meus pais, Marinalva e José, pela minha formação como pessoa, baseada no amor, respeito e confiança. À minha irmã, Vanessa, pelo sempre constante apoio e solidariedade durante estes vinte e três anos. À minha noiva, Érica, pelo sublime amor e pelo nobre companheirismo para com meus sonhos. Aos meus amigos da Poli, que me ensinaram a ser uma pessoa melhor.

Clayton Silva Oliveira

SUMÁRIO

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$\mathbf{X}$	Conjunto de vetores-entrada
$\mathbf{Y}$	Conjunto de classes atribuídas a cada vetor do conjunto X
$\mathbf{w}$	Matriz que contém o conexionismo entre X e Y, ou seja, que implementa $f(X) = Y$.
α	Coeficiente de expansão linear do vetor $\mathbf{w}$
K	Kernel – relação entre os vetores de $\mathbf{X}$ utilizados durante o treinamento do classificador de dados
m	Número de elementos da base de treinamento
n	Número de parâmetros de cada elemento da base de treinamento
Z	Espaço dos dados de treinamento ($X \times Y$)
ϕ	*Feature* – mapeamento de X em um outro espaço vetorial
$\mathbf{G}$	Matriz Gram
λ	Medida de flexibilidade da SVM
$\mathbf{w}_{svm}$	Vetor de pesos da SVM
f_w	$f(X) = Y$ implementado por $\mathbf{w}_{svm}$
W	Wolfe Dual – funcional de otimização da SVM

1 Introdução

Essa seção introdutória procura contextualizar o atual significado de inteligência artificial e suas correntes de pensamento ao longo do tempo, bem como introduzir sucintamente as *Support Vector Machines (SVMs)* no cenário avançado da área de inteligência artificial. Por fim, são descritas algumas aplicações recentes das *SVMs* em áreas de reconhecimento de padrões.

1.1 O que é Inteligência Artificial (IA)? [1]

As correntes de pensamento que se cristalizaram em torno da IA já estavam em gestação desde os anos 30 [2]. No entanto, oficialmente, a IA nasceu em 1956 com uma conferência de verão em Dartmouth College, NH, USA. Na proposta dessa conferência, escrita por John McCarthy (Dartmouth), Marvin Minsky (Hardward), Nathaniel Rochester (IBM) e Claude Shannon (Bell Laboratories) e submetida à fundação Rockfeller, consta a intenção dos autores de realizar "um estudo durante dois meses, por dez homens, sobre o tópico inteligência artificial". Ao que tudo indica, esta parece ser a primeira menção oficial à expressão "*Inteligência Artificial*" [3]. Desde seus primórdios, a IA gerou polêmica, a começar pelo seu próprio nome, considerada presunçoso por alguns, até a definição de seus objetivos e metodologias. O desconhecimento dos princípios que fundamentam a inteligência, por um lado, e dos limites práticos da capacidade de processamento dos computadores, por outro, levou periodicamente a promessas exageradas e às correspondentes decepções.

Dada a impossibilidade de uma definição formal precisa para IA, visto que para tanto seria necessário definir, primeiramente, a própria inteligência, foram propostas algumas definições operacionais: "uma máquina é inteligente se ela é capaz de solucionar uma classe de problemas que requerem inteligência para serem solucionados por seres humanos" [4]; "Inteligência Artificial é a parte da ciência da computação que compreende o projeto de sistemas computacionais que exibam características associadas, quando presentes no comportamento humano, à inteligência" [2]; ou ainda, "Inteligência Artificial é o estudo das faculdades mentais através do uso de modelos computacionais" [5]. Outros se recusam a propor uma

definição para o termo e preferem estabelecer os objetivos da IA: "tornar os computadores mais úteis e compreender os princípios que tornam a inteligência possível" [6].

1.1.1 História e épocas da IA [1]

Existem duas linhas principais de pesquisa para a construção de sistemas inteligentes: *a linha conexionista e a linha simbólica*. A *linha conexionista* visa à modelagem da inteligência humana através da simulação dos componentes do cérebro, isto é, de seus neurônios, e de suas interligações. Esta proposta foi formalizada inicialmente em 1943, quando o neuropsicólogo McCulloch e o lógico Pitts propuseram um primeiro modelo matemático para um neurônio. Um primeiro modelo de rede neuronal, isto é, um conjunto de neurônios interligados, foi proposto por Rosenblatt. Este modelo, chamado Perceptron, teve suas limitações demonstradas por Minsky e Papert [7] em livro onde as propriedades matemáticas de redes artificiais de neurônios são analisadas. Durante um longo período essa linha de pesquisa não foi muito ativa, mas o advento dos microprocessadores, pequenos e baratos, tornou praticável a implementação de máquinas de conexão compostas de milhares de microprocessadores, o que, aliado à solução de alguns problemas teóricos importantes, deu um novo impulso às pesquisas na área. O modelo conexionista deu origem à área de redes neuronais artificiais.

A linha simbólica segue a tradição lógica e teve em McCarthy e Newell seus principais defensores. Os princípios dessa linha de pesquisa são apresentados no artigo *Physical symbol systems* de Newell [8]. O sucesso dos sistemas especialistas (SE) (do inglês, "expert system"), a partir da década de setenta, estabeleceu a manipulação simbólica de um grande número de fatos especializados sobre um domínio restrito como o paradigma corrente para a construção de sistemas inteligentes do tipo simbólico. Para facilitar a apresentação, vamos dividir a história da IA simbólica em "épocas", conforme proposto em relatórios internos do MIT (Massachusetts Institute of Technology):

Clássica (1956-1970)

- Objetivo: simular a inteligência humana.

- Métodos: solucionadores gerais de problemas e lógica.
- Motivo do fracasso: subestimação da complexidade computacional dos problemas.

Romântica (1970-1980)

- Objetivo: simular a inteligência humana em situações pré-determinadas.
- Métodos: formalismos de representação de conhecimento adaptados ao tipo de problema, mecanismos de ligação procedural visando maior eficiência computacional.
- Motivo do fracasso: subestimação da quantidade de conhecimento necessária para tratar mesmo o mais banal problema de senso comum.

Moderna (1980-1990)

- Objetivo: simular o comportamento de um especialista humano ao resolver problemas em um domínio específico.
- Métodos: Sistemas de regras, representação da incerteza, conexionismo.
- Motivo do fracasso: subestimação da complexidade do problema de aquisição de conhecimento.

A seguir, pretende-se contextualizar o presente Projeto de Conclusão de Curso dentro da época Moderna de estudos da Inteligência Artificial.

1.1.2 A época Moderna de estudos em IA e o projeto de "Classificação Automática de Dados com *Support Vector Machines* (SVMs) a Partir de Bases de Dados Incompletas"

Pode-se verificar a partir da definição atual do que é Inteligência Artificial (item 1.1.1 – História e épocas da IA) que os métodos utilizados para "simular o comportamento de um especialista humano ao resolver problemas específicos" podem ser exemplificados por representação da incerteza e conexionismo.

Como será detalhado a seguir, o projeto estruturado pelo presente texto tem o objetivo de construir uma máquina inteligente (uma *Support Vector Machine* (SVM),

que será estudada nas seções a seguir) capaz de classificar automaticamente dados a partir de uma base de dados incompleta. Como esta base de dados a ser classificada estará incompleta, deve-se ter aí uma definição e metodologia de como representar as incertezas contidas nos dados desta base, já que estes não estão completos. Dessa forma, a classificação dos dados pela SVM construída deverá ser acompanhada de dados estatísticos, a fim de mostrar quais probabilidades envolvidas nas classificações de dados de uma base incompleta.

Adicionalmente, como os dados deverão ser classificados, a SVM construída deverá entender, ou aprender, quais semelhanças existentes entre dados de uma mesma classe, e classificá-los de forma coerente. Entenda-se por coerência o fato de que dados de uma mesma classe devem possuir analogias entre si, ou deve haver conexionismo entre dados de uma mesma classe. Por exemplo, quando se diz que uma *maçã* pertence à classe *frutas* é porque intrinsecamente os seres humanos sabem que existe certo *conexionismo* entre as características de uma maçã e outras frutas, a partir de um aprendizado contínuo durante sua vida que lhe fornece conhecimento de mundo suficiente para esta simples classificação. Esse conexionismo entre os dados deve ser passado à máquina inteligente através de um algoritmo de aprendizagem, ou treinamento (que será mais detalhado posteriormente), apresentando à máquina os dados a serem classificados, e esta extraindo o conexionismo existente entre os elementos desse conjunto de treinamento. Isto seria análogo a uma criança, através da observação diária do mundo, aprendendo a diferenciar sabores doces, salgados, amargos e azedos dentro de um conjunto de alimentos a ela oferecida durante sua fase de aprendizagem.

Na próxima seção, são apresentados maiores detalhes do que são as *Support Vector Machines*, ou os classificadores que serão utilizados no presente Projeto de Conclusão de Curso.

1.2 SVMs

1.2.1 SVMs – Pequena introdução sobre as pesquisas iniciais que resultaram neste tipo de classificador de dados

O estudo de classificadores para o reconhecimento de padrões vem assumindo papel relevante para a análise de dados, e cada vez mais novas técnicas são introduzidas e aperfeiçoadas para o melhor aproveitamentos desses dados e para a correta análise e classificação destes. Um dos recentes métodos propostos consiste em uma classificador de dados chamado *Support Vector Machines*. *Support Vector Machines (SVM)* é um tipo de técnica que pode ser usada para classificação de padrões e regressão linear, proposta por Vapnik (Boser,Guyon e Vapnik,1992 [3];Cortes e Vapnik,1995 [4];Vapnik,1995 [5],1998 [6]). Qualquer que seja a tarefa de aprendizagem, o SVM fornece um método para controlar a complexidade, independente da dimensionalidade do problema.Para evitar a necessidade de se definir e calcular os parâmetros do hiperplano ótimo em um espaço de alta dimensionalidade, enfoca-se a forma dual do *perceptron*, o qual não usa o vetor peso e sim o produto interno das entradas, transformando o problema original em um problema de otimização. Usando um núcleo de produto interno adequado (*kernel*), o SVM calcula automaticamente todos os parâmetros importantes da rede relativos àquela escolha de núcleo.

1.2.2 SVMs

Nos anos 90, um novo tipo de estudo de algoritmos foi desenvolvido, baseado em resultados de teorias de aprendizagem de estatística: os chamados *Support Vector Machines* (SVMs). Tal aprendizado foi se desenvolvendo até uma nova classe de máquinas de aprendizagem que usa o conceito central de SVMs, que são os *kernels*, para um grande número de rotinas.

Kernels machines providenciam uma área de trabalho (*framework*) modular que podem ser adaptadas a diferentes tarefas e domínios por uma escolha de uma função *kernel* e um algoritmo base. Atualmente, o conceito de SVM e os algoritmos

envolvidos vêm substituindo as redes neurais em campos variados, incluindo engenharia, *information retrieval*, e bioinformática [9, 10].

Support Vector Machine foi proposta recentemente como um método muito eficaz para reconhecimento de padrões de propósito geral. (Pontil and Verri, 1998; Guo et al., 2000). A idéia principal do SVM (Vapnik, 1995) é a seguinte: dado um conjunto de pontos intuitivamente que pertençam a qualquer uma de duas classes, uma SVM encontra o hiperplano que separa a maior fração possível de pontos da mesma classe para o mesmo lado, enquanto maximiza a distância de cada classe do hiperplano. Para o caso de múltiplas classes é utilizado um grafo acíclico direcionado de decisão em que seu nó inicial contém a lista de todas as classes e cada nó subseqüente elimina uma das classes da lista. E esse processo termina quando resta apenas uma classe na lista. Então, para um problema com N classes, N – 1 decisões são feitas. .

1.3 Aplicações das SVMs

Neste tópico, são introduzidas algumas aplicações de reconhecimento de padrões (*pattern recognition*) que já existem e empregam inteligências artificiais, como as SVMs, para desempenhar o papel de classificadores.

1.3.1 Detecção de características faciais e reconhecimento de faces em imagens 2D e 3D

O reconhecimento de faces é um dos mais importantes meios utilizados em aplicações para controle de acesso a lojas/edifícios e tem sido bastante pesquisado nos últimos anos. Entretanto, poucos classificadores tem atingido uma performance confiável devido à dificuldade de distinguir as diferenças individuais daqueles que tem a mesma configuração facial e ainda compete com variações diversas na aparência de uma face em particular devido a mudanças na postura, iluminação, uso de maquiagem e expressão facial.

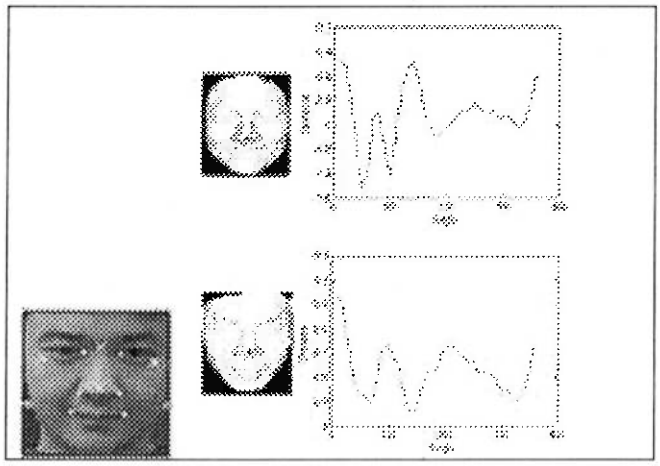


Figura 1-Detecção de características faciais e reconhecimento de faces.

1.3.2 Reconhecimento de dígitos escritos à mão

O reconhecimento de dígitos escritos à mão constitui uma aplicação típica de reconhecimento de padrões empregada em equipamentos como PALM em que as classes são os algarismos numéricos. Por meio de dígitos escritos à mão, é possível identificar o algarismo mais plausível a que a escritura se assemelha.

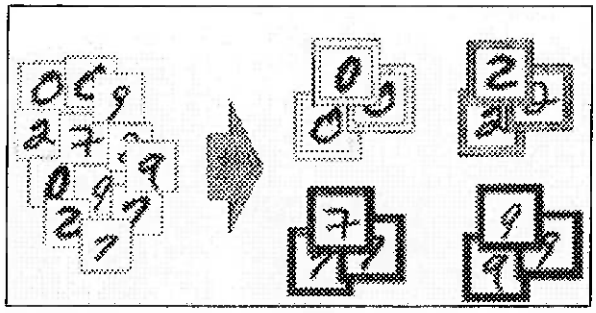


Figura 2 - Treinamento de classificação de dígitos escritos à mão.

Na Figura 2 acima, dado uma amostragem de imagens de quatro diferentes classes "zero", "dois", "sete" e "nove" a treinamento consiste em se procurar uma função que mapeia as imagens para as suas classes correspondentes (indicada pelo formato diferente das bordas em volta dos algarismos).

1.3.3 Verificação da autenticidade de assinaturas

Com os recursos tecnológicos disponíveis hoje no mercado, é possível se verificar a autenticidade de uma assinatura comparando-a com um original por meio de uma inteligência artificial que pode usufruir-se de algoritmos baseado em SVM.

2 Objetivos do Projeto de Conclusão de Curso

O presente projeto de conclusão de curso teve como objetivo construir uma *SVM* que implementa um classificador de dados a partir de uma base de dados incompleta, além de validar a performance desse classificador.

A princípio foi feito um estudo detalhado sobre SVMs, por meio das bibliografias encontradas, buscando conhecimentos sobre todos os tópicos que envolvam uma *Support Vector Machine*, como algoritmos de otimização e teorias estatísticas, além de definições sobre as principais características estruturais e funcionais das SVMs.

Posteriormente, construi-se um algoritmo que implementa uma SVM a partir de uma base de dados incompleta (treinamento semi-supervisionado), denominada *SVMsst (Support Vector Machine – semi-supervised, trained)*. Depurou-se e foi realizado testes dessa SVM a partir de bases padronizadas (*UCI Database Repository* [12]) para a validação do algoritmo e levantamento de desempenho.

Por fim, o projeto foi finalizado comparando-se as performances de SVMs treinadas de formas distintas:

- treinamento com base de dados completa;
- treinamento com base de dados incompleta (treinamento semi-supervisionado) e;
- treinamento semi-supervisionado seguido de re-treinamento.

Para ilustrar o que são bases de dados *completas* e *incompletas*, têm-se as seguintes explanações.

2.1 Base de dados completa e classificada

Um exemplo de base completa poderia ser a seguinte classificação de ações de uma bolsa de valores em duas classes:

Ação / Características	Tendência de valorização para o dia posterior?	Variação nos últimos 21 dias	Variação nos últimos 3 dias	Imagem da empresa (nota de 0 a 10 segundo pesquisa do órgão A)
Embratel	Não	-20%	+3%	5
Petrobrás	Sim	+1%	+2	7
Perdigão	Sim	+2	+3%	9
Kablin	Não	+10%	+6	7
Itaú	Sim	+21%	+4	8

Tabela 1- Base de dados completa

Como é possível observar, a base de dados está completa, e totalmente *classificada*. Esta base pode ser utilizada para treinar uma SVM que seja capaz de inferir a *tendência de valorização para o dia posterior*, ou seja, *classificar* para outras ações e outros dias dadas as características daquele papel para determinado dia.

Por exemplo, esta base pode ser montada a partir de dados reais coletados durante duzentos dias de pregão em uma bolsa de valores.

2.2 Base incompleta

Diferentemente da base citada anteriormente, uma base incompleta pode ser assim ilustrada:

Ação / Características	Tendência de valorização para o dia posterior?	Variação nos últimos 21 dias	Variação nos últimos 3 dias	Imagem da empresa (nota de 0 a 10 segundo pesquisa do órgão A)
Embratel	Não	-20%	+3%	5
Petrobrás	?	+1%	+2	7

Perdigão	?	+2	+3%	9
Kablin	Sim	+10%	+6	7
Itaú	?	+21%	+4	8

Tabela 2 - Base de dados incompleta

Pode-se notar que a base de dados não está completamente classificada como a anterior, estando assim *incompleta.*

Dessa forma, o objetivo de nosso Projeto de Conclusão de Curso se situa nesse problema: como treinar uma SVM a partir de uma base de dados ***incompleta?***

3 Alguns conceitos introdutórios sobre SVMs

Um sistema inteligente, ou um sistema que implemente Inteligência Artificial para a resolução de um certo problema, como classificação de dados de um certo conjunto, cujo contexto é o mesmo deste projeto, pode ser estabelecido pela seguinte relação de conexionismo:

$$\boxed{Y = w.X}$$

na qual X é um dado a ser classificado (um vetor, por exemplo), w é o conhecimento adquirido pela máquina inteligente do conexionismo entre os dados do conjunto de treinamento (uma matriz, por exemplo), e Y é a classe a qual X pertence, segundo decisões da máquina a partir de seu conhecimento sobre os dados (w).

O conjunto de treinamento de uma máquina inteligente pode ser dado por um conjunto de pares entradas e saídas da seguinte forma:

$$\{X_1 \rightarrow Y_1, X_2 \rightarrow Y_2, ..., X_m \rightarrow Y_m\}$$

na qual m é o número de pares do conjunto de treinamento. A partir da apresentação desse conjunto de treinamento, a máquina inteligente deve extrair todo o conexionismo existente entre estes pares, e construir a matriz w, ou seja, seu conhecimento do conjunto de treinamento previamente apresentado.

Atualmente as *kernel machines* - conjunto de máquinas inteligentes no qual estão contidas as SVMs - representam um sistema inteligente a partir da relação:

$$\boxed{Y(X) = \sum_{i=1}^{m} \alpha_i K(X_i, X)}$$

em que $K(X_i, X)$ é chamado de ***kernel*** do conjunto de treinamento, ou seja, uma definição do conexionismo dentre os dados do conjunto de treinamento. Por exemplo, $K(X_i, X)$ pode ser simplesmente o produto escalar entre X_i e X. α_{is} podem ser constantes que apenas ajustem o mapeamento entre o vetor X e Y através do

kernel do conjunto de treinamento. É relevante explicitar que X é um dado externo ao conjunto de treinamento, mas que possui analogias com os elementos deste, e que deve ser classificado de forma semelhante aos pares $X_i \rightarrow Y_i$ apresentados à máquina.

Com todas as definições acima demonstradas, percebe-se que o sistema inteligente não mais terá que montar, diretamente, a matriz w para adquirir conhecimento sobre o conjunto de treinamento: ele utilizará o *kernel* entre os dados do conjunto de treinamento, que também expressa o conexionismo entre os dados deste conjunto, determinando posteriormente os α_{is} necessários ao correto mapeamento $X \rightarrow Y$. Assim, a máquina faz proveito do conexionismo intrínseco entre os dados de treinamento, ou do *kernel* destes.

Dentro do contexto deste Projeto, os vetores X seriam dados incompletos a serem classificados dentro de classes Y.

3.1 Poder de generalização

O poder de generalização é definido como a capacidade da máquina inteligente em reconhecer padrões pertencentes ao universo da base de dados, após a fase de aprendizagem, mas não fornecidos para o treinamento.

4 Support Vector Machines (SVMs)

Como descrito anteriormente, a principal idéia de SVM é: dado um conjunto de pontos intuitivamente que pertençam a qualquer uma de duas classes, um algoritmo de SVM encontra o hiperplano que separa a maior fração possível de pontos da mesma classe para o mesmo lado, enquanto maximiza a distância de cada classe do hiperplano, como mostra a Figura 3 seguinte:

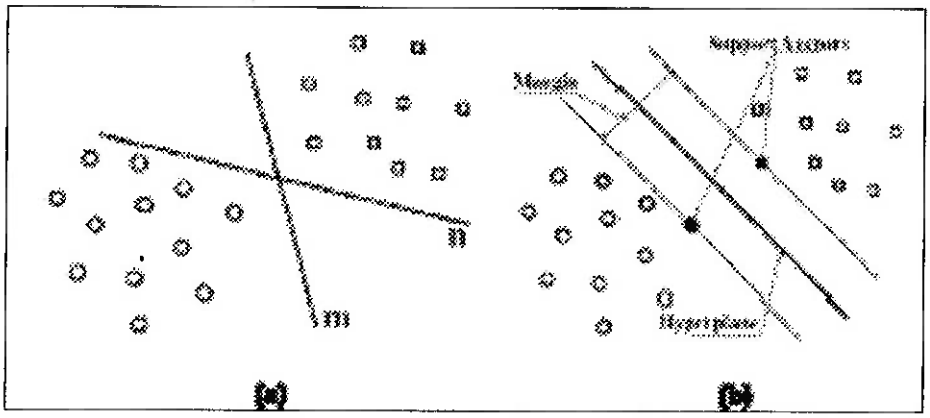


Figura 3 - (a) Separação dos objetos em diferentes classes (b) Hiperplano ótimo que melhor separa os objetos de classes diferentes.

Essa maximização da distância que separa cada classe é chamada de margem. A margem delimita o espaço das *Support Vectors* (como mostrado na Figura 3) que são os vetores de apoio do hiperplano para a separação das classes.

Antes de se descrever o processo de classificação de dados por SVM, é substancialmente relevante se estabelecer alguns pré-conceitos básicos de definição para o estudo de SVM. Primeiramente, o aprendizado de classificação é um problema em se encontrar uma estratégia-ótima para assimilar classes a objetos, baseado em observações passadas de pares objeto-classe.

Em todo o trabalho a seguir, assumiremos que todos os objetos x estão contidos em um conjunto χ, referenciado como o espaço de entrada (ou, simplesmente, conjunto de treinamento)..

E o espaço de saída é o conjunto finito de classes que referenciaremos como Y. Para o caso de duas classes, considera-se apenas o espaço de saída de dois

elementos {-1, +1}. Neste caso o problema de classificação é um aprendizado de **classificação binária**.

Em resumo, suponha-se que é dado que uma amostragem de m objetos treinados,

$$x = (x_1, x_2, ..., x_m) \in \chi^m,$$

juntamente com uma amostragem das classes correspondentes de cada objeto,

$$y = (y_1, y_2, ..., y_m) \in Y^m.$$

Então, a amostragem a ser utilizada para o treinamento é definida como:

$$z = (x, y) = ((x_1, y_1), (x_2, y_2), ..., (x_1, y_1)) \in (\chi \times Y)^m = Z^m,$$

em que z é identicamente e independentemente distribuído (iid) de acordo com algumas medidas de probabilidade **Pz** desconhecidas [9,10].

Assim, o problema de treinamento é encontrar uma relação funcional desconhecida $f \in Y^{\chi}$ entre os objetos $x \in \chi$ e as classes $y \in Y$ baseado na amostragem $z = (x, y) = ((x_1, y_1), (x_2, y_2), ..., (x_1, y_1)) \in (\chi \times Y)^m = Z^m$, de tamanho m $\in N$. Se o espaço de saída Y contém um número finito $|Y|$ de elementos, então trata-se de uma classificação automática de dados.

Outra definição importante é a diferença entre *treinamento supervisionado* e *semi-supervisionado*. No caso em que para cada elemento de z (ou seja, um vetor x_i qualquer) exista um único y_i do conjunto Y diretamente relacionado para ser apresentado ao classificador de dados, trata-se de *treinamento supervisionado*. Se para algum x_i não existir nenhum y_i diretamente relacionado, então ter-se-á um treinamento semi-supervisionado, já que o classificador deverá inferir quais classes y_i atribuir aos vetores x_i não classificados.

Uma forma de resolver o problema acima é a partir de uma metodologia chamada *transdução*, que será apresentada a seguir.

As definições tratadas a seguir fundamentam um embasamento completo que formulam os critérios estruturais de uma SVM.

4.1 Feature Space

Uma função $\phi_i : \chi \rightarrow \Re$ que mapeia cada objeto $x \in \chi$ a um valor real $\phi_i(x)$ é chamada de "*feature*". Combinando *n features* $\phi_1,...,\phi_n$ resulta-se em um mapeamento de *feature* $\phi : \chi \rightarrow K \subseteq \ell_2^n$ e o espaço K é chamado de espaço *feature (feature space)*.

O vetor de *feature* ϕ é também chamado de representação de $x \in \chi$ no espaço K. Daqui para frente, $\phi(x_i)$ será simplesmente chamado de $\mathbf{x_i}$.

Como ilustração, verifica-se a partir da Figura 4 que a escolha de um *feature* ϕ correto garantiu, depois do mapeamento, separabilidade linear a uma base anteriormente não separável por uma reta.

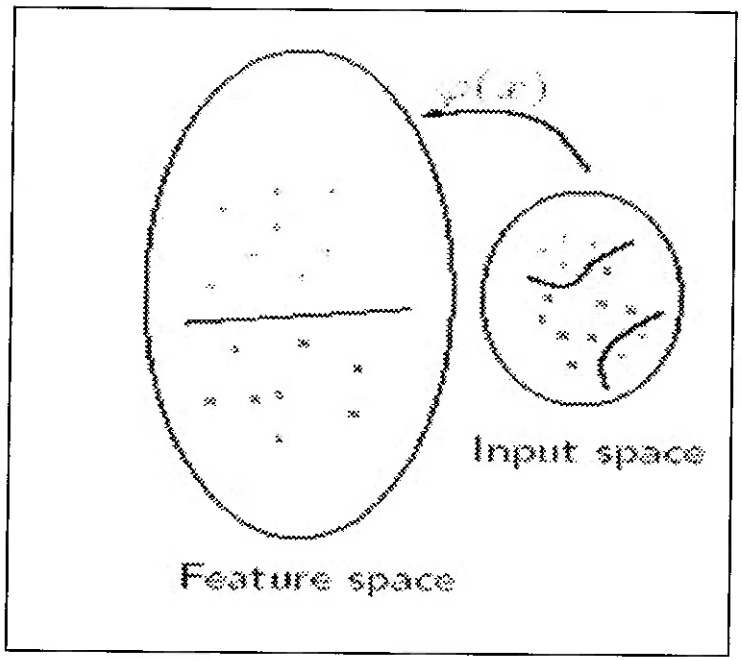


Figura 4 – *Feature space*

4.2 Kernel

Suponha que é dado um mapeamento de *feature* $\phi: \chi \to \mathrm{K} \subseteq \ell_2^n$. O *kernel* é uma função de produto interno *k:* $\chi \times \chi \to \Re$ em K, i.e., para todo x_i , $x_j \in \chi$,

$$k(x_i, x_j) \overset{def}{=} \langle \phi(x_i), \phi(x_j) \rangle$$

4.3 Matriz Gram

Dado um *kernel k*: $\chi \times \chi \to \Re$ e um conjunto de dados de amostragem $x=(x_1,...x_m) \in \chi^m$ de m objetos em χ , chama-se matriz **G**, *m* x *m* como:

$$G_{ij} \overset{def}{=} k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$

a matriz Gram de *k* em *x*.

Pela definição acima, observa-se que a matriz Gram é o m-vetor dimensional da "*kernel evaluation*" entre os objetos treinados x_i e um novo objeto-teste $x \in \mathrm{K}$ satisfazem o aprendizado e a classificação, respectivamente. É importante ressaltar que a matriz Gram e a *feature space* são chamadas de matriz kernel e espaço kernel, respectivamente.

A definição acima de matriz Gram deve ser utilizada para a implementação de SVMs do tipo *hard margin* (ou *margem rígida*), ou seja, quando da utilização de conjuntos de treinamento *linearmente separáveis* [9,10] e Figura 4. Quando os conjuntos podem não ser linearmente separáveis, então a matriz Gram deve ser modificada, para que uma SVM do tipo *soft margin* (ou *margem flexível*) seja implementada. Assim, a nova matriz Gram para *soft margin* pode ser descrita pela seguinte equação:

$$\tilde{G}_{new} \overset{def}{=} G_{old} + \lambda.m.I$$

em que: G_{new}: é a nova matriz Gram modificada para *soft margin*;

G_{old}: é a matriz Gram para *hard margin;*

λ: é um parâmetro de compromisso (λ>0) que fornece a medida de flexibilidade da margem, uma vez que para λ →0, tende- se ao caso de *hard margin* [9,10];

m: é o número de vetores do conjunto de treinamento;

I: matriz identidade;

O presente Projeto considera em suas equações, sempre uma matriz Gram capaz de implementar o caso de *soft margin*. Dessa forma, a função *SVMsst* pode implementar uma SVM a partir de treinamento por um conjunto de vetores não linearmente separáveis (Apêndice A).

4.4 Maximizando a margem

De acordo com a Figura 5, uma amostragem treinada z em R^2 juntamente com um classificador é mostrado.

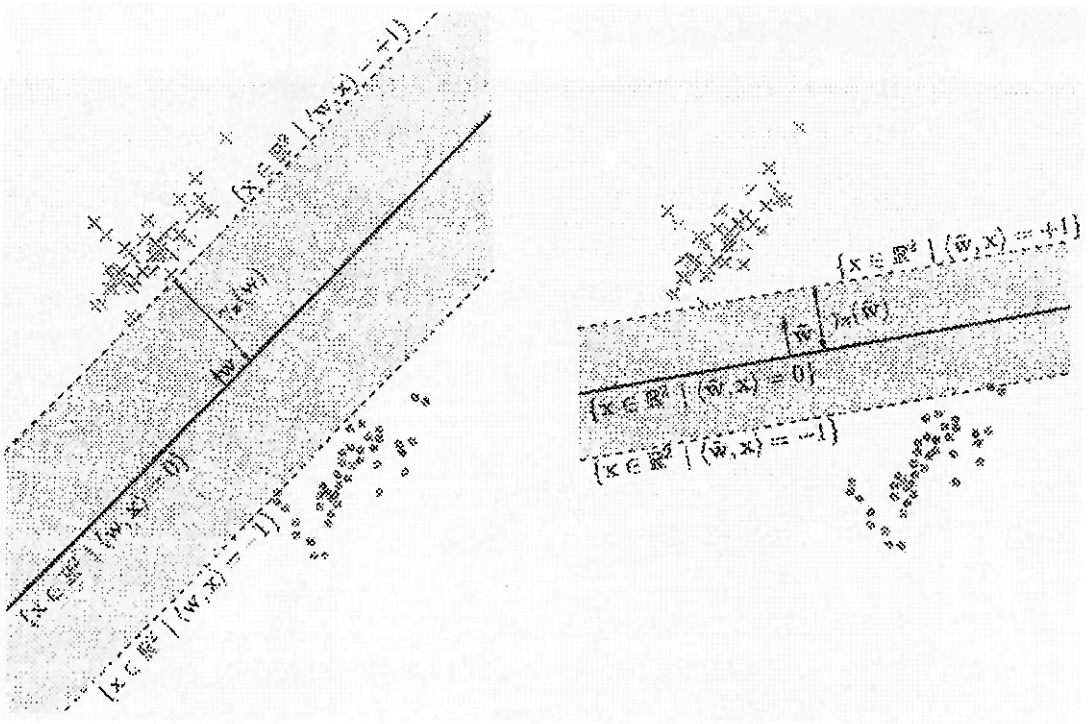


Figura 5 - Plano das margens geométricas em R^2. À esquerda (Fig 5.a), o classificador f_w com a maior margem geométrica $\gamma_z(w)$; e à direita (Fig. 5b) com a menor margem geométrica.

O classificador f_w na Figura 5(a) tem uma "zona morta", separando os dois conjuntos de pontos, maior que o classificador $f_w^{\sim}$ na Figura 5(b). Em ambas figuras, a "zona morta" é o tubo em volta da superfície de decisão (linear) que não contém nenhum objeto treinado $(x_i, y_i) \in z$. Para se medir a extensão do tubo , pode-se usar a norma do vetor peso **w**, parametrizando o classificador f_w. De fato, o tamanho do tubo precisa ser inversamente proporcional ao mínimo valor real de saída , $y_i \langle x_i, w \rangle$ do classificador **w** numa amostragem treinada z.Este quantificador é também conhecido como margem funcional numa amostragem treinada z e precisa ser normalizado para ser utilizável para comparações de diferentes vetores peso **w**, não necessariamente de comprimento unitário.

4.5 Vetor $\mathbf{w}_{svm}$

O problema em se determinar o vetor $\mathbf{w}_{svm}$, se torna um problema de minimização do funcional risco em uma programação convexa, obtendo-se o seguinte funcional:

$$\textit{minimizar } \|\mathbf{w}_{svm}\|^2 = \|f_w\|^2,$$

$$\textit{tal que } y_i.\langle \mathbf{x}_i, \mathbf{w}_{svm} \rangle \geq 1,$$

$i = 1,...,m$ (m é o número de vetores do conjunto de treinamento),

na qual $\mathbf{w}_{svm}$ é o vetor de pesos da SVM, f_w é a função implementada pela SVM, y_i é a classe atribuída ao vetor x_i, $\mathbf{x}_i$ é o mapeamento de x_i no espaço *feature* Φ selecionado, ou seja, $\mathbf{x}_i = \Phi(x_i)$, e $\langle \mathbf{x}_i , \mathbf{w}_{svm} \rangle$ é o produto interno entre os vetores $\mathbf{x}_i$ e $\mathbf{w}_{svm}$ (neste caso, é o produto escalar entre os vetores).

No caso, a função objetivo é quadrática e as restrições são lineares. Como consequência, a solução precisa ser expressiva em sua forma dual. Introduzindo os m multiplicadores de Lagrange α_i , a solução $\mathbf{w}_{svm}$ por ser obtida através do seguinte funcional, denominado de *Wolfe Dual*:

$$\left(\tilde{w}, \tilde{\alpha}\right) = \arg\min_{w \in K} \arg\max_{0 \le \alpha} L(w, \alpha)$$

$$L(w, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i y_i \langle x_i, w \rangle + \alpha' 1$$

Derivando em w, obtém-se:

$$\left.\frac{\partial L(w, \alpha)}{\partial w}\right|_{w=\hat{w}} = \tilde{w} - \sum_{i=1}^{m} \alpha_i y_i \phi_i = 0 \Leftrightarrow \tilde{w} = \sum_{i=1}^{m} \alpha_i y_i \phi_i$$

Substituindo na função Lagrangeana, a solução (chamada de Wolfe dual) fica:

$$\begin{aligned} W(\alpha) &= \alpha' 1 - \frac{1}{2}\alpha' Y G Y \alpha \\ \tilde{\alpha} &= \arg\max_{0 \le \alpha} W(\alpha) \end{aligned}, \qquad (1)$$

em que G é a matriz Gram m x m e Y é a matriz diagonal $(y_1, y_2, \ldots, y_m)$.

Portanto a determinação de $\mathbf{w_{svm}}$ é:

$$w_{svm} = \sum_{i=1}^{m} \alpha_i \tilde{y}_i \phi_i ,$$

cujo esforço computacional pode ser dado por $O(m^2)$.

4.6 Classificação para o caso de duas classes

Para um novo dado de ponto *x*, a classificação é simplesmente resolvida pelo seguinte produto escalar:

$$\langle w_{svm}, \varphi(x) \rangle ,$$

em que w_{svm} é o vetor encontrado do treinamento da base de dados e $\varphi(x)$ é a *feature* do vetor de entrada *x*.

Caso o valor retornado esteja no intervalo $-1<\langle w_{svm}, \varphi(x)\rangle <1$, não atribui-se uma classificação ao ponto dado (x) uma vez que esse intervalo corresponde a faixa dentro da *soft margin*, chamada de *dead zone*.

Entretanto se o valor do produto escalar $\langle w_{svm}, \varphi(x)\rangle$ for menor ou igual a -1, atribui-se ao ponto (x) a classe correspondente a -1. Da mesma forma, se o produto escalar for maior que 1, a classe do ponto (x) será a correspondente a 1.

4.7 Caso de multi-classes

Um sistema de reconhecimento de padrões multi-classes pode ser obtido combinando-se duas classes SVMs. Nesse estudo, emprega-se uma arquitetura de aprendizado: *Decision Directed Acyclic Graph (DDAG)*. DAG é um grafo cujos limites possuem uma orientação e não ciclos. O DDAG é equivalente a operar em uma lista, onde cada nó elimina uma classe da lista. A lista é inicializada com uma lista de todas as classes. Um ponto teste é avaliado contra um nó de decisão que corresponde ao primeiro e último elemento da lista. Se um nó prefere uma das duas classes, a outra classe é eliminada da lista, e o DDAG procede a testar o primeiro e último elemento de uma nova lista. A DDAG termina quando somente uma classe resta na lista. Assim, para um problema de N classes, N-1 nós de decisão serão avaliados em ordem até obter um resultado. Decisão DAG permite uma representação mais eficiente de redundâncias e repetições que ocorrem em diferentes marcas de árvores.

Para maiores detalhes sobre implementação para casos de multi-classes, consulte [9,10].

5 Implementação de SVM de treinamento supervisionado

5.1 Implementação de uma SVM que trabalha apenas com treinamento supervisionado (bases completas)

Esta implementação e testes foram executados no início das atividades deste presente Projeto, como forma de se interar com as características funcionais e estruturais de SVMs.

Tomando-se como base os algoritmos fornecidos nas referências [9, 10] que implementam SVMs, uma SVM foi construída sobre a plataforma MatLab, que trabalha apenas com dados *linearmente separáveis*[1] e classes binárias (1 ou –1). Adicionalmente, o treinamento desta SVM pode ser feito apenas de forma *supervisionada*, ou seja, apenas com *bases classificadas*.

5.2 Testes e resultados obtidos

Este algoritmo foi testado com uma base de dados padrão retirada do site *UCI Database Repository* [12], chamada *Johns Hopkins University Ionosphere database.* Esta base de dados continha 350 padrões e 34 parâmetros, representando dados coletados de um sistema de radar em *Goose Bay, Labrador* (EUA). Quando os dados realmente representassem dados obtidos de sinais vindos da ionosfera, estes eram classificados como *bons* (*classe +1*); caso contrário, em *ruins* (*classe -1*).

A base de dados foi utilizada para o teste da SVM implementada de acordo com a metodologia *k-fold cross validation*. Esta divide a base de dados em *k* conjuntos de padrões; cada pequeno conjunto então é separado para teste, e os restantes são fornecidos à SVM como um conjunto de treinamento. Após a fase de aprendizagem, o conjunto de teste é usado para verificar o *poder de generalização* da SVM treinada. Repete-se o procedimento para cada fold; posteriormente, tem-se uma média de acertos e um desvio padrão e, por conseguinte, uma medida estatística do poder de generalização da *Support Vector Machine*.

---

[1] Dados linearmente separáveis podem ser exemplificados em um caso no espaço IR^2: os dados podem ser *separados em duas classes por uma única reta.* Vide Figura 4 como ilustração.

Primeiramente, a base de dados original foi utilizada com *2-fold validation*. Os resultados obtidos são expressos pela Tabela 3:

Fold	*Porcentagem de acertos*
1	54.2857
2	88.5714
	Média de acertos: (71.43 ± 24.24) %

Tabela 3 – Resultados para cada *fold* utilizando a base de dados original (350 padrões)

Posteriormente, a base foi truncada em 150 padrões, e utilizada com *10-fold validation*. Isto foi feito para que o treinamento e testes da SVM fossem feitos com menor tempo computacional. Os resultados estão expostos na Tabela 4:

Fold	*Porcentagem de acertos (%)*
1	66,67
2	66,67
3	66,67
4	80,00
5	53,33
6	53,33
7	66,67
8	66,67
9	53,33
10	73,33
	Média de acertos: (64.67 ± 8.92) %

Tabela 4 – Resultados obtidos para cada fold com a base de dados truncada em 150 padrões.

No primeiro teste, como o número de conjuntos de testes (*k-fold*) era pequeno, o desvio padrão é alto, representando quase 35% da média de acertos. Dessa forma, a média de acertos obtida para este caso não representa bem a capacidade de generalização da SVM implementada.

Já no segundo caso, o desvio padrão representa quase 15% da média de acertos. Neste caso então, a média pode ser utilizada para a representação do poder de generalização da SVM. Levando-se em conta que a base foi truncada em quase 2/3, assim diminuindo a quantidade de informação passada à SVM, pode-se dizer que o desempenho da SVM é regular, validando a utilização da SVM para a solução deste problema e atestando a confiabilidade do algoritmo escrito.

6 Implementação de SVM de treinamento semi - supervisionado

6.1 Tentativa inicial de Implementação da Support Vector Machine de treinamento semi-supervisionado

Inicialmente, para a implementação de uma SVM de treinamento semi-supervisionado, tentou-se o uso do algoritmo *S3VM*, desenvolvida pelos autores de [11]. Porém, o algoritmo se mostrou numericamente instável, e preferiu-se então partir para o desenvolvimento de uma nova SVM de treinamento semi-supervisionado.

O desenvolvimento desta SVM se baseou numa metodologia para o treinamento semi-supervisionado chamada de *transdução*.

6.2 Transdução

Um problema de transdução pode ser caracterizado pela otimização da SVM a partir de dados previamente classificados, juntamente com as variáveis das classes faltantes da parte não classificada do conjunto de treinamento.

O problema de transdução tem sido exaustivamente estudado por pesquisadores [11,13], pois permite a inclusão de dados não classificados para o treinamento de SVMs, o que caracterizaria treinamento semi-supervisionado.

6.3 Modificação da Support Vector Machine de treinamento supervisionado para receber treinamento semi-supervisionado a partir de transdução – *SVMsst* (SVM semi-supervised-trained)

Baseando-se no funcional W (Equação 1) apresentado para a implementação da SVM de treinamento supervisionado, decidiu-se adaptá-lo para que o *problema de otimização também tratasse a inclusão de vetores não classificados*, otimizando o funcional W em relação aos vetores classificados e vetores não-classificados (treinamento semi-supervisionado) ao mesmo tempo (transdução). Dessa forma, W estaria em função *não somente dos vetores classificados, mas também em função dos vetores não classificados*.

Assim, a matriz diagonal **Y** foi adaptada para que as classes inexistentes dos vetores não classificados fossem consideradas como variáveis dentro desta matriz, da seguinte forma:

$$\mathbf{Y_{transdução}} = \mathbf{Y_{trans}} = \begin{bmatrix} y_1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & y_2 & & & & & & \vdots \\ \vdots & & \ddots & & & & & \vdots \\ \vdots & & & y_l & & & & \vdots \\ \vdots & & & & q_1 & & & \vdots \\ \vdots & & & & & q_2 & & \vdots \\ \vdots & & & & & & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & q_k \end{bmatrix}_{\substack{l+k=m, \\ m \times m}},$$

na qual $y_i,\ldots,y_l$ são as classes dos vetores classificados do conjunto de treinamento, e $q_1,\ldots,q_k$ são variáveis que representam as classes inexistentes dos vetores não classificados. Para receber as adaptações relativas ao treinamento semi-supervisionado, o funcional W_{trans} teve de ser elaborado (com base no funcional W apresentado na equação (1)), e pode ser escrito da seguinte forma:

$$W_{transdução} = W_{trans}(\boldsymbol{\alpha}, \mathbf{q}) =$$
$$\sum_{i=1}^{m=l+k} \alpha_i - \frac{1}{2}\left[\sum_{a=1}^{l} \alpha_a y_{trans_a}\left(\sum_{i=1}^{l} \alpha_i y_{trans_i} G_{i,a} + \sum_{j=l+1}^{m} \alpha_j q_j G_{j,a}\right) + \sum_{b=l+1}^{m} \alpha_b q_b \left(\sum_{i=1}^{l} \alpha_i y_{trans_i} G_{i,b} + \sum_{j=l+1}^{m} \alpha_j q_j G_{j,b}\right)\right], \quad (2)$$

tal que 1) $\boldsymbol{\alpha} \geq \mathbf{0}$

2) $sinal(\mathbf{x}_i' . \mathbf{w}_{svm}) = q_i$, $i = 1,\ldots,k$ (k = número de vetores não classificados)

Note-se que $W_{transdução}$ fica em função não somente de $\boldsymbol{\alpha}$, mas também de $\mathbf{q}$, caracterizando-se assim um problema de transdução.

Outro ponto relevante que fornece coerência para este problema de transdução é a restrição de número 2): *a otimização só está concluída quando a classe q_i, que está sendo atribuída pelo algoritmo de otimização ao vetor x_i não classificado* (membro direito da restrição), *é a mesma classe que está sendo atribuída ao vetor x_i pela SVM* (membro esquerdo da restrição).

6.4 Modificações no algoritmo de transdução

Durante as análises dos experimentos feitos para levantamento das características da SVM de treinamento semi-supervisionado (que serão mostrados a seguir), notou-se que a inclusão de vetores não classificados durante a otimização (transdução) não trazia nenhuma melhora de performance (maior poder de reconhecimentos dos padrões), o que segundo a referência [11] pode acontecer: *ou a transdução melhora a performance da SVM ou se obtém a mesma performance de uma SVM treinada de forma supervisionada a partir da mesma base de dados utilizada na transdução.*

Porém, em alguns casos, se percebe que a porcentagem de acertos das classes que o algoritmo atribui aos vetores não classificados pode chegar a taxas razoáveis (60% a 65%), como será visto na parte experimental. Assim, se caracteriza um questionamento sobre o porquê do não aumento de performance da SVM pela inclusão destes vetores não classificados.

A partir desta observação, propôs-se a seguinte metodologia para treinamento supervisionado da SVM:

1) primeiramente, a partir de transdução, obtém-se as classes dos vetores não classificados;

2) posteriormente, uma nova SVM é treinada de forma supervisionada, com as classes antes inexistentes substituídas pelas classes obtidas quando da resolução do problema de transdução em 1).

Com esta modificação, houve melhora de performance da SVM em alguns casos, comprovando a validade da metodologia proposta. Esta metodologia foi utilizada para a implementação da SVM.

A seguir, serão mostrados os experimentos realizados e os resultados obtidos.

6.5 Testes, resultados e discussões

Nesta seção, serão apresentados os testes executados para validação da SVMsst, os resultados obtidos e algumas discussões para justificar estes resultados.

6.5.1 Bases utilizadas

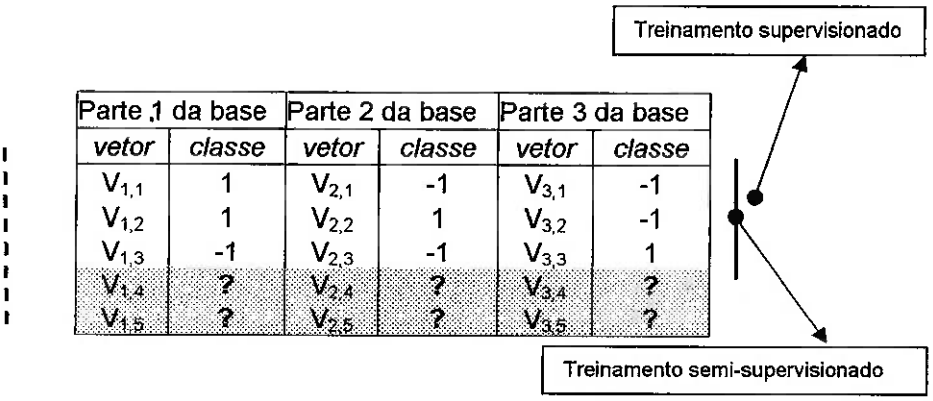
Para os experimentos realizados, foram utilizadas bases de dados do repositório de dados da *UCI Database Repository* [12]. Geralmente, estas bases de dados são utilizadas pela comunidade científica para levantamento de performances e validação de algoritmos de classificadores e reconhecedores de padrão, como em [11]. Deste repositório, duas bases foram utilizadas:

- Johns Hopkins University Ionosphere Database: esta base de dados continha 350 padrões de 34 parâmetros (34 componentes), representando dados coletados de um sistema de radar em *Goose Bay, Labrador* (EUA). Quando os dados realmente representassem dados obtidos de sinais vindos da ionosfera, estes eram classificados como *bons* (*classe +1*); caso contrário, em *ruins* (*classe -1*);
- Sonar, Mines vs. Rocks: esta base continha 208 padrões de 60 parâmetros, representando sinais retornados de um cilindro de metal ou de um cilindro de rocha.

A seguir, serão relatados os experimentos realizados com a SVMsst.

6.5.2 Experimentos realizados

Para a execução dos experimentos, as bases de dados foram divididas em três partes, cada uma destas contendo vetores classificados e não classificados (na verdade, as classes destes vetores eram retiradas quando do treinamento da SVM, sendo que esta os tinha como vetores não classificados; para maiores detalhes, vide Apêndice A). Sendo a SVM treinada a partir de uma destas partes, a base inteira era posteriormente fornecida à SVM para o teste de performance, para que esta classificasse todos os dados da base (inclusive os previamente fornecidos durante o treinamento). A Tabela 5 esquematiza a divisão da base em partes.



Parte 1 da base		Parte 2 da base		Parte 3 da base	
vetor	*classe*	*vetor*	*classe*	*vetor*	*classe*
$V_{1,1}$	1	$V_{2,1}$	-1	$V_{3,1}$	-1
$V_{1,2}$	1	$V_{2,2}$	1	$V_{3,2}$	-1
$V_{1,3}$	-1	$V_{2,3}$	-1	$V_{3,3}$	1
$V_{1,4}$	?	$V_{2,4}$	?	$V_{3,4}$	?
$V_{1,5}$	?	$V_{2,5}$	?	$V_{3,5}$	?

Tabela 5– Resultados obtidos para cada fold com a base de dados truncada em 150 padrões.

Note-se que cada parte da base possui vetores classificados e não classificados, para que treinamentos supervisionados e semi-supervisionados fossem realizados.

Cada uma das partes da base, fornecidas à SVM como conjunto de treinamento supervisionado (parte classificada) e semi-supervisionado (parte classificada mais parte não classificada), permitiam diferentes performances para o reconhecimentos da base inteira, o que permitia o levantamento de uma média de performance mais um desvio padrão, caracterizando-se assim uma medida estatística da performance da SVM treinada em relação à base utilizada. A Figura 6 ilustra o procedimento adotado.

Durante os experimentos, foi também controlado o número de vetores não classificados de cada divisão da base que seriam considerados durante o treinamento

da SVM, tendo-se assim um treinamento *mais* ou *menos* semi-supervisionado, de acordo com o número destes vetores utilizados.

Um outro experimento executado durante os testes tinha o objetivo de fornecer um embasamento experimental para uma intuição adquirida durante o desenvolvimento da SVM: caso a *Support Vector Machine* tivesse uma boa performance para inferir as classes dos vetores não classificados fornecidos como dados de treinamento, o número de acertos para um treinamento com {X vetores classificados + Y vetores não classificados (ou seja, apresentados à SVM sem suas respectivas classes) } deveria ser aproximadamente o mesmo quando da apresentação de {$X + Y$ vetores classificados classificados (ou seja, apresentados à SVM com suas respectivas classes) }, sendo os X e Y vetores considerados sempre os mesmos em ambos os casos.

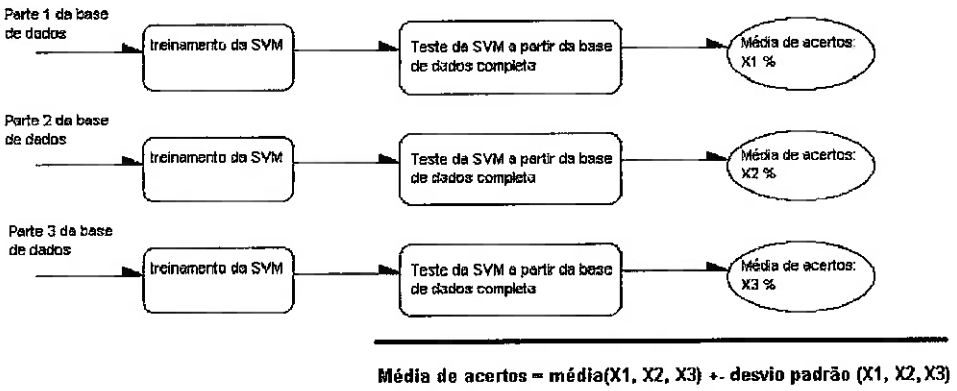


Figura 6 – Procedimento para levantamento de performance da SVMsst. Esta metodologia de teste permitiu o levantamento de estatísticas

6.5.3 Resultados obtidos

A seguir, serão mostrados os resultados obtidos para cada base utilizada e casos de treinamento.

Base de dados *Johns Hopkins University Ionosphere Database*

Para o treinamento da SVM, foram escolhidos os 240 primeiros vetores desta base (que continha 350 vetores). Cada parte continha 80 vetores: a parte 1 continha os 80 primeiros vetores, a parte 2 os 80 próximos e assim por diante. Dentro de cada

parte, os 50 primeiros foram considerados para um treinamento inteiramente supervisionado, e a cada novo experimento, 10 vetores dentro de cada parte eram adicionalmente considerados para treinamento semi-supervisionado, até que se completasse os 80 dentro de cada parte do conjunto de treinamento.

Outro parâmetro importante que também foi medido foi a porcentagem de acertos das classes atribuídas pela SVM aos vetores não classificados

Dessa forma, os resultados obtidos estão mostrados na Tabela 6.

Significado da linha	Descrição da parte 1 da base	Descrição da parte 2 da base	Descrição da parte 3 da base	Resultado geral
Descrição das partes da base – treinamento supervisionado	Parte 1) - 50 vetores classificados	Parte 2) - 50 vetores classificados	Parte 3) - 50 vetores classificados	---
Porcentagens de acerto para a base inteira	66,57 %	69,71 %	74,28 %	70,19 % ± 3,88 %
Descrição das partes da base – treinamento semi-supervisionado	Parte 1) - 50 vetores classificados + 10 vetores não classificados	Parte 2) - 50 vetores classificados + 10 vetores não classificados	Parte 3) - 50 vetores classificados + 10 vetores não classificados	---
Porcentagem de acertos para a atribuição das classes dos vetores não classificados	60 %	60 %	60 %	60 % ± 0 %
Porcentagens de acerto para a base inteira	66,57 %	69,71 %	74,28 %	70,19 % ± 3,88 %
Descrição das partes da base – treinamento semi-supervisionado	Parte 1) - 50 vetores classificados + 20 vetores não classificados	Parte 2) - 50 vetores classificados + 20 vetores não classificados	Parte 3) - 50 vetores classificados + 20 vetores não classificados	---
Porcentagem de acertos para a atribuição das classes dos vetores não classificados	60 %	65 %	60 %	61,67 % ± 2,89 %
Porcentagens de acerto para a base inteira	66,57 %	69,71 %	74,28 %	70,19 % ± 3,88 %
Descrição das partes da base – treinamento semi-supervisionado	Parte 1) - 50 vetores classificados + 30 vetores não classificados	Parte 2) - 50 vetores classificados + 30 vetores não classificados	Parte 3) - 50 vetores classificados + 30 vetores não classificados	---
Porcentagem de acertos para a atribuição das classes dos vetores não classificados	56,67 %	53,33 %	60 %	56,67 % ± 3,34 %
Porcentagens de acerto para a base inteira	66,57 %	69,71 %	74,28 %	70,19 % ± 3,88 %

Tabela 6 – Resultados obtidos com a partir de treinamento semi-supervisionado com a SVMsst, a partir da base *Ionosphere*

Dessa forma, percebe-se que a inclusão de vetores não classificados para o treinamento semi-supervisionado não trouxe melhorias para a performance da SVM quando treinada somente de forma supervisionada, porém com um número menor de vetores de treinamento. Isto realmente pode ocorrer posteriormente ao treinamento semi-supervisionado [11]. Outra conclusão importante que se pode extrair da Tabela 6 é o fato de que o aumento da relação do número *de vetores não classificados para o número de vetores classificados* dentro da base de treinamento utilizada para a transdução pode afetar a performance da SVM em atribuir corretamente as classes não existentes, pois como a SVM utiliza os vetores classificados para inferir as classes dos vetores não classificados (restrição 2) da Equação (2)), a existência de muitos vetores não classificados em relação aos classificados pode fazer com que a SVM fique sem uma boa base de dados classificados para completar satisfatoriamente o aprendizado semi-supervisionado.

De acordo com o comentado no item 6.5.2, para comparar as performances da SVM treinada com {X vetores classificados + Y vetores não classificados} e com {X + Y vetores classificados}, outros experimentos foram executados análogos aos citados no parágrafo anterior, porém fornecendo-se à SVM as classes dos vetores anteriormente considerados como não classificados. Os resultados seguem na Tab. 7.

Significado da linha	**Descrição da parte 1 da base**	**Descrição da parte 2 da base**	**Descrição da parte 3 da base**	**Resultado geral**
Descrição das partes da base – treinamento supervisionado	Parte 1) - 60 vetores classificados	Parte 2) - 60 vetores classificados	Parte 3) - 60 vetores classificados	---
Porcentagens de acerto para a base inteira	77,43 %	70,86 %	78,00 %	75,43 % ± 3,97 %
Descrição das partes da base – treinamento supervisionado	Parte 1) - 70 vetores classificados	Parte 2) - 70 vetores classificados	Parte 3) - 70 vetores classificados	---
Porcentagens de acerto para a base inteira	79,14 %	72,28 %	79,14 %	76,86 % ± 3,96 %
Descrição das partes da base – treinamento supervisionado	Parte 1) - 80 vetores classificados	Parte 2) - 80 vetores classificados	Parte 3) - 80 vetores classificados	---
Porcentagens de acerto para a base inteira	79,14 %	76,28 %	79,14 %	78,19 % ± 1,65 %

Tabela 7 – Resultados obtidos a partir de treinamento supervisionado com a SVMsst, para comparação de performance com o caso de treinamento semi-supervisionado

Percebe-se que para estes experimentos, cujos resultados estão na Tabela 7, que a porcentagem de acertos é maior em relação aos acertos quando do treinamento semi-supervisionado da SVM. Estas diferenças nas porcentagens de acertos obtidas podem ser justificadas pelo tipo de treinamento: no caso da Tabela 6, todos os vetores possuíam suas respectivas classes, enquanto que no primeiro caso, a SVM tinha de atribuir classes a estes vetores durante o treinamento.

De qualquer forma, pode ser observado pela Tabela 6 que a porcentagem de acertos quando da atribuição de classes aos vetores não classificados é razoável, o que deveria contribuir para algum aumento na performance da SVM, o que não ocorreu. As novas porcentagens de acertos deveriam ser análogas àquela quando do treinamento da SVM de forma supervisionada, porém com as classes dos vetores previamente não classificados definidas como sendo as classes atribuídas a estes vetores pela própria SVM durante o treinamento semi-supervisionado. Assim, testes para levantar estas "possíveis" performances foram feitos, cujos resultados podem ser observados de acordo com a Tabela 8.

Significado da linha	Descrição da parte 1 da base	Descrição da parte 2 da base	Descrição da parte 3 da base	Resultado geral
Descrição das partes da base – treinamento supervisionado	Parte 1) - 50 vetores classificados + 10 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	Parte 2) - 50 vetores classificados + 10 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	Parte 3) - 50 vetores classificados + 10 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	---
Porcentagens de acerto para a base inteira	74,86 %	68,28 %	74,28 %	72,47 % ± 3,64 %
Descrição das partes da base – treinamento supervisionado	Parte 1) - 50 vetores classificados + 20 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	Parte 2) - 50 vetores classificados + 20 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	Parte 3) - 50 vetores classificados + 20 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	---
Porcentagens de acerto para a base inteira	72,86 %	69,14 %	72,86 %	71,62 % ± 2,15 %
Descrição das partes da base – treinamento supervisionado	Parte 1) - 50 vetores classificados + 30 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	Parte 2) - 50 vetores classificados + 30 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	Parte 3) - 50 vetores classificados + 30 vetores classificados de acordo com as classes atribuídas a estes vetores pela SVM quando do treinamento semi-supervisionado utilizando-se desta mesma parte da base	---
Porcentagens de acerto para a base inteira	70,57 %	68,28 %	72,00 %	70,28 % ± 1,88 %

Tabela 8 – Resultados obtidos a partir de treinamento semi-supervisionado, seguido de re-treinamento

Da Tabela 8, pode-se notar que as porcentagens são maiores do que os resultados obtidos no caso da Tabela 6, o que evidencia a validade da atribuição de classes por transdução pela SVM durante treinamento semi-supervisionado.

O que pode explicar a não obtenção dos mesmos resultados entre as Tabelas 6 e 8, ou pelo menos resultados bem próximos, é o fato de, durante o treinamento semi-supervisionado, parte do vetor $\mathbf{w}_{svm}$ estar determinado pela parte classificada do conjunto de treinamento, e como as classes atribuídas pela SVM são conferidas pelo vetor $\mathbf{w}_{svm}$ (restricão 2) da equação (2)), estas não acrescentarão maiores conhecimentos sobre a base no vetor $\mathbf{w}_{svm}$. Já no caso da Tabela 8, o posterior treinamento supervisionado depois do respectivo treinamento semi-supervisionado permite a inclusão total dos vetores anteriormente não classificados ao vetor $\mathbf{w}_{svm}$, aumentando o conhecimento da SVM sobre a base de treinamento.

De outra forma, pode-se observar a partir da comparação entre as Tabelas 7 e 8, que mesmo sendo utilizado treinamento supervisionado em ambos os casos, os resultados obtidos são diferentes. Isto pode ser justificado pelo fato de que, no caso da Tabela 8, parte das classes atribuídas a. alguns vetores foram definidos pela própria SVM quando do anterior treinamento semi-supervisionado por transdução, que não garante 100% de eficiência (ou seja, 100% de acertos na atribuição das classes), como visto na Tabela 6.

Base de dados *Sonar, Mines vs. Rocks*

Para o treinamento da SVM, foram escolhidos os 194 primeiros vetores desta base (que continha 208 vetores). Cada parte continha 60 vetores: a parte 1 continha os 60 primeiros vetores, a parte 2 os 60 próximos e assim por diante. Dentro de cada parte, os 50 primeiros foram considerados para um treinamento inteiramente supervisionado, e posteriormente, 10 vetores foram incluídos dentro de cada parte para o treinamento semi-supervisionado, completando-se assim os 60 vetores dentro de cada parte do conjunto de treinamento.

Como no caso da base *Ionosphere*, foi medida a porcentagem de acertos das classes atribuídas pela SVM aos vetores não classificados

Dessa forma, os resultados obtidos estão mostrados na Tabela 9.

Significado da linha	Descrição da parte 1 da base	Descrição da parte 2 da base	Descrição da parte 3 da base	Resultado geral
Descrição das partes da base – treinamento supervisionado	Parte 1) - 50 vetores classificados	Parte 2) - 50 vetores classificados	Parte 3) - 50 vetores classificados	---
Porcentagens de acerto para a base inteira	54,12 %	62,89 %	48,45 %	55,15 % ± 7,28 %
Descrição das partes da base – treinamento semi-supervisionado	Parte 1) - 50 vetores classificados + 10 vetores não classificados	Parte 2) - 50 vetores classificados + 10 vetores não classificados	Parte 3) - 50 vetores classificados + 10 vetores não classificados	---
Porcentagem de acertos para a atribuição das classes dos vetores não classificados	50 %	50 %	50 %	50 % ± 0 %
Porcentagens de acerto para a base inteira	54,12 %	62,89 %	48,45 %	55,15 % ± 7,28 %

Tabela 9 – Resultados obtidos a partir de treinamento semi-supervisionado com a SVMsst, para a base *Sonar*

Como no caso da base *Ionosphere*, percebe-se que a inclusão de vetores não classificados para o treinamento semi-supervisionado não trouxe melhorias para a performance da SVM quando treinada somente de forma supervisionada, porém com um número menor de vetores de treinamento.

Ainda para comparar as performances da SVM treinada com {X vetores classificados + Y vetores não classificados} e com {X + Y vetores classificados}, foram executados experimentos análogos aos citados no parágrafo anterior, porém fornecendo-se à SVM as classes dos vetores anteriormente considerados como não classificados. Os resultados seguem na Tabela 10.

Significado da linha	Descrição da parte 1 da base	Descrição da parte 2 da base	Descrição da parte 3 da base	Resultado geral
Descrição das partes da base – treinamento supervisionado	Parte 1) - 60 vetores classificados	Parte 2) - 60 vetores classificados	Parte 3) - 60 vetores classificados	---
Porcentagens de acerto para a base inteira	52,58 %	61,34 %	49,48 %	54,47 % ± 6,15 %

Tabela 10 – Resultados obtidos para o treinamento supervisionado com a SVMsst, para a base *Sonar*, para comparação com os resultados obtidos com o caso semi-supervisionado

Percebe-se que para estes experimentos, cujos resultados estão na Tabela 10, que a porcentagem de acertos é praticamente igual em relação aos acertos quando do

treinamento semi-supervisionado da SVM. Isto pode ser justificado pela possível falta de relação entre os vetores do conjunto de treinamento, que pode caracterizar uma base ruim para o treinamento da SVM e conseqüente determinação do vetor $\mathbf{w}_{svm}$. Isto pode ser observado pelas porcentagens que se situam em torno de 50%: como a classificação da base é binária, e a distribuição das classes entre os vetores da base está uniformemente distribuída, se a SVM atribuir a todos os vetores uma mesma classe, ter-se-á uma porcentagem de acertos de 50%, isentando a SVM de aprender a relação existente entre os vetores do conjunto de treinamento e a qual se quer incorporar no vetor $\mathbf{w}_{svm}$. Pode se verificar este caso também nas porcentagens de acerto das classes atribuídas aos vetores lançados como não classificados à SVM durante o treinamento semi-supervisionado. Por esse motivo, o experimento de utilizar estas classes atribuídas aos vetores não classificados agora em um treinamento supervisionado não,foi executado para a base *Sonar, Mines vs. Rocks*.

7 Conclusão

Como foi demonstrado na parte experimental deste trabalho, treinamento semi-supervisionado (a partir de transdução) fazendo-se uso de uma base de dados com uma parte não classificada, nem sempre traz melhorias para a performance da SVM (poder de reconhecimento dos padrões apresentados e eventuais padrões não apresentados), em comparação com esta mesma SVM treinada de forma supervisionada, utilizando-se somente da parte classificada da mesma base de dados.

De qualquer forma, o algoritmo aqui implementado (Equação (2)) pode inferir as classes inexistentes dos vetores não classificados, e estas classes atribuídas podem ser reutilizadas em um posterior re-treinamento da SVM (Tabela 8), contribuindo para a melhoria do conhecimento da SVM sobre a base de dados de treinamento e conseqüente aumento de eficiência no reconhecimento de padrões, conforme visto na parte experimental deste texto. Dessa forma, duas metodologias podem ser utilizadas para o treinamento semi-supervisionado: treina-se a SVMsst a partir de bases semi-classificadas, ou, de posse das classes atribuídas pela SVMsst aos vetores não classificados, executa-se um novo treinamento para a SVM, porém agora de forma supervisionada.

Pelos resultados aqui obtidos, é observável que a utilização de vetores não classificados para o treinamento da SVM não prejudica o aprendizado desta, o que indica que o usuário pode fazer uso de uma parte não classificada de sua base de dados para treinar uma SVM, ao invés de somente desconsiderá-la, aproveitando o máximo das informações existentes sobre um certo universo de dados que se deseja incorporar em f_w. Logicamente, como anteriormente verificado (Tabela 6), o número de vetores não classificados em relação ao número de vetores classificados não pode ser tal que o desempenho da otimização seja prejudicado.

Assim, conclui-se que treinamento semi-supervisionado e transdução podem ser utilizados quando não se tem uma base de dados inteiramente classificada, mas se deseja construir um classificador o mais eficiente possível para atividades automáticas de classificação de dados e padrões.

Ainda em tempo, a implementação de algoritmos para a execução de treinamentos semi-supervisionados vêm sendo exaustivamente buscados por vários pesquisadores [11,13], sem a definição, até o momento, de um algoritmo comprovadamente eficiente, o que sugere a dificuldade que existiu para a definição de estratégias de algoritmos na implementação da SVMsst durante o período deste Projeto de Conclusão de Curso.

O usuário de MatLab pode utilizar a SVMsst implementada em *software*, que faz parte deste Projeto de Conclusão de Curso, para a construção de classificadores de dados a partir de bases semi-classificadas por transdução. Maiores detalhes podem ser encontrados na listagem do código do programa no CD entregue junto com este texto, ou listado no Apêndice B, ou no Manual do Usuário, que também está no CD e no Apêndice A.

8 Trabalhos futuros

Como pôde ser visto nos resultados deste trabalho, a performance da SVMsst em inferir as classes não fornecidas para o treinamento semi-supervisionado, pode ditar o desempenho posterior da SVM quanto ao número de acertos na classificação de dados que fazem parte do mesmo espaço vetorial do conjunto de treinamento, não importando se o re-treinamento da SVMsst será feito a partir daquelas classes inferidas.

Dessa forma, um trabalho futuro pode ser *terceirizar* a inferência destas classes através de outros classificadores de dados (como redes Bayesianas ou redes neurais), combinar esta *terceirização* com a já utilizada transdução, ou até reformular o algoritmo de transdução, para que sua capacidade em inferir classes ainda não existentes seja majorada.

Ainda em tempo, a implementação de algoritmos para a execução de treinamentos semi-supervisionados vêm sendo exaustivamente buscados por vários pesquisadores [11,13], sem a definição de um algoritmo comprovadamente eficiente até o momento, o que abre caminhos para idéias inovadoras que ainda poderão surgir neste campo.

9 Referências Bibliográficas

[1] www.lcmi.ufsc.br/gia/history - Texto baseado em

[2] A. Barr and E.A. Feigenbaum, editors. The Handbook of Artificial Intelligence, volume I-II. William Kaufmann Inc., Los Altos, California, 1981.

[3] P. McCorduck. Machines Who Think. Freeman, San Francisco, 1979.

[4] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. Michie and B. Meltzer, editors, Machine Intelligence 4, pages 463-502. Edinburgh University Press, Edinburgh, GB, 1969.

[5] E. Charniak and D. McDermott. Introduction to Artificial Intelligence. Addison-Wesley Publishing Company, Reading, MA, 1985.

[6] P.H. Winston. Artificial Intelligence (2nd Edition). Addison-Wesley Publishing Company, Reading, MA, 1984.

[7] M.L. Minsky and S.A. Papert. Perceptrons: An Introduction to Computational Geometry. M.I.T. Press, 1969.

[8] A. Newell. Physical symbol systems. Cognitive Science, 4:135-183, 1980.

[9] R. Herbrich. Learning Kernel Classifiers – Theory and Algorithms. The MIT Press, 2002.

[10] AJ Smola, G. Rätsch, B. Schölkopf. Learning with Kernels. MIT Press, Cambridge, MA, 2002.

[11] K. P. Bennett, A. Demiriz. Semi-Supervised Support Vector Machines. Rensselaer Polytechnic Institute.

[12] www.ics.uci.edu/~mlearn/MLRepository.html – Repositório de base de dados padronizadas para testes com classificadores de dados.

[13] J. Thorsten. Transductive Inference for Text Classification Using Support Vector Machines. Proceedings of {ICML}-99, 16th International Conference on Machine Learning. Morgan Kaufmann Publishers, San Francisco, US, 1999.

10 Apêndices A e B

Apêndice A

Manual do Usuário SVMsst

Manual do usuário – SVMsst

Manual do usuário para a função MatLab ***SVMsst*** (SVM semi-supervised trained) e ***teste_SVMsst*** (módulo de testes para a SVMsst)

Autores: Clayton Silva Oliveira

Eduardo Takashi Inowe

A seguir, é apresentado um breve manual explicando como usar os códigos-fonte do algoritmo da SVM desenvolvidos para o MATLAB. Convém lembrar a necessidade do programa MATLAB e a disponibilidade da função FMINCON do pacote de otimização instalados no computador.

Foram desenvolvidas duas funções com diferentes funcionalidades:

- **SVMsst.m**: implementa uma *Support Vector Machine* (SVM) capaz de ser treinada a partir de treinamento semi-supervisionado, utilizando-se de transdução;
- **teste_SVMsst.m**: outro que fornece o desempenho da SVMsst, após treinada, para uma dada base de teste, que pertença ao mesmo espaço da base de treinamento previamente fornecida quando da aprendizagem da SVMsst.

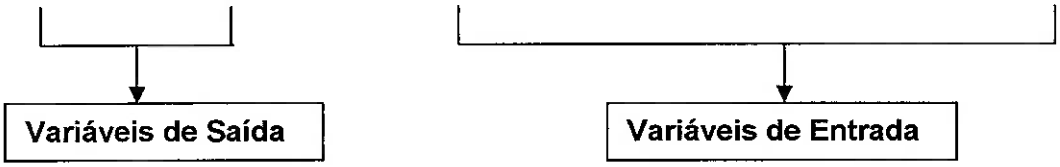
A funcionalidade de cada programa bem como uma explicação das entradas e saídas são fornecidas a seguir.

SVMsst

FUNCIONALIDADE: Esta função MatLab implementa uma *Support Vector Machine* (SVM) capaz de ser treinada a partir de treinamento semi-supervisionado (bases de dados cuja classificação está incompleta), utilizando-se de transdução. Para maiores detalhes sobre a implementação e principais teorias sobre as quais se baseia o presente *software*, por favor, consulte o texto completo do Projeto de Conclusão de Curso intitulado "Classificação Automática de Dados com *Support Vector Machines* (SVMs) a Partir de Bases de Dados Incompletas", disponível na Biblioteca do Departamento de Engenharia Mecatrônica e Sistemas Mecânicos (PMR) da Poli-USP.

Esta função do MatLab pode ser descrità pela seguinte estrutura:

[Wsvm, classes_atribuidas] = **SVMsst** ('nome_base', flag_feature, lambda, num_max_iteracoes)



VARIÁVEIS DE ENTRADA: (*Tipo, nome da variável → descrições*)

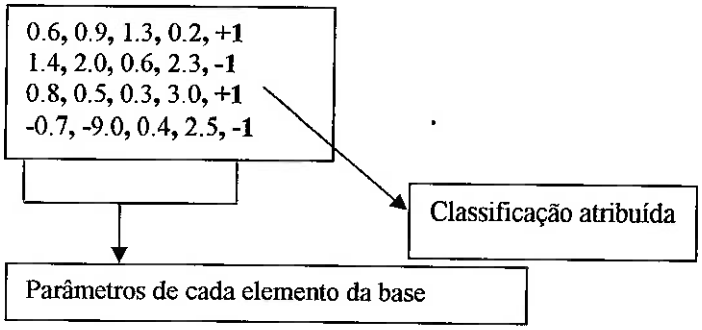
string, nome_base → nome do arquivo que contém a base de dados a ser utilizada (entre aspas simples).

Cada elemento da base se situa em uma linha do arquivo, e os parâmetros da base devem ser separados por vírgulas. O último vetor-coluna da base deve conter a classificação (+1/-1). A base de dados não precisa ser linearmente separável, ou de outra forma, qualquer base classificada de forma binária pode ser utilizada para o treinamento da SVM.

As bases utilizadas para os testes de validação da *SVMsst* (vide o texto do Projeto de Conclusão de Curso), *Ionosphere* e *Sonar*, estão disponíveis no CD que também contém este Manual.

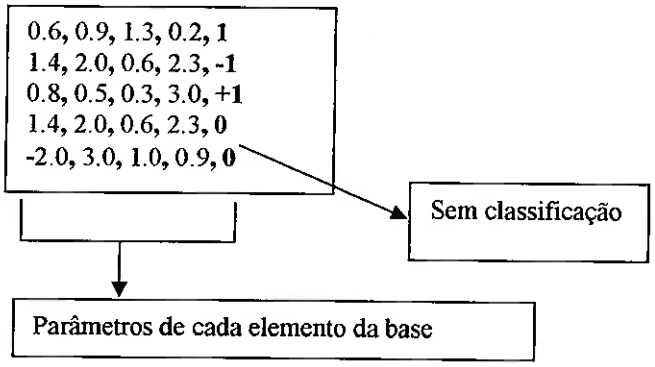
Observação: para serem utilizadas, as bases devem ser formatadas. Por exemplo, a base *Ionosphere* tem seus vetores classificados em **g** (*good*, ou bom) ou **b** (*bad*, ou ruim). Estas classes devem ser remapeadas para **+1** ou **-1**.

Exemplo de configuração de uma base completa (2 elementos, 4 parâmetros, classificação binária):



Para o caso de treinamento semi-supervisionado, para os elementos não classificados, atribuir o valor *zero* em sua classificação. Estes vetores devem estar nas últimas linhas da base de treinamento.

Exemplo de configuração de uma base incompleta para treinamento semi-supervisionado (2 elementos, 4 parâmetros, classificação binária):



int, flag_*feature* → tipo do *feature* a ser utilizado para o mapeamento dos vetores do conjunto de treinamento em algum outro espaço matemático mais favorável ao treinamento da *Support Vector Machine*.

Para utilizar o *feature* originalmente usado na implementação da *SVMsst*, o valor da variável deve ser *flag_feature* = 0. O *feature default* utilizada pode ser fornecido pela seguinte relação:

$$fi(\mathbf{x}) = (x_1.x_1,\ x_1.x_2,...,x_1.x_N,\ x_2.x_1,.....,x_N.x_N),$$

caso contrário, basta o usuário disponibilizar uma função chamada ***feature*** (*feature.m*) no diretório de trabalho utilizado, para que o treinamento da *SVMsst* seja baseado nesse novo *feature*, e informando o valor da variável como *flag_feature* = 1;

int, lambda → parâmetro *lambda* a ser utilizado como parâmetro da SVM com *soft margin*.

O valor *default* a ser informado deve ser *lambda* = 1. Para maiores detalhes sobre este parâmetro, consulte o texto completo do Projeto de Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector Machines (SVMs) a Partir de Bases de Dados Incompletas".

int, num_max_iteracoes → número máximo de iterações permitidos para o algoritmo de otimização.

O valor *default* a ser informado deve ser *num_max_iteracoes* = 10000.

VARIÁVEIS DE SAÍDA: (*Tipo, nome da variável* → *descrições*)

vetor, Wsvm → vetor-coluna $\mathbf{w_{svm}}$ (matriz de pesos) construída pelo treinamento da SVMsst. Este vetor deve ser posteriormente utilizado na função *teste_SVMsst* (apresentada posteriormente) para classificar bases.

vetor, classes_atribuídas → vetor-coluna com as classes atribuídas pela SVM, por transdução, aos vetores da base de dados não-classificados.

Para *re-treinamento após execução de transdução*, basta substituir as classes nulas dos vetores da base de treinamento não-classificados pelas classes fornecidas pela SVMsst.

teste_SVMsst

FUNCIONALIDADE: Esta função MatLab implementa um módulo de testes para a *SVMsst*. A presente função fornece o desempenho da *SVMsst*, após treinada, para uma dada base de teste, que pertença ao mesmo espaço da base de treinamento previamente fornecida quando da aprendizagem da *SVMsst*. Para maiores detalhes sobre a implementação e principais teorias sobre as quais se baseia o presente software, por favor, consulte o texto completo do Projeto de Conclusão de Curso intitulado "Classificação Automática de Dados com *Support Vector Machines* (SVMs) a Partir de Bases de Dados Incompletas", disponível na Biblioteca do Departamento de Engenharia Mecatrônica e Sistemas Mecânicos (PMR) da Poli-USP.

Esta função do MatLab pode ser descrita pela seguinte estrutura:

function [acertos, classificação] = **teste_SVMsst** (Wsvm, nome_base, flag_feature)

VARIÁVEIS DE ENTRADA: (*Tipo, nome da variável* → *descrições*)

vetor, Wsvm → vetor-coluna $\mathbf{w}_{svm}$ (matriz de pesos) construída pelo prévio treinamento da SVMsst (pela função *SVMsst*).

string, nome_base → nome do arquivo que contém a base de dados a ser utilizada (entre aspas simples).

Esta base de dados deve ser apresentada da mesma forma que a citada na função *SVMsst*. As classes informadas na última coluna, servirão para medir o desempenho da *SVMsst* em inferir corretamente as classes dos vetores fornecidos para teste.

Caso o usuário deseje apenas *classificar* os dados da base de teste, sem necessidade de se medir a performance da SVM, pode-se informar qualquer valor na última coluna do arquivo que contém a base de dados.

int, flag_feature → tipo do *feature* a ser utilizado para o mapeamento dos vetores do conjunto de treinamento em algum outro espaço matemático mais favorável ao treinamento da *Support Vector Machine*.

Selecionar o mesmo *feature* utilizado quando da execução da função *SVMsst*.

VARIÁVEIS DE SAÍDA: (*Tipo, nome da variável* → *descrições*)

float, acertos → porcentagem de acertos obtida pela *SVMsst* na classificação dos vetores da base de teste fornecida.

No caso em que o usuário está interessado apenas em classificar uma base de dados, esta saída não terá nenhum uso.

vetor, classificação → vetor com a classificação dada pela *SVMsst* para cada vetor da base de testes.

Caso o usuário esteja apenas classificando uma base de dados, sem levar em consideração a performance da SVM, este vetor será a única saída "útil" da função.

Apêndice B

Listagem dos códigos da SVMsst escritos em MatLab

```
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% ESCOLA POLITECNICA DA UNIVERSIDADE DE SAO PAULO
% DEPARTAMENTO DE ENGENHARIA MECATRONICA E SISTEMAS MECANICOS (PMR)
% Sao Paulo, 01 de dezembro de 2003.
%
% PROJETO DE CONCLUSAO DE CURSO: "Classificação Automática de Dados com Support Vector Machines (SVMs)
%                                 a Partir de Bases de Dados Incompletas"
%
%
% ALUNOS: Clayton Silva Oliveira     ORIENTADOR: Prof. Dr. Fabio Gagliardi Cozman
%         Eduardo Takashi Inowe
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% TITULO DA FUNCAO: SVMsst
%
% AVISO IMPORTANTE: E NECESSARIA A EXISTENCIA DA FUNCAO 'fmincon.m' DO PACOTE DE OTIMIZACAO DO
% MATLAB INSTALADO EM SEU COMPUTADOR PARA A UTILIZACAO DA FUNCAO!
%
%
% FUNCIONALIDADE: Esta funcao MatLab implementa uma Support Vector Machine (SVM) capaz de ser treinada
%                 a partir de treinamento semi-supervisionado (bases de dados cuja classificacao esta
%                 incompleta), utilizando-se de transducao.
%
%                 Para maiores detalhes sobre a implementacao e principais teorias sobre as quais se
%                 baseia o presente software, por favor, consulte o texto completo do Projeto de
%                 Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector
%                 Machines (SVMs) a Partir de Bases de Dados Incompletas", disponivel na Biblioteca
%                 do Departamento de Engenharia Mecatronica e Sistemas Mecanicos (PMR) da Poli-USP.
%
%                 Para maiores detalhes sobre como utilizar a funcao, por favor, consulte o "Manual do
```

```
%                 Usuario - SVMsst", disponivel no texto e no CD anexo ao Projeto de Conclusao de
%                 Curso.
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% DESCRICAO DAS VARIAVEIS DE ENTRADA E SAIDA DA FUNCAO "SVMsst.m"
%
% VARIAVEIS DE ENTRADA: Tipo, nome da variavel --> descricoes
%
% string, nome_base  --> nome do arquivo que contem a base de dados a ser utilizada (entre aspas
%                        simples);
%                    --> o formato em que o arquivo deve estar pode ser encontrado no "Manual do
%                        Usuario - SVMsst": consulte-o principalmente caso a base de dados possua
%                        vetores nao-classificados;
%
% int, flag_feature  --> tipo do feature a ser utilizado para o mapeamento dos vetores do conjunto de
%                        treinamento em algum outro espaco matematico mais favoravel ao treinamento da
%                        Support Vector Machine;
%                    --> para utilizar o feature originalmente usado na implementacao da SVMsst, o
%                        valor da variavel deve ser flag_feature = 0; caso contrario, basta o usuario
%                        disponibilizar uma funcao chamada feature (feature.m) no diretorio de trabalho
%                        utilizado, para que o treinamento da SVMsst seja baseado nesse novo feature,
%                        e informando o valor da variavel como flag_feature = 1;
%                    --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
% int, lambda        --> parametro lambda a ser utilizado como parametro da SVM com soft margin;
%                    --> valor default a ser informado: lambda = 1;
%                    --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
% int, num_max_iteracoes --> numero maximo de iteracoes permitido para o algoritmo de otimizacao.
%                        --> valor default a ser informado: num_max_iteracoes = 10000
%                        --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
```

```
%
%
% VARIAVEIS DE SAIDA: Tipo, nome da variavel --> descricoes
%
% vetor, Wsvm               --> vetor-coluna W (matriz de pesos) construida pelo treinamento da SVM;
%
% vetor, classes_atribuidas --> vetor-coluna com as classes atribuidas pela SVM, por transducao, aos
%                               vetores da base de dados nao-classificados.
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------

function [Wsvm, classes_atribuidas] = SVMsst(nome_base, flag_feature, lambda, num_max_iteracoes)

% Carregando a base de dados.
fprintf(1,'Loading training data...');
sample = dlmread(nome_base,',');
fprintf(1,'Training data loaded!\n');

% Determinacao do tamanho da base de dados.
[m_sample,n] = size(sample);

% Disponibilizacao da variavel para as rotinas de otimizacao do MatLab.
save m_sample m_sample;

% Determinacao do numero de vetores nao-classificados na base de dados.
dados_n_classificados_sample = 0;
for i=1:m_sample
    if sample(i,n) == 0
        dados_n_classificados_sample = dados_n_classificados_sample + 1;
    end
end

% Disponibilizacao da variavel para as rotinas de otimizacao do MatLab.
save dados_n_classificados_sample dados_n_classificados_sample;
```

```
% Determinacao do numero de vetores classificados na base de dados.
dados_classificados_sample = m_sample - dados_n_classificados_sample;

% Matriz "Y" dos resultados da base a ser utilizada para o treinamento.
y_classificados_sample = sample(1:dados_classificados_sample, n);

% Disponibilizacao da variavel para as rotinas de otimizacao do MatLab.
save y_classificados_sample y_classificados_sample;

% Determinacao do feature a ser utilizado para mapeamento dos vetores da base de dados utilizada.
fprintf(1,'Extracting the feature from training data loaded...');
if flag_feature == 0 % Uso do feature default
    feature=feature_default(sample(:,1:(n-1)));
elseif flag_feature == 1 % Uso do feature disponibilizado pelo usuario
    feature=feature(sample(:,1:(n-1)));
end
fprintf(1,'Feature extracted!\n');

% Disponibilizacao da variavel para as rotinas de otimizacao do MatLab.
save feature feature;

% Funcao WolfeDual para encontrar vetor alfa otimo e classes atribuidas ao dados nao classificados.
if dados_n_classificados_sample == 0
    fprintf(1,'Training SVMsst by Supervised Training Only...');
else
    fprintf(1,'Training SVMsst by Semi-Supervised Training...');
end
x=WolfeDual(feature, lambda, dados_n_classificados_sample, num_max_iteracoes);
alpha=x(1:m_sample);


% Montando a matriz Y ja com as classes dos vetores nao-classificados atribuidas pela por transducao.
for i=1:dados_classificados_sample
    Y(i,i) = y_classificados_sample(i);
end
indice = 0;
```

```
for i=(m_sample+1):(m_sample+dados_n_classificados_sample)
    Y(i,i) = menosum_ou_um(x(i));
    indice = indice+1;
    classes_atribuidas(indice,1) = Y(i,i);
end

% Determinacao de Wsvm
Wsvm=0;
for i=1:m_sample
    Wsvm=Wsvm+alpha(i)*Y(i,i)*feature(i,:)';
end

% ------------------------------------ fim da funcao "SVMsst.m" ------------------------------------
% --------------------------------------------------------------------------------------------------


% --------------------------------------------------------------------------------------------------
% --------------------------------------------------------------------------------------------------
% TITULO DA FUNCAO: feature_default
%
%
% FUNCIONALIDADE: Esta funcao MatLab implementa o feature default, cuja forma e a seguinte:
%
%                 fi(x) = (x1.x1, x1.x2,...,x1.xN, x2.x1,.....,xN.xN)
%
%                 Para maiores detalhes sobre a implementacao e principais teorias sobre as quais se
%                 baseia o presente software, por favor, consulte o texto completo do Projeto de
%                 Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector
%                 Machines (SVMs) a Partir de Bases de Dados Incompletas", disponivel na Biblioteca
%                 do Departamento de Engenharia Mecatronica e Sistemas Mecanicos (PMR) da Poli-USP.
%
% --------------------------------------------------------------------------------------------------
% --------------------------------------------------------------------------------------------------


% --------------------------------------------------------------------------------------------------
```

```
% ------------------------------------------------------------------------------------------------
% DESCRICAO DAS VARIAVEIS DE ENTRADA E SAIDA DA FUNCAO feature_default
%
% VARIAVEIS DE ENTRADA: Tipo, nome da variavel --> descricoes
%
% vetor, sample  --> matriz contendo os vetores da base de dados a serem mapeados de acordo com o
%                    feature selecionado;
%                --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
%
% VARIAVEIS DE SAIDA: Tipo, nome da variavel --> descricoes
%
% vetor, feature  --> vetor contendo os vetores da base de dados mapeados de acordo com o
%                     feature selecionado;
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------

function [feature]=feature_default(sample)

[m,n]=size(sample);
contador=1;

for k=1:m,
    for i=1:n,
        for j=1:n,
            feature(k,contador)=sample(k,i)*sample(k,j)';
            contador=contador+1;
        end
    end
    contador=1;
end

% ---------------------------------- fim da funcao feature_default ---------------------------------
% ------------------------------------------------------------------------------------------------
```

```
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% TITULO DA FUNCAO: WolfeDual
%
% AVISO IMPORTANTE: E NECESSARIA A EXISTENCIA DA FUNCAO 'fmincon.m' DO PACOTE DE OTIMIZACAO DO
% MATLAB INSTALADO EM SEU COMPUTADOR PARA A UTILIZACAO DA FUNCAO!
%
%
% FUNCIONALIDADE: Esta funcao MatLab implementa o problema de otimizacao Wolfe Dual;
%
%                 Para maiores detalhes sobre a implementacao e principais teorias sobre as quais se
%                 baseia o presente software, por favor, consulte o texto completo do Projeto de
%                 Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector
%                 Machines (SVMs) a Partir de Bases de Dados Incompletas", disponivel na Biblioteca
%                 do Departamento de Engenharia Mecatronica e Sistemas Mecanicos (PMR) da Poli-USP.
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% DESCRICAO DAS VARIAVEIS DE ENTRADA E SAIDA DA FUNCAO WolfeDual
%
% VARIAVEIS DE ENTRADA: Tipo, nome da variavel --> descricoes
%
% vetor, feature  --> matriz contendo os vetores da base de dados mapeados de acordo com o feature
%                     selecionado;
%                 --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
% int, lambda     --> parametro lambda a ser utilizado como parametro da SVM com soft margin;
%                 --> valor default: lambda = 1;
%                 --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
%
```

```
% VARIAVEIS DE SAIDA: Tipo, nome da variavel --> descricoes
%
% vetor, vetor_otimo  --> vetor contendo os alfas otimos e as classes atribuidas, por transducao, pela
%                         SVM;
%
% int, dados_n_classificados_sample --> numero de dados nao-classificados na base de dados;
%
% int, num_max_iteracoes --> numero maximo de iteracoes permitido para o algoritmo de otimizacao.
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------

function [vetor_otimo] = WolfeDual(feature, lambda, dados_n_classificados_sample, num_max_iteracoes)

[m,n]=size(feature);

% Matriz Gram - G.
for i=1:m
    for j=1:m
        G(i,j)=feature(i,:)*feature(j,:)';
    end
end

% Matriz Gram modificada (Quadratic Approximation) para soft margin.
I=eye(m);
G=G+lambda*m*I;

% Disponibilizacao da variavel para as rotinas de otimizacao do MatLab.
save Gram_Matrix G;

% Matriz de restricoes - Alfa >= 0
restricao=0;
for i=1:m
    restricao(i,i)=-1;
end
for i=(m+1):(m + dados_n_classificados_sample)
```

```
    restricao(i,i)=0;
end
for i=1:(m + dados_n_classificados_sample)
    escalar(i)=0;
end

% "Chute" inicial para o algoritmo de otimizacao.
x0(1:m) = 1;
x0(m+1:m+dados_n_classificados_sample) = rand(1,dados_n_classificados_sample)/10;

% Otimizacao da funcao Wolfe Dual com as restricoes impostas
options=optimset('largescale', 'off', 'MaxFunEvals', num_max_iteracoes);
if dados_n_classificados_sample ~= 0
    [vetor_otimo]  = fmincon(@myfun_SVMsst,x0,restricao,escalar,[],[],[],[],@nonlcon_SVMsst,options);
else
    [vetor_otimo]  = fmincon(@myfun_SVMsst,x0,restricao,escalar,[],[],[],[],[],options);
end


% ---------------------------------- fim da funcao WolfeDual ----------------------------------------
% ---------------------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------------------
% ---------------------------------------------------------------------------------------------------
% TITULO DA FUNCAO: myfun_SVMsst
%
%
% FUNCIONALIDADE: funcao objetivo a ser minimizada para se encontrar os valores otimos dos alfas na
%                 funcao Wolfe Dual;
%
%                 Para maiores detalhes sobre a implementacao e principais teorias sobre as quais se
%                 baseia o presente software, por favor, consulte o texto completo do Projeto de
%                 Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector
%                 Machines (SVMs) a Partir de Bases de Dados Incompletas", disponivel na Biblioteca
%                 do Departamento de Engenharia Mecatronica e Sistemas Mecanicos (PMR) da Poli-USP.
```

```
%
% ------------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------------

function f = myfun_SVMsst(x)

load y_classificados_sample y_classificados_sample;
y = y_classificados_sample;
L = size(y);

load dados_n_classificados_sample dados_n_classificados_sample;
dados_n_classificados = dados_n_classificados_sample;

load Gram_Matrix G;
load m_sample m_sample;
m = m_sample;

A = 0;
for i=1:m
    A = A + x(i);
end

B_um = 0;
for a = 1:L
    for i = 1:L
        B_um = B_um + x(a)*y(a)*x(i)*y(i)*G(i,a);
    end
end

B_dois = 0;
for a = 1:L
    for j = (L+1):m
        B_dois = B_dois + x(a)*y(a)*x(j)*x(dados_n_classificados + j)*G(j,a);
    end
end
```

```
B_tres = 0;
for b=(L+1):m
    for i=1:L
        B_tres = B_tres + x(b)*x(dados_n_classificados + b)*x(i)*y(i)*G(i,b);
    end
end

B_quatro = 0;
for b=(L+1):m
    for j=(L+1):m
        B_quatro = B_quatro + x(b)*x(dados_n_classificados + b)*x(j)*...
        x(dados_n_classificados + j)*G(j,b);
    end
end


B = 0.5*(B_um + B_dois + B_tres + B_quatro);


f = 1/(A - B);

% ---------------------------------- fim da funcao myfun_SVMsst ------------------------------------
% --------------------------------------------------------------------------------------------------


% --------------------------------------------------------------------------------------------------
% --------------------------------------------------------------------------------------------------
% TITULO DA FUNCAO: nonlcon_SVMsst
%
%
% FUNCIONALIDADE: restricoes nao-lineares da funcao objetivo a ser minimizada;
%
%                 Para maiores detalhes sobre a implementacao e principais teorias sobre as quais se
%                 baseia o presente software, por favor, consulte o texto completo do Projeto de
%                 Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector
%                 Machines (SVMs) a Partir de Bases de Dados Incompletas", disponivel na Biblioteca
```

```
%                do Departamento de Engenharia Mecatronica e Sistemas Mecanicos (PMR) da Poli-USP.
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------

function [C, Ceq] = nonlcon_SVMsst(x)

C = [ ];

load dados_n_classificados_sample dados_n_classificados_sample;
dados_n_classificados = dados_n_classificados_sample;

load m_sample m_sample;
m = m_sample;

load y_classificados_sample y_classificados_sample;
classes_classificados = y_classificados_sample;

load feature feature;
[m, dim_feature] = size(feature);

Wsvm_inst(1,1:dim_feature) = 0;
for i=1:(m-dados_n_classificados) % vetores classificados
    Wsvm_inst = Wsvm_inst + x(i)*classes_classificados(i)*feature(i,:);
end
i_n_class = 0;
for i = (m-dados_n_classificados+1):m % vetores nao-classificados
    i_n_class = i + dados_n_classificados;
    Wsvm_inst = Wsvm_inst + x(i)*x(i_n_class)*feature(i,:);
end

i_n_class = 0;
cont = 0;
for i=(m-dados_n_classificados+1):m
    i_n_class = i + dados_n_classificados;
```

```
    cont = cont + 1;
    Ceq(cont) = menosum_ou_um(Wsvm_inst*feature(i,:)') - x(i_n_class);
end

% ---------------------------------- fim da funcao nonlcon_SVMsst ------------------------------------
% ----------------------------------------------------------------------------------------------------
```

```
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% ESCOLA POLITECNICA DA UNIVERSIDADE DE SAO PAULO
% DEPARTAMENTO DE ENGENHARIA MECATRONICA E SISTEMAS MECANICOS (PMR)
% Sao Paulo, 01 de dezembro de 2003.
%
% PROJETO DE CONCLUSAO DE CURSO: "Classificação Automática de Dados com Support Vector Machines (SVMs)
%                                 a Partir de Bases de Dados Incompletas"
%
%
% ALUNOS: Clayton Silva Oliveira      ORIENTADOR: Prof. Dr. Fabio Gagliardi Cozman
%         Eduardo Takashi Inowe
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% TITULO DA FUNCAO: teste_SVMsst
%
%
%
% FUNCIONALIDADE: Esta funcao MatLab implementa um modulo de testes para a SVMsst.
%                 A presente funcao fornece o desempenho da SVMsst, apos treinada, para uma dada base
%                 de teste, que pertenca ao mesmo espaco da base de treinamento previamente fornecida
%                 quando da aprendizagem da SVMsst.
%
%                 Para maiores detalhes sobre a implementacao e principais teorias sobre as quais se
%                 baseia o presente software, por favor, consulte o texto completo do Projeto de
%                 Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector
%                 Machines (SVMs) a Partir de Bases de Dados Incompletas", disponivel na Biblioteca
%                 do Departamento de Engenharia Mecatronica e Sistemas Mecanicos (PMR) da Poli-USP.
%
%                 Para maiores detalhes sobre como utilizar a funcao, por favor, consulte o "Manual do
%                 Usuario - SVMsst", disponivel no texto e no CD anexo ao Projeto de Conclusao de
```

```
%                 Curso.
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% DESCRICAO DAS VARIAVEIS DE ENTRADA E SAIDA DA FUNCAO "teste_SVMsst.m"
%
% VARIAVEIS DE ENTRADA: Tipo, nome da variavel --> descricoes
%
% vetor, Wsvm        --> vetor-coluna W (vetor de pesos) construida pelo previo treinamento da SVMsst;
%
% string, nome_base  --> nome do arquivo que contem a base de dados a ser utilizada (entre aspas
%                        simples);
%                    --> o formato em que o arquivo deve estar pode ser encontrado no "Manual do
%                        Usuario - SVMsst": consulte-o principalmente caso a base de dados possua
%                        vetores nao-classificados;
%
% int, flag_feature  --> tipo do feature a ser utilizado para o mapeamento dos vetores do conjunto de
%                        treinamento em algum outro espaco matematico mais favoravel ao treinamento da
%                        Support Vector Machine;
%                    --> para utilizar o feature originalmente usado na implementacao da SVMsst, o
%                        valor da variavel deve ser flag_feature = 0; caso contrario, basta o usuario
%                        disponibilizar uma funcao chamada feature (feature.m) no diretorio de trabalho
%                        utilizado, para que o treinamento da SVMsst seja baseado nesse novo feature,
%                        e informando o valor da variavel como flag_feature = 1;
%                    --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
%
% VARIAVEIS DE SAIDA: Tipo, nome da variavel --> descricoes
%
%
% float, acertos         --> porcentagem de acertos obtida pela SVMsst na classificação dos vetores da base
%                            de teste fornecida;
```

```
%                       --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
% vetor, classificacao  --> vetor com a classificacao dada pela SVMsst para cada vetor da base de
%                           testes.
%                       --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
% ------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------
function [acertos, classificacao] = teste_SVMsst(Wsvm, nome_base, flag_feature)

% Carregando a base de dados.
fprintf(1,'Loading test data...');
sample = dlmread(nome_base,',');
fprintf(1,'Test data loaded!\n');
[m,n] = size(sample);

% Determinacao do feature a ser utilizado para mapeamento dos vetores da base de dados utilizada.
fprintf(1,'Extracting the feature from the test data loaded...');
if flag_feature == 0 % Uso do feature default
    feature=feature_default(sample(:,1:(n-1)));
elseif flag_feature == 1 % Uso do feature disponibilizado pelo usuario
    feature=feature(sample(:,1:(n-1)));
end
fprintf(1,'Feature extracted!\n');

% Classificacao da base de testes fornecida
fprintf(1,'Classifying the test data...');
classificacao(1:m,1) = 0;
acertos = 0;
for i=1:m
    classificacao(i,1) = menosum_ou_um(feature(i,:)*Wsvm);
    if classificacao(i,1) == sample(i,n)
        acertos = acertos + 1;
    end
end
```

```
acertos = acertos/m;
fprintf(1,'All the classifications successfully done!');

% ---------------------------------- fim da funcao teste_SVMsst ----------------------------------
% ------------------------------------------------------------------------------------------------

% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% TITULO DA FUNCAO: feature_default
%
%
% FUNCIONALIDADE: Esta funcao MatLab implementa o feature default, cuja forma e a seguinte:
%
%                 fi(x) = (x1.x1, x1.x2,...,x1.xN, x2.x1,.....,xN.xN)
%
%                 Para maiores detalhes sobre a implementacao e principais teorias sobre as quais se
%                 baseia o presente software, por favor, consulte o texto completo do Projeto de
%                 Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector
%                 Machines (SVMs) a Partir de Bases de Dados Incompletas", disponivel na Biblioteca
%                 do Departamento de Engenharia Mecatronica e Sistemas Mecanicos (PMR) da Poli-USP.
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------

% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% DESCRICAO DAS VARIAVEIS DE ENTRADA E SAIDA DA FUNCAO feature_default
%
% VARIAVEIS DE ENTRADA: Tipo, nome da variavel --> descricoes
%
% vetor, sample  --> matriz contendo os vetores da base de dados a serem mapeados de acordo com o
%                    feature selecionado;
%                --> para maiores detalhes, consulte o "Manual do Usuario - SVMsst".
%
```

```
%
% VARIAVEIS DE SAIDA: Tipo, nome da variavel --> descricoes
%
% vetor, feature  --> vetor contendo os vetores da base de dados mapeados de acordo com o
%                     feature selecionado;
%
% ------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------

function [feature]=feature_default(sample)

[m,n]=size(sample);
contador=1;

for k=1:m,
    for i=1:n,
        for j=1:n,
            feature(k,contador)=sample(k,i)*sample(k,j)';
            contador=contador+1;
        end
    end
    contador=1;
end

% ---------------------------------- fim da funcao feature_default ----------------------------------
% ------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------
% TITULO DA FUNCAO: menosum_ou_um
%
%
%
% FUNCIONALIDADE: Classifica um numero 'x' em +1 no caso em que x >= 1, em -1 caso x <= -1, ou em 0
%                 caso contrario.
```

```
%
%               Para maiores detalhes sobre a implementacao e principais teorias sobre as quais se
%               baseia o presente software, por favor, consulte o texto completo do Projeto de
%               Conclusao de Curso intitulado "Classificação Automática de Dados com Support Vector
%               Machines (SVMs) a Partir de Bases de Dados Incompletas", disponivel na Biblioteca
%               do Departamento de Engenharia Mecatronica e Sistemas Mecanicos (PMR) da Poli-USP.
%
%               Para maiores detalhes sobre como utilizar a funcao, por favor, consulte o "Manual do
%               Usuario - SVMsst", disponivel no texto e no CD anexo ao Projeto de Conclusao de
%               Curso.
%
% ------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------


function resp_menosum_ou_um = menosum_ou_um(x)

if x >= 0.9999
    resp_menosum_ou_um = 1;
elseif x <= -0.9999
    resp_menosum_ou_um = -1;
else
    resp_menosum_ou_um = 0;
end

% ---------------------------------- fim da funcao menosum_ou_um ---------------------------------
% ------------------------------------------------------------------------------------------------
```