Universidade de São Paulo

Instituto de Matemática e Estatística

MAC0499 Trabalho de Formatura Supervisionado

Máquinas de suporte vetorial e sua aplicação na detecção de spam

Iniciação Científica

Aluno: Antonio Carlos dos Santos Nº USP: 3534575
Orientador: Paulo José da Silva e Silva DCC - IME - USP

Resumo

Neste trabalho foram estudados conceitos envolvendo máquinas de suporte vetorial e sua aplicação em uma situação real: a detecção de spam. Dado um conjunto de pontos que representam objetos nos quais estamos interessados, com rótulos associados indicando suas classes, as máquinas de suporte vetorial procuram encontrar um hiperplano separador que os classifique corretamente, deixando todos os dados de uma mesma classe de um lado e os da outra classe do outro lado do hiperplano. Tais conceitos podem ser estendidos para a classificação em várias classes.

Na primeira parte da monografia, procurou-se apresentar o trabalho feito durante a iniciação científica, explicando as idéias principais estudadas. Na segunda parte, é apresentada a experiência pessoal obtida durante a iniciação, sua relação com o bacharelado em Ciência da Computação do IME - USP e quais as novas direções de pesquisa.

Palavras-chaves: Máquinas de suporte vetorial, aprendizado estatístico, função kernel, teorema de Mercer.

Sumário

Parte I

Iniciação científica

Capítulo 1

Introdução

Com o uso cada vez maior da Internet, a troca de mensagens eletrônicas, os *e-mails*, tornou-se uma ação muito freqüente entre seus usuários, tanto para assuntos profissionais quanto para pessoais. Atualmente, segundo o sítio Spam-Filter-Review [Sfr05], aproximadamente 31 bilhões de mensagens eletrônicas foram enviadas pela Internet diariamente em todo o mundo durante 2004.

Entretanto, o número de mensagens indesejadas recebidas, os **spams** (***S**tupid **P**ointless **A**nnoying **M**essages*), também é muito grande. Apenas o CERT (Centro de Estudos, Resposta e Tratamento de Incidentes no Brasil) [Cer05], de janeiro até o final de novembro de 2005, foi notificado sobre mais de dois milhões de spams. E o sítio Spam-Filter-Review [Sfr05] calculou em aproximadamente 12,4 bilhões a quantidade de spams enviados diariamente em 2004. Esta situação tem gerado vários problemas, tanto para empresas provedoras de acesso à Internet, que têm uma carga maior de uso de seus servidores, quanto para os usuários, que gastam muito tempo lendo spams para depois descartá-los, e podem ainda serem prejudicados por programas maliciosos, como vírus e *spywares*. Segundo a empresa americana de consultoria Ferris Research [Fer05], as perdas mundiais resultantes de problemas com spams chegarão a US$50 bilhões em 2005, principalmente devido à queda de produtividade dos funcionários.

Assim, para evitar tais problemas, atualmente tem-se pesquisado bastante novas formas de detectar e bloquear mensagens consideradas spams automaticamente. Entre elas, está o uso de máquinas de suporte vetorial.

Máquinas de suporte vetorial, em inglês *Support Vector Machines* (SVMs), representam um conceito novo na área de sistemas de aprendizado compu-

tacional. São baseadas na teoria de aprendizado estatístico, desenvolvida principalmente por Vladimir Vapnik[Vap98], tendo como uma de suas principais idéias o mapeamento dos dados de entrada para um outro espaço onde haja um hiperplano que os separe linearmente.

SVMs têm apresentado um bom desempenho em algumas aplicações como, por exemplo, reconhecimento de imagens e classificação de seqüências de DNA, e ainda têm um grande potencial para serem aplicadas na solução de outros problemas. Além disso, SVMs têm apresentado algumas vantagens sobre outros sistemas de aprendizado computacional. Um exemplo é o fato de não apresentarem mínimos locais[1], situação que acontece com redes neurais.

1.1 Objetivos do trabalho

Este trabalho tem como objetivo sistematizar o estudo feito durante iniciação científica entre março e dezembro de 2005, sobre máquinas de suporte vetorial e sua aplicação na detecção de spam, sob a orientação do professor Paulo José da Silva e Silva (IME - USP).

A iniciação teve como base os artigos *Support Vector Machines for Spam Categorization* [HWV99], *A Tutorial on Support Vector Machines for Pattern Recognition* [Bur98] e o livro *An Introduction to Support Vector Machines and other kernel-based methods* [CST02]. Além disso, para estudar os conceitos de programação não-linear existentes na teoria de SVMs, foram estudados os livros-texto [Fri05], [NW99] e [Ber95]. Para a compreensão da teoria de aprendizado computacional, as referências principais foram [AB92] e [DHS01]. Durante o texto, serão citadas as demais referências adotadas durante a iniciação científica.

Inicialmente, apresentaremos os conceitos básicos de álgebra linear e a notação que serão utilizados no texto. Em seguida, apresentaremos os conceitos principais da teoria de aprendizado computacional. Finalmente, apresentaremos a teoria de SVMs, sua aplicação na detecção de spam e os resultados e conclusões obtidos.

[1] mínimos de funções que não são globais.

Capítulo 2

Notações e conceitos básicos

Neste capítulo apresentaremos alguns conceitos básicos usados durante o texto e também a notação adotada para representá-los. Para maiores detalhes sobre estes conceitos, sugerimos os livros de Cristianini e Shawe-Taylor [CST02] e Watkins [Wat91].

2.1 Vetores e matrizes

Nós utilizaremos letras minúsculas em negrito para indicar vetores. Durante todo o texto, trataremos que os vetores estão no universo dos reais, mencionando explicitamente quando tal fato não ocorrer. Assim, por exemplo, escreveremos um vetor no $\mathbb{R}^n$ como $\boldsymbol{v}$. Cada componente de um vetor será indicada pela mesma letra usada para representar o vetor, porém em fonte normal, seguida pelo índice da coordenada subescrito. O *vetor nulo* será denotado por $\mathbf{0}$ e os vetor cujas componentes sejam todas 1 será denotado por $\mathbf{1}$.

Uma matriz também será definida sobre o universo dos reais e será denotada por letras maiúsculas em negrito. Diremos que uma matriz tem dimensão $m \times n$ se ela possui m linhas e n colunas. Assim, por exemplo, uma matriz pertencente a $\mathbb{R}^{m \times n}$ seria denotada por $\boldsymbol{A}$. Cada componente de uma matriz será indicada pela letra minúscula correspondente à letra utilizada para representar a matriz, em fonte normal, seguida dos indíces da linha e da coluna subescritos. Cada coluna de uma matriz será indicada pela letra minúscula correspondente à letra usada para representar a matriz, em negrito, seguida pelo índice da coluna subescrito. Opcionalmente, usa-

remos letras maiúsculas em fonte normal, seguidas por índices para indicar a dimensão da matriz, como por exemplo, $(M_{ij})_{i,j=1}^{n}$, para representar uma matriz de dimensão $n \times n$.

Usaremos $\boldsymbol{I}$ para denotar a *matriz identidade*. A *matriz inversa* de uma matriz $\boldsymbol{A}$ é uma matriz $\boldsymbol{B}$ de mesma dimensão que $\boldsymbol{A}$ tal que $\boldsymbol{AB} = \boldsymbol{BA} = \boldsymbol{I}$. A *matriz transposta* de uma matriz $\boldsymbol{A}$ de dimensão $m \times n$ é uma matiz $\boldsymbol{B}$ de dimensão $n \times m$ tal que $a_{ij} = b_{ji}$. Denotaremos a matriz transposta de uma matriz $\boldsymbol{A}$ por $\boldsymbol{A}'$.

2.2 Conceitos básicos de álgebra linear

Um conjunto V é um *espaço vetorial* se as operações de adição e de multiplicação por escalar estão definidas e fechadas nele, tal que para todo $\boldsymbol{x}$, $\boldsymbol{y}$ $\in V$ e $\lambda \in \mathbb{R}$:

$$\begin{aligned} \boldsymbol{x} + \boldsymbol{y} &\in V, && (2.1)\\ \lambda \boldsymbol{x} &\in V, && (2.2)\\ 1\boldsymbol{x} &= \boldsymbol{x}, && (2.3)\\ 0\boldsymbol{x} &= \boldsymbol{0}. && (2.4) \end{aligned}$$

Um *subespaço vetorial* de V será qualquer subconjunto S de V tal que S seja também um subespaço. Uma *combinação linear* de vetores $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ num espaço vetorial V será uma soma do formato $\alpha_1 \boldsymbol{x_1} + \cdots + \alpha_n \boldsymbol{x_n}$, em que $\alpha_i \in \mathbb{R},\ i = 1, \cdots, n$.

Diremos que um conjunto de vetores $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ é *linearmente dependente* se existirem números reais $\alpha_1, \cdots, \alpha_n$ não todos nulos tais que $\alpha_1 \boldsymbol{x_1} + \cdots + \alpha_n \boldsymbol{x_n} = \boldsymbol{0}$. Se todos os α_i somente forem iguais a 0, então diremos que o conjunto é *linearmente independente*. Um conjunto $S = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ é uma *base* de um espaço vetorial V se for linearmente independente e se todo $v \in V$ pude ser expresso de forma única como uma combinação linear de elementos de S.

Um *espaço vetorial V com norma definida* possui uma função norma, representada por $\|\boldsymbol{.}\|$, mapeando cada elemento de V para um número real e satisfazendo as propriedades abaixo:

1. $\|\boldsymbol{v}\| \geq 0$, para todo $v \in V$, e $\|\boldsymbol{v}\| = 0$ se e somente se $v = \boldsymbol{0}$,
2. $\|\boldsymbol{v} + \boldsymbol{u}\| \leq \|\boldsymbol{v}\| + \|\boldsymbol{u}\|$, para todo $v, u \in V$,

3. $\|\alpha \boldsymbol{v}\| = |\alpha| \|\boldsymbol{v}\|$, para todo α real e $v \in V$.

Se $V = \mathbb{R}^n$, a *norma euclidiana* será definida por:

$$\|\boldsymbol{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}. \tag{2.5}$$

Um *espaço vetorial V com produto interno definido* possui um mapeamento $\langle \boldsymbol{.}, \boldsymbol{.} \rangle$ de V para $\mathbb{R}$ tal que todo $\boldsymbol{v}, \boldsymbol{u} \in V$:

1. $\langle \boldsymbol{v}, \boldsymbol{u} \rangle = \langle \boldsymbol{u}, \boldsymbol{v} \rangle$,
2. $\langle \boldsymbol{v}, \boldsymbol{v} \rangle \geq 0$ e $\langle \boldsymbol{v}, \boldsymbol{v} \rangle = 0 \Leftrightarrow v = \boldsymbol{0}$.

Se $V = \mathbb{R}^n$, o *produto interno euclidiano* será definido por:

$$\langle \boldsymbol{v}, \boldsymbol{u} \rangle = v_1 u_1 + v_2 u_2 + \cdots + v_n u_n. \tag{2.6}$$

Assim, para um espaço vetorial V com norma e produto interno euclidianos, temos:

$$\|\boldsymbol{v}\| = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle}, \text{para todo } v \in V. \tag{2.7}$$

Durante o restante do texto, caso não explicitemos qual o produto interno e norma utilizados, então estaremos usando o produto interno e norma euclidianos.

2.3 Convexidade

Um conjunto $S \subseteq \mathbb{R}^n$ será chamada de *convexo* se e somente se para todo $x, y \in S$ e $\lambda \in [0, 1]$, tivermos:

$$\lambda x + (1 - \lambda) y \in S. \tag{2.8}$$

Uma função f definida em um convexo S será *convexa* se e somente para todo $x, y \in S$ e $\lambda \in [0, 1]$, tivermos:

$$f(\lambda x + (1 - \lambda) y) \leq \lambda f(x) + (1 - \lambda) f(y). \tag{2.9}$$

Capítulo 3

Aprendizado Computacional

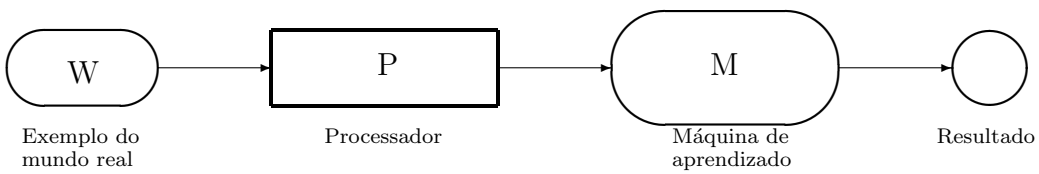
Seja W um conjunto representando o "mundo real", ou seja, um conjunto de objetos nos quais estaremos interessados e chamaremos de *exemplos* ou *dados*.

Seja P um processador que codifica os objetos do mundo real em mensagens. As mensagens são então lidas e analisadas por uma máquina M, cujo propósito é classificar os exemplos. Ou seja, associar rótulos a eles. O tipo de classificação dos exemplos (quantidade de classes e quais os elementos de cada classe) dependerá do estado atual da máquina e da natureza do problema estudado.

M será chamada de *máquina de aprendizado* e o objetivo do processo de aprendizado será fazer mudanças no estado da máquina a partir dos exemplos apresentados para que ela faça uma classificação desejada dos mesmos.

Se tivermos apenas duas classes, chamaremos o processo de classificação de *reconhecimento*. Por exemplo, nós poderíamos ter uma máquina encarregada de reconhecer a letra A digitada em um texto, reconhecendo (classificando) cada símbolo do texto como "A" ou não.

O diagrama abaixo ilustra o modelo de aprendizado apresentado:



3.1 Conceito e hipótese

Seja $\sum$ um conjunto, o qual chamaremos de *alfabeto*, que descreve os exemplos. Durante o restante do texto, consideraremos $\sum$ como $\{0, 1\}$ ou como $\mathbb{R}$. Entretanto, $\sum$ pode ser qualquer outro conjunto, como, por exemplo, formado por símbolos ou palavras. Seja $\sum^n$ o conjunto de n-tuplas de $\sum$ e $\sum^*$ o conjunto de todas cadeias finitas de elementos de $\sum$.

Seja X um subconjunto de $\sum^*$ e R o conjunto de todos os rótulos possíveis atribuídos aos elementos de X. Assim, cada rótulo representará uma classe na qual estamos interessados em separar nossos exemplos. Um *conceito* c será uma função

$$c : X \to R \tag{3.1}$$

que associa um rótulo de R a cada elemento de X. Para a classificação binária, teremos $R = \{-1, 1\}$ e os elementos x de X para os quais $c(x) = 1$ serão chamados de *positivos* e os elementos para os quais $c(x) = -1$ serão chamados de *negativos*.

Assim, o objetivo de uma máquina de aprendizado será aprender a função c (conceito) para o problema estudado, a qual irá mapear corretamente o exemplo para o seu rótulo (classe). A máquina de aprendizado determina um conjunto de funções que associam os exemplos a rótulos, havendo uma função associada a cada estado da máquina. Cada função da máquina de aprendizado será chamada de *hipótese* e o conjunto de todas as hipóteses da máquina de aprendizado será o *espaço de hipóteses*.

O processo de aprendizagem computacional objetiva escolher uma hipótese no espaço de hipóteses de M que melhor representa o conceito desejado c. Entretanto, em situações reais, escolher uma hipótese h que represente corretamente um conceito c pode gastar muitos recursos e tempo. Assim, em situações reais, procura-se escolher uma hipótese h que seja uma boa aproximação do conceito c.

3.2 Treinamento e teste

Geralmente, o processo de aprendizado computacional será feito através da entrada de vários pares de exemplos associados com respectivos rótulos corretos, de forma semelhante a uma criança aprende a escrever. Desta forma, faremos o *treinamento* da máquina, esperando que ela "aprenda" o mapeamento correto (o conceito) dos dados a partir dos exemplos dados.

Um *dado de treinamento* será formado por um exemplo de treinamento associado com o seu rótulo correto. Assim, será um elemento de $(X \times R)$. Uma amostra de treinamento s de tamanho m será uma seqüência de m dados de treinamento, ou seja, um seqüência em $(X \times R)^m$.

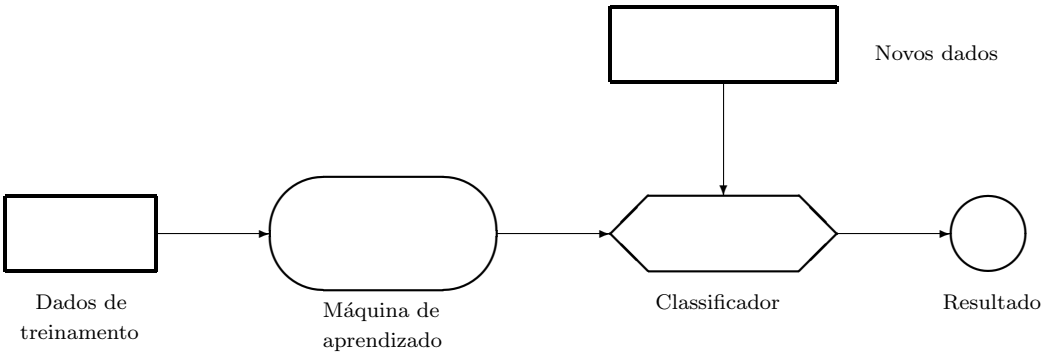
Podemos imaginar que os dados de treinamento são fornecidos por um professor, que sabe qual a classe dos exemplos apresentados por ele. Além disso, iremos supor que todos os exemplos têm uma distribuição de probabilidade desconhecida e são independentemente e identicamente distribuídos.

Um algoritmo de aprendizado será encarregado de ler uma amostra e produzir uma hipótese h para um conceito c a partir desta amostra. Tal hipótese será chamada de *classificador*. Podemos imaginar este algoritmo como uma função L que mapeia uma amostra s para uma hipótese h:

$$h = L(s). \tag{3.2}$$

Uma hipótese h será *consistente* com uma amostra s se o rótulo atribuído pela hipótese a cada dado de treinamento for igual ao rótulo associado a cada exemplo de treinamento. Ou seja, se para cada dado de treinamento (x, y), tivermos $h(x) = y$.

Além disso, a hipótese escolhida pelo algoritmo de aprendizado deve ser capaz de classificar corretamente novos dados que sejam apresentados à máquina depois da fase de treinamento. Tais dados serão chamados de *dados de teste*. A capacidade de classificar corretamente novos dados será chamada de *generalização*. A figura abaixo representa o processo de aprendizado por exemplos:



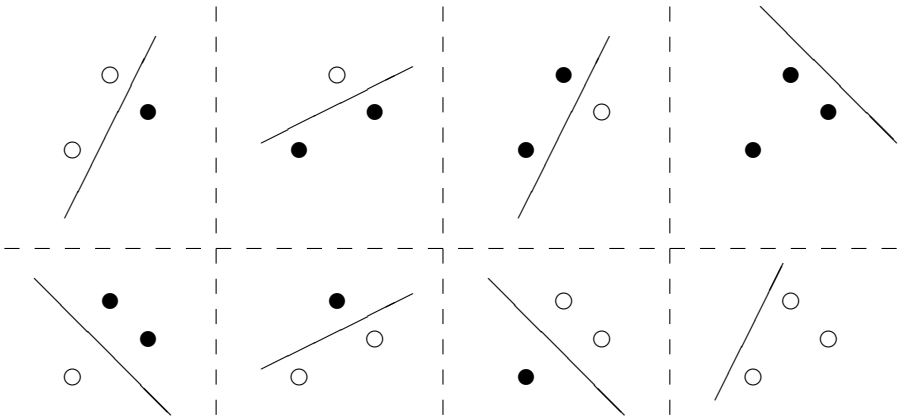
A maioria dos algoritmos de aprendizado procurará obter hipóteses que tenham um equilíbrio entre a consistência e a generalização.

3.3 Dimensão de Vapnik Chervonenkis (VC)

A dimensão de Vapnik Chervonenkis (VC) é uma propriedade de classes de classificadores (espaços de hipóteses) e está relacionada à capacidade de uma máquina de aprendizado classificar qualquer conjunto de treinamento sem erro. Ou seja, a capacidade da máquina aprender corretamente cada dado de treinamento.

Sejam l exemplos de treinamento que podem pertencer a n classes distintas e M uma máquina de aprendizado. Também suponhamos que os l exemplos de treinamento são distintos dois a dois (sem repetições). Assim, temos n^l classificações (rotulações) distintas possíveis para o nosso conjunto de treinamento. Se para uma destas rotulações, há uma hipótese produzida por M que classifica os dados corretamente, então diremos que estes dados são *dominados* por M, ou seja, dominados pela classe de classificadores gerados por M. Nós definiremos a *dimensão de Vapnik Chervonenkis (dimVC)* desta classe de classificadores (da sua máquina de aprendizado associada) como o número máximo de dados de treinamento que são dominados por M. Se não houver um número máximo, diremos que a dimensão de Vapnik Chervonenkis é infinita. Observemos que se a dimensão de uma máquina de aprendizado é d, então *há pelo menos* um conjunto de d de pontos de treinamentos que são dominados pela classe de classificadores gerados por M, mas *não é necessariamente verdade que todos* os conjuntos de d pontos de treinamento sejam dominados por M.

Exemplo 3.1 *Suponhamos que os dados estejam representados no* $\mathbb{R}^2$ *e que o espaço de hipóteses* H *seja formado por retas (hiperplanos no* $\mathbb{R}^2$*). Assim, faremos uma classificação binária da seguinte forma: os dados que ficarem de um lado da reta estarão em uma classe e os que ficarem do outro lado estarão na outra classe. Conforme a figura abaixo mostra, é possível encontrar 3 pontos que são dominados por* H*:*



Dados dominados no $\mathbb{R}^2$

Portanto, a dimensão VC do espaço de hipóteses H, ou seja, da máquina de aprendizado associada a H, é maior ou igual a 3 porque, para qualquer rotulação dos dados da figura, há um hiperplano classificando-os corretamente. Mas, se tomarmos quatro pontos no $\mathbb{R}^2$, teremos uma das seguintes situações:

- Os quatro pontos formam um polígono (quadrilátero) convexo. Se os vértices opostos pertencerem à mesma classe e os adjacentes pertencerem a classes distintas, não é possível encontrar um hiperplano que os classifique corretamente.

- Três pontos formam um polígono (triângulo) e o quarto ponto está dentro deste polígono. Se o ponto que está no interior do polígono pertencer a uma classe e os demais pertencerem à outra classe, também não será possível encontrar um hiperplano que os classifique corretamente.

Portanto, não há quatro pontos ou mais que são dominados por H e, desta forma, temos que $\dim\mathrm{VC}(H) = 3$. Mostraremos na próxima seção a importância de usar-se máquinas de aprendizado cujo espaço de hipóteses é formado por hiperplanos e apresentaremos um teorema sobre a dimensão de tais espaços de hipóteses.

Apresentaremos agora um teorema que mostra um limite superior sobre a dimensão VC de um máquina de aprendizado com espaço de hipóteses H.

Teorema 3.1 *Seja H um espaço de hipóteses finito associado a uma máquina de aprendizado que faz uma classificação em n classes. Então:*

$$dim VC\,(H) \leq \log_n |H|. \tag{3.3}$$

Prova: *Sabemos que a dimensão VC de um espaço de hipóteses é o tamanho do maior conjunto de dados dominados por este espaço. Sendo d este número, então H realiza corretamente n^d classificações distintas. Porém, por outro lado, também sabemos que para cada classificação distinta feita pela máquina de aprendizado, há uma função h associada e que cada função h faz somente uma classificação. Assim, temos:*

$$n^d \leq |H| \iff d \leq \log_n |H|, \tag{3.4}$$

conforme nós queríamos demonstrar.□

3.3.1 Hiperplanos dominando pontos no $\mathbb{R}^n$

Vamos considerar agora espaços de hipóteses formados por hiperplanos no $\mathbb{R}^n$. Se um conjunto de dados pode ser classificado corretamente por hiperplanos, diremos que este conjunto é *linearmente separável*. Tal classe de classificadores será muito útil no próximo capítulo, quando apresentarmos os conceitos de máquinas de suporte vetorial, principalmente devido à facilidade de formulação do problema de classificação e à rapidez de descobrir-se a classe de um exemplo na fase de treinamento. O teorema a seguir apresenta um resultado muito importante sobre a dimensão VC de espaços de hipóteses formados por hiperplanos:

Teorema 3.2 *Escolhamos um conjunto de m pontos no $\mathbb{R}^n$ e definamos um deles como origem. Então, estes m dados são dominados por hiperplanos no $\mathbb{R}^n$ se e somente os vetores posição dos demais pontos em relação à origem são linearmente independentes.*

Prova: *A demonstração encontra-se em [Bur98, Apêndice].*

Um corolário importante resultante do teorema acima é que a dimensão VC do espaço de hipóteses formando por hiperplanos no $\mathbb{R}^n$ é $n + 1$, já que sempre podemos escolher $n+1$ pontos no $\mathbb{R}^n$, fixando um deles como origem, de forma que os n vetores posições, formados pela diferença entre os demais pontos e a origem, são linearmente independentes. Mas não podem escolher $n + 2$ pontos (ou mais), pois sabemos que a dimensão de $\mathbb{R}^n$ é n e, assim, só há n vetores linearmente independentes.

Capítulo 4

Máquinas de suporte vetorial

Sejam l dados de treinamento (amostras), cada um formado por um vetor $\boldsymbol{x_i} \in \mathbb{R}^d$ e um rótulo $y_i \in \{-1, 1\}$. Suponhamos que estes dados possuam uma distribuição de probabilidade $P(\boldsymbol{X}, Y)$ desconhecida e sejam independentes e identicamente distribuídos. Em outras palavras, são atribuídos rótulos y_i a $\boldsymbol{x_i}$ segundo uma distribuição de probabilidade desconhecida.

Suponhamos que temos uma máquina (no nosso caso, um programa) encarregado de aprender o mapeamento

$$\boldsymbol{x_i} \longmapsto y_i.$$

Este máquina será definida por um conjunto de funções $\boldsymbol{x} \mapsto y$. Durante o restante do texto, suporemos que a máquina de aprendizado será determinística: para um dado $\boldsymbol{x}$ atribuirá sempre o mesmo rótulo y. Ou seja, haverá um rótulo fixo y para cada dado de entrada $\boldsymbol{x}$.

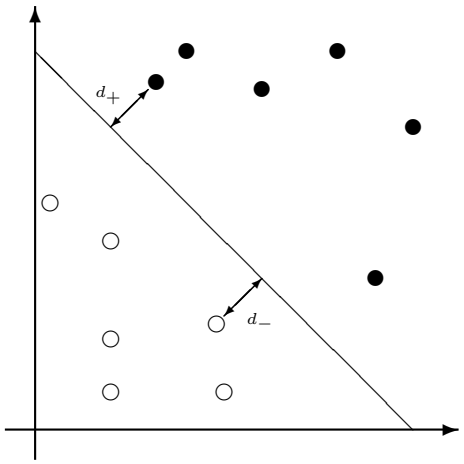
Apresentaremos inicialmente SVMs para dados linearmente separáveis. Em seguida, estenderemos as idéias para o caso de dados não-separáveis linearmente.

4.1 Dados linearmente separáveis

Sejam $\{\boldsymbol{x_i}, y_i\}, \quad i = 1, \ldots, l, \quad \boldsymbol{x_i} \in \mathbb{R}^d, y_i \in \{-1, 1\}$ dados de treinamento como apresentados na seção anterior. Suponhamos que haja um hiperplano definido em $\mathbb{R}^d$ que separe as amostras positivas ($y_i = +1$) das negativas ($y_i = -1$) e chamemos de *hiperplano separador* tal hiperplano. Seja $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$ uma equação para este hiperplano, em que $\boldsymbol{w}$ é o vetor normal ao hiperplano,

$\|\boldsymbol{w}\|$ é uma norma de $\boldsymbol{w}$ e $|b|/\|\boldsymbol{w}\|$ é a distância perpendicular do hiperplano da origem.

Sejam d_+ a menor distância dos pontos classificados como positivos ao hiperplano e d_- a menor distância dos pontos classificados como negativos ao hiperplano. A seguir, temos um exemplo que ilustra tais conceitos:



Hiperplano separador

Definimos por $d_+ + d_-$ a *margem* do hiperplano. A margem de um hiperplano representa a soma das distâncias entre hiperplano e o dados positivos mais próximos e entre o hiperplano e dados os negativos mais próximos.

O objetivo das máquinas de suporte vetorial é maximizar a margem, ou seja, encontrar um hiperplano separador com a maior distância possível do conjunto de dados de treinamento. Definiremos tal hiperplano como *hiperplano separador ótimo.*

Formulemos mais detalhadamente o problema. Suponhamos que as amostras satisfazem as seguintes restrições:

$$\left.\begin{array}{ll} \langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b \geq +1 & \text{se} \quad y_i = +1 \\ \langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b \leq -1 & \text{se} \quad y_i = -1 \end{array}\right\} \qquad i = 1, \ldots, l. \tag{4.1}$$

Estas duas restrições podem ser combinadas em uma só:

$$y_i(\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b) - 1 \geq 0 \qquad i = 1, \ldots, l. \tag{4.2}$$

Os valores +1 e -1 apresentados em (4.1) foram escolhidos apenas para simplificar a formulação do problema que será feita em seguida. Quaisquer

outros valores β e $-\beta$ poderiam ter sido escolhidos ao invés de +1 e -1, respectivamente, desde que $\beta \geq 0$. Para isso, bastaria mudar a escala entre $|b|$ e $\|\boldsymbol{w}\|$.

Suponhamos também que haja pontos para os quais valha a igualdade em (4.2). Tal suposição pode ser feita, pois caso a desigualdade fosse estrita em (4.2), poderíamos trocar o 1 por um outro valor menor, de forma que a igualdade passe a valer para alguns dados de treinamento. Isso também poderia ser feito ao simplesmente mudar a escala entre $|b|$ e $\|\boldsymbol{w}\|$.

Assim, os pontos positivos que satisfazem a igualdade em (4.2) estão no hiperplano $\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b = 1$, que têm uma distância perpendicular da origem de $|1 - b|/\|\boldsymbol{w}\|$, e os negativos estão no hiperplano $\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b = -1$, cuja distância perpendicular da origem é $| - 1 - b|/\|\boldsymbol{w}\|$. Tais hiperplanos são paralelos e, por (4.2), nenhuma amostra de treinamento está entre eles. Assim, temos $d_+ = d_- = 1/\|\boldsymbol{w}\|$ e margem igual a $2/\|\boldsymbol{w}\|$. Portanto, nós queremos resolver o seguinte problema:

$$\begin{array}{rl} \max & 2/\|\boldsymbol{w}\| \\ \text{s.a} & y_i(\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b) - 1 \geq 0 \qquad i = 1, \ldots, l. \end{array} \tag{4.3}$$

Sendo $\|\boldsymbol{w}\| \geq 0$ para qualquer vetor $\boldsymbol{w}$, então maximizar $2/\|\boldsymbol{w}\|$ é equivalente a maximizar $2/\|\boldsymbol{w}\|^2$, que é equivalente a minimizar $\|\boldsymbol{w}\|^2/2$. Portanto, o problema (4.3) é equivalente a:

$$\begin{array}{rl} \min & \|\boldsymbol{w}\|^2/2 \\ \text{s.a} & y_i(\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b) - 1 \geq 0 \qquad i = 1, \ldots, l. \end{array} \tag{4.4}$$

A restrições deste problema são lineares e, portanto, convexas. A função objetivo deste problema também é convexa. Assim, o problema (4.4) é convexo.

A função Lagrangeana associada ao problema (4.4) é:

$$L_P(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{\|\boldsymbol{w}\|^2}{2} - \sum_{i=1}^{l} \alpha_i y_i (\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b) + \sum_{i=1}^{l} \alpha_i \qquad \alpha_i \geq 0, \quad i = 1, \ldots, l. \tag{4.5}$$

em que os α_i são os multiplicadores de Lagrange não-negativos associados à restrições do problema (4.4).

4.1.1 As condições de Karush-Kuhn-Tucker

As condições de Karush-Kuhn-Tucker (KKT)[Ber95, Proposition 3.3.1] são muito importantes tanto na teoria quanto na prática de otimização restrita.

Para o problema (4.4), se $(\boldsymbol{w}^*, b^*)$ for um mínimo local, então há um vetor $\boldsymbol{\alpha}$, para o qual as condições KKT são válidas:

$$\begin{aligned}
\nabla_{\boldsymbol{w}} L_P(\boldsymbol{w^*}, b^*, \boldsymbol{\alpha}) &= \boldsymbol{w^*} - \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x_i} = \mathbf{0} && (4.6)\\
\nabla_b L_P(\boldsymbol{w^*}, b^*, \boldsymbol{\alpha}) &= -\sum_{i=1}^{l} \alpha_i y_i = 0 && (4.7)\\
y_i(\langle \boldsymbol{x_i}, \boldsymbol{w^*} \rangle + b^*) - 1 &\geq 0 \qquad i = 1, \ldots, l && (4.8)\\
\alpha_i &\geq 0 \qquad i = 1, \ldots, l && (4.9)\\
\alpha_i(y_i(\langle \boldsymbol{x_i}, \boldsymbol{w^*} \rangle + b^*) - 1) &= 0 \qquad i = 1, \ldots, l && (4.10)
\end{aligned}$$

Conforme já mencionado anteriormente, sabemos que a função objetivo do problema de SVMs é convexa, tendo restrições lineares e, portanto, um conjunto viável poliedral. Assim, temos que as condições KKT são necessárias e suficientes para $(\boldsymbol{w^*}, b^*)$ ser uma solução ótima do problema (4.4).

4.1.2 O problema dual

Associada ao problema (4.4), o qual chamaremos de *primal*, há também uma *função dual* L_D dada por:

$$L_D(\boldsymbol{\alpha}) = \inf_{(\boldsymbol{w},b) \in (\mathbb{R}^d \times \mathbb{R})} L_P(\boldsymbol{w}, b, \boldsymbol{\alpha}). \tag{4.11}$$

Analisando as condições de KKT (4.6) e (4.7), percebemos que elas também são condições necessárias e suficientes para a função Lagrangeana atingir seu mínimo em relação a $(\boldsymbol{w}, b)$, sendo $\boldsymbol{\alpha}$ fixo. Assim, se $(\boldsymbol{w}^*, b^*)$ for uma solução ótima do problema (4.4), então as condições de KKT são válidas e se nós substituirmos as relações (4.6) e (4.7) na função Lagrangeana L_p acima, temos:

$$L_D(\boldsymbol{\alpha}) = \left(\sum_{i=1}^{l} \alpha_i \; - \; \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle \right) \qquad \alpha_i \geq 0, \quad i = 1, \ldots, l. \tag{4.12}$$

O problema *dual* associado ao problema primal será:

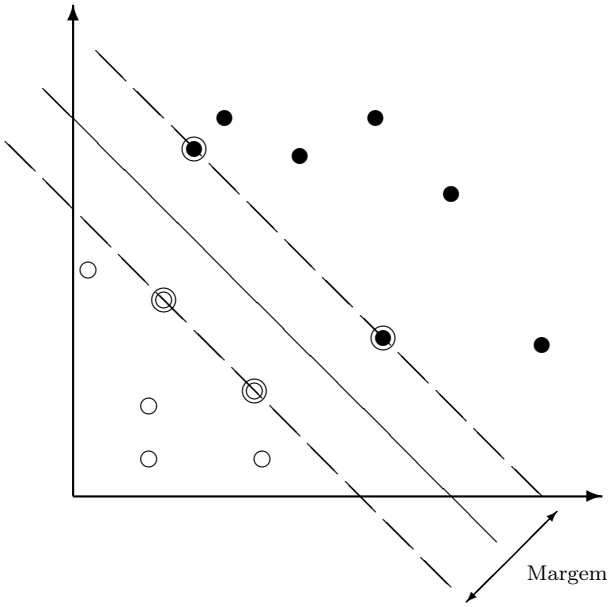
$$\begin{aligned}
\max \quad & L_D(\boldsymbol{\alpha}) \\
\text{s.a} \quad & \alpha_i \geq 0 \qquad i = 1, \ldots, l.
\end{aligned} \tag{4.13}$$

E, segundo [Ber95, Seção 3.4.2], pelo Teorema da Dualidade, nós temos que se o problema (4.4) tiver uma solução ótima, então o problema (4.13) também terá uma solução ótima e o valor ótimo de ambas será o mesmo. Assim, resolver o problema dual é equivalente a resolver o problema primal.

Além disso, a formulação dual do problema tem como vantagem o fato dos dados de treinamentos aparecerem apenas como produtos internos na função objetivo, o que será muito importante nas próximas seções. Outro fato importante é que as restrições da formulação dual são mais simples que as existentes no problema original.

4.1.3 Vetores suporte

Os dados de treinamento para os quais é válida a igualdade em (4.2) são chamados *vetores suporte*, em inglês *support vectors*. Tais pontos definem o hiperplano separador ótimo e sua remoção ou deslocamento pode afetar a solução do problema (4.4), porque pode alterar o valor da margem. A figura a seguir apresenta um exemplo para $\boldsymbol{w} \in \mathbb{R}^2$ com os vetores suporte destacados:



A partir da equação (4.10), vemos que há um multiplicador de Lagrange α_i associado a cada dado de treinamento $(\boldsymbol{x_i}, y_i)$. A partir de (4.10), temos que se $\alpha_i > 0$, então é válida a igualdade em (4.2) e o correspondente dado

de treinamento $(\boldsymbol{x_i}, y_i)$ é um vetor suporte. Por outro lado, se $\alpha_i = 0$, então o dado de treinamento $(\boldsymbol{x_i}, y_i)$ não influi no valor da margem, pois não altera o valor da função objetivo dual[1]. Assim, a condição (4.6) pode ser reduzida para:

$$\nabla_{\boldsymbol{w}} L_P(\boldsymbol{w^*}, b^*, \boldsymbol{\alpha}) = \boldsymbol{w^*} - \sum_{i \in S} \alpha_i y_i \boldsymbol{x_i} \ = \ \boldsymbol{0}, \tag{4.14}$$

em que $S = \{i \mid (\boldsymbol{x_i}, y_i)$ é um vetor suporte$\}$.

Além disso, se todos os demais dados de treinamento fossem removidos ou se fossem deslocados, desde que não se tornassem vetores suporte, o hiperplano separador ainda permaneceria o mesmo.

4.1.4 Fase de testes

Depois de termos treinado uma SVM, como podemos usá-la para classificar novos dados? Basta sabermos em qual lado do hiperplano separador o dado de teste $\boldsymbol{x}$ está. Isso pode ser feito por:

$$\text{sgn}(f(\boldsymbol{x})) = \text{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b), \tag{4.15}$$

atribuindo ao dado de teste o valor devolvido por esta função[2].

Caso o problema dual seja resolvido ao invés do primal, pela equação (4.6), a classe do novo dado será dada por:

$$\text{sgn}(f(\boldsymbol{x})) = \text{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i \langle \boldsymbol{x_i}, \boldsymbol{x} \rangle\right) = \text{sgn}\left(\sum_{i \in S} \alpha_i y_i \langle \boldsymbol{x_i}, \boldsymbol{x} \rangle\right), \tag{4.16}$$

em que $S = \{i \mid (\boldsymbol{x_i}, y_i)$ é um vetor suporte$\}$.

Tal abordagem pode parecer inicialmente pior que a da equação (4.15), pois temos que calcular l produtos internos para saber qual a classe de $\boldsymbol{x}$. Porém, ela será bastante importante nas próximas seções, quando trabalharmos com dados não-linearmente separáveis.

4.2 Dados linearmente não-separáveis

E se os dados não forem linearmente separáveis? Esta situação pode aparecer em muitos problemas reais, seja pela dificuldade de classificação dos dados

[1] Note, entretanto, que o ponto ainda pode ser um vetor suporte

[2] Lembre-se de que a função *sgn* devolve +1, se seu argumento for maior ou igual a zero, e -1, caso contrário.

ou pela existência de ruído. Em tais problemas, a função objetivo dual L_D crescerá arbitrariamente, indo para o infinito. Mas como podemos estender tais idéias para dados não-linearmente separáveis?

Iremos introduzir uma penalização para a violação das restrições (4.2). Desta forma, se um dado de treinamento violar a sua restrição, estando a uma distância perpendicular do hiperplano menor que 1, então haverá um aumento na função objetivo primal. Faremos isto ao adicionar variáveis de folga positivas $\xi_i,\ i = 1,\ldots,l$ às restrições (4.1). Assim, temos novas restrições:

$$\left.\begin{array}{rcll} \langle \boldsymbol{x_i}, \boldsymbol{w}\rangle + b & \geq & +1-\xi_i & \text{se } y_i = +1 \\ \langle \boldsymbol{x_i}, \boldsymbol{w}\rangle + b & \leq & -1+\xi_i & \text{se } y_i = -1 \\ \xi_i & \geq & 0 & \end{array}\right\} \quad i = 1,\ldots,l. \tag{4.17}$$

Podemos combiná-las em duas restrições:

$$\left.\begin{array}{rcl} y_i(\langle \boldsymbol{x_i}, \boldsymbol{w}\rangle + b) - 1 + \xi_i & \geq & 0 \\ \xi_i & \geq & 0 \end{array}\right\} \quad i = 1,\ldots,l. \tag{4.18}$$

Para haver um erro de classificação, devemos ter $\xi_i > 1$, para que o ponto fique do lado oposto do hiperplano separador. Assim, $\sum_{i=1}^{l} \xi_i \geq N_e$, em que N_e é o número de erros de treinamento.

Uma forma de aumentar o valor da função objetivo quando houver violação das restrições é adicionar um termo $C\left(\sum_{i=1}^{l}\xi_i\right)^k$ à mesma, sendo C um parâmetro escolhido pelo usuário, geralmente a partir da natureza do problema e de testes realizados. k indica o quanto queremos penalizar erros de classificação: quanto maior for k, maior será a penalização de erros de classificação ($\xi_i > 1$) e menor será a penalização para dados que violam o hiperplano positivo (ou negativo) de margem 1, mas não são classificados incorretamente ($\xi_i \leq 1$). C representa a importância dos erros para a função objetivo. Se k é inteiro, então o problema é convexo e se $k = 1$ ou $k = 2$, o problema é também quadrático. Para $k = 1$, teremos o seguinte problema:

$$\begin{array}{lll} \min & \frac{\|\boldsymbol{w}\|^2}{2} + C\displaystyle\sum_{i=1}^{l}\xi_i & \\ \text{s.a} & y_i(\langle \boldsymbol{x_i}, \boldsymbol{w}\rangle + b) - 1 + \xi_i \ \geq\ 0 & i = 1,\ldots,l \\ & \xi_i \ \geq\ 0 & i = 1,\ldots,l. \end{array} \tag{4.19}$$

Este problema, o qual chamaremos de *primal*, tem uma função objetivo convexa e restrições lineares e, portanto, ele é convexo.

A função Lagrangeana associada ao problema (4.19) será:

$$\begin{aligned} L_P(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) &= \frac{\|\boldsymbol{w}\|^2}{2} - \sum_{i=1}^{l} \alpha_i (y_i(\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b) - 1 + \xi_i) + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \mu_i \xi_i \\ &= \frac{\|\boldsymbol{w}\|^2}{2} - \sum_{i=1}^{l} \alpha_i y_i (\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b)) + \sum_{i=1}^{l} (C - \alpha_i - \mu_i) \xi_i + \sum_{i=1}^{l} \alpha_i \\ & \alpha_i, \mu_i, \xi_i \geq 0, \quad i = 1, \ldots, l, \end{aligned} \tag{4.20}$$

em que $\boldsymbol{\alpha}$ é o vetor dos multiplicadores de Lagrange associados às restrições do problema original e $\boldsymbol{\mu}$ é o vetor dos multiplicadores de Lagrange associado às restrições de $\xi_i \geq 0,\ i = 1, \ldots, l$.

As condições KKT do problema (4.19) são:

$$\nabla_{\boldsymbol{w}} L_P = \boldsymbol{w} - \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x_i} = \boldsymbol{0} \tag{4.21}$$

$$\nabla_b L_P = -\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{4.22}$$

$$\nabla_{\xi_i} L_P = C - \alpha_i - \mu i = 0 \qquad i = 1, \cdots, l \tag{4.23}$$

$$y_i(\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b) - 1 + \xi_i \geq 0 \qquad i = 1, \cdots, l \tag{4.24}$$

$$\alpha_i \geq 0 \qquad i = 1, \cdots, l \tag{4.25}$$

$$\mu_i \geq 0 \qquad i = 1, \cdots, l \tag{4.26}$$

$$\xi_i \geq 0 \qquad i = 1, \cdots, l \tag{4.27}$$

$$\alpha_i((y_i(\langle \boldsymbol{x_i}, \boldsymbol{w} \rangle + b) - 1 + \xi_i) = 0 \qquad i = 1, \cdots, l \tag{4.28}$$

$$\mu_i \xi_i = 0 \qquad i = 1, \cdots, l. \tag{4.29}$$

A função dual associada ao problema (4.19) será:

$$L_D(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \inf_{(\boldsymbol{w}, b, \boldsymbol{\xi}) \in (\mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^l)} L_P(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}). \tag{4.30}$$

Se $(\boldsymbol{w^*}, b, \boldsymbol{\xi})$ for uma solução do problema (4.19), então as condições de KKT descritas acima são válidas e, além disso, as condições (4.21), (4.22) e (4.23) garantem que L_p terá valor mínimo para $\boldsymbol{\alpha}$ e $\boldsymbol{\mu}$ fixos. Assim, ao substituí-las na equação (4.20), a função dual torna-se:

$$L_D(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y j \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle \right), \tag{4.31}$$

ou seja, a função dual (4.31) do problema que aceita dados não-linearmente separáveis é igual à função dual (4.12) do problema que trata somente dados linearmente separáveis.

De forma análoga à empregada na seção anterior, temos o seguinte problema dual:

$$\begin{aligned} \max \quad & L_D(\boldsymbol{\alpha}, \boldsymbol{\mu}) \\ \text{s. a} \quad & 0 \leq \alpha_i \leq C \quad i = 1, \ldots, l. \end{aligned} \tag{4.32}$$

Nós podemos observar neste problema que as variáveis de folga ξ_i e os multiplicadores de Lagrange μ_i associados não aparecem, o que reduz a quantidade de variáveis envolvidas. Além disso, os dados de treinamento aparecem apenas como produtos internos, o que será muito importante na próxima seção.

4.3 Máquinas de suporte vetorial não-lineares

Será possível fazer corretamente a classificação dos dados quando não é possível encontrar um hiperplano que separe os dados? Ou seja, quando não é possível encontrar uma função linear que os classifique corretamente. Mostraremos como a classificação poderá ser feita em tais casos.

Consideremos como exemplo a função:

$$f(m_1, m_2, r) = C\frac{m_1 m_2}{r^2}, \tag{4.33}$$

que representa a lei de gravitação de Newton, que determina a força de atração gravitacional entre dois corpos de massa m_1 e m_2 que estão a uma distância r um do outro. Podemos notar que esta função não é linear e, assim, não seria possível fazer encontrar um hiperplano separador dos dados se eles fossem classificados por f. Entretanto, se fizermos uma mudança não-linear nas coordenadas dos dados:

$$(m_1, m_2, r) \longmapsto (t, u, v) = (\ln m_1, \ln m_2, \ln r), \tag{4.34}$$

teremos uma nova função:

$$g(t, u, v) = \ln f(m_1, m_2, r) = \ln C + \ln m_1 + \ln m_2 - 2\ln r = c + t + u - 2v, \tag{4.35}$$

para qual é possível encontrar um hiperplano separador dos dados.

Assim, o mapeamento dados de um espaço para outro pode tornar possível a existência de um hiperplano separador dos dados, permitindo a separação linear.

Seja $\mathcal{L} = \mathbb{R}^d$ o espaço de Hilbert no qual estão os dados inicialmente. Chamemos de *espaço origem* tal espaço. E suponhamos que os dados sejam mapeados para um outro espaço de Hilbert $\mathcal{H}$ através de de uma função Φ:

$$\Phi : \mathcal{L} \mapsto \mathcal{H} \tag{4.36}$$

Como no espaço origem os dados aparecem apenas como produtos internos $\langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle$ no problema, então no espaço $\mathcal{H}$, o qual iremos chamar de *espaço de características*, eles também irão aparecer como produtos internos da forma $\langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x_j}) \rangle$.

Porém, pode ser muito difícil calcular Φ explicitamente, pois $\mathcal{H}$ pode ter uma dimensão muito grande, ou até mesmo infinita. Mas se houvesse uma função K tal que $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x_j}) \rangle$, e se nós usássemos $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ no lugar de $\langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x_j}) \rangle$ no algoritmo de treinamento, então teríamos uma nova máquina de suporte vetorial que faria a separação linear no espaço $\mathcal{H}$. Além disso, não precisaríamos calcular explicitamente Φ e nem mesmo saber como ela é. Tal função K será chamada de *função kernel* e possibilitará fazer a classificação em um tempo próximo ao gasto no espaço origem $\mathcal{L}$.

Mas como iremos determinar a classe de um novo dado de teste $\boldsymbol{x}$ após termos feito o treinamento da máquina no espaço $\mathcal{H}$? Substituindo $\langle \boldsymbol{x_i}, \boldsymbol{x} \rangle$ por $\langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x}) \rangle$ na equação (4.16), teremos:

$$\begin{aligned} sgn(f(\boldsymbol{x})) &= sgn(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b) = sgn\left(\sum_{i \in S} \alpha_i y_i \langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x}) \rangle + b \right) \\ &= sgn\left(\sum_{i \in S} \alpha_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b \right). \end{aligned} \tag{4.37}$$

Notemos que agora $\boldsymbol{w}$ está no espaço $\mathcal{H}$, que é onde é feita a separação dos dados. Porém, geralmente, não haverá ou não será fácil encontrar um vetor $\boldsymbol{z} \in \mathcal{L}$ tal que $\Phi(\boldsymbol{z}) = \boldsymbol{w}$. Por tal motivo, é geralmente sv vezes mais lento fazer a classificação dos dados no espaço de características que no espaço origem, sendo sv a quantidade de vetores suporte. Portanto, se houver tal vetor, haverá uma redução de sv vezes no tempo de classificação

de um dado de teste da SVM, pois não será necessário fazer a somatória da equação (4.37).

Apresentaremos agora um exemplo básico de um kernel e de uma função Φ que fará o mapeamento do dados do espaço $\mathcal{L}$ para o espaço $\mathcal{H}$.

Exemplo 4.1 *Suponhamos que nossos dados estejam em* $\mathbb{R}^2$ *e seja* $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle)^2$. *Vamos encontrar um espaço de características* $\mathcal{H}$ *e uma mapeamento* Φ *tal que* $(\langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle)^2 = \langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x_j}) \rangle$. *Seja* $\mathcal{H} = \mathbb{R}^3$ *e*

$$\Phi(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

a função que mapeia os dados de $\mathcal{L}$ *em* $\mathcal{H}$. *Sendo* $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x_j}) \rangle$, *observamos que* K *é um kernel em* $\mathcal{H}$.

Mas nem o espaço de características e nem o mapeamento Φ *são únicos para um kernel. No exemplo anterior, poderíamos ter* $\mathcal{H} = \mathbb{R}^3$ *e*

$$\Phi(\boldsymbol{x}) = \frac{1}{\sqrt{2}} \begin{pmatrix} x_1^2 - x_2^2 \\ 2x_1x_2 \\ x_1^2 + x_2^2 \end{pmatrix}$$

ou $\mathcal{H} = \mathbb{R}^4$ *e*

$$\Phi(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ x_1x_2 \\ x_1x_2 \\ x_2^2 \end{pmatrix},$$

e, em ambos, K *seria um kernel também.*

4.4 Teorema de Mercer

O uso de funções kernel parece ser muito interessante para máquinas de suporte vetorial, pois não precisamos determinar explicitamente qual será o novo espaço de características e nem como será feito o mapeamento Φ entre os espaços, o que pode ser muito difícil, tanto computacional quanto matematicamente.

Mas, para quais funções kernels K existe um espaço de Hilbert $\mathcal{H}$ e uma mapeamento Φ tal que $K(\boldsymbol{x}, \boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$, para quaisquer dados $\boldsymbol{x}$ e $\boldsymbol{z}$?

Duas condições necessárias para K são:

- A função deve ser simétrica, isto é, para quaisquer dados $\boldsymbol{x}$ e $\boldsymbol{z}$ devemos ter:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle = \langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{x}) \rangle = K(\boldsymbol{z}, \boldsymbol{x}), \tag{4.38}$$

- Deve satisfazer a desigualdade de Cauchy-Schwarz:

$$\begin{array}{rcl} K(\boldsymbol{x}, \boldsymbol{z})^2 & = & \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle^2 \ \leq \ \|\Phi(\boldsymbol{x})\|^2 \|\Phi(\boldsymbol{z})\|^2 \\ & = & \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}) \rangle \langle \Phi(\boldsymbol{z}), \Phi(\boldsymbol{z}) \rangle = K(\boldsymbol{x}, \boldsymbol{x}) K(\boldsymbol{z}, \boldsymbol{z}) \end{array} \tag{4.39}$$

Mas estas condições não são suficientes para garantir a existência de um espaço de características $\mathcal{H}$. A resposta será dada pelo Teorema de Mercer [CST02, Seção 3.3.1].

Consideremos inicialmente n dados no espaço origem finito $\mathcal{L} = \{x_1, \ldots, x_n\}$. E seja $K(\boldsymbol{x}, \boldsymbol{z})$ uma função simétrica definida sobre o espaço de origem. Definiremos a matriz $\boldsymbol{K}$ como:

$$\boldsymbol{K} = (K(\boldsymbol{x_i}, \boldsymbol{x_j}))_{i,j=1}^{n}. \tag{4.40}$$

Esta matriz é simétrica, pois K é simétrica e, assim, existe uma matriz ortogonal $\boldsymbol{V}$ tal que $\boldsymbol{K} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V'}$, em que $\boldsymbol{V}$ é matriz diagonal contendo os autovalores λ_i de $\boldsymbol{K}$, com autovetores correspondentes $\boldsymbol{v_i}, i = 1, \ldots, n$.

Suponhamos agora que os autovalores de $\boldsymbol{K}$ são todos não-negativos e seja Φ o seguinte mapeamento:

$$\Phi : \boldsymbol{x_i} \mapsto (\sqrt{\lambda_i} v_{ti})_{t=1}^{n} \in \mathbb{R}^n, i = 1, \ldots, n. \tag{4.41}$$

Agora, utilizando Φ em um produto interno no $\mathbb{R}^n$, temos:

$$\langle \Phi(\boldsymbol{x_i}), \Phi(\boldsymbol{x_j}) \rangle = \sum_{t=1}^{n} \lambda_i v_{ti} v_{tj} = (\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V'})_{ij} = \boldsymbol{K}_{ij} = K(\boldsymbol{x_i}, \boldsymbol{x_j}), \tag{4.42}$$

resultando que K é uma função kernel associada com Φ, tendo como $\mathbb{R}^n$ como espaço de características.

A exigência de que $\boldsymbol{K}$ tenha autovalores não-negativos é necessária. Suponhamos que $\boldsymbol{K}$ tenha uma autovalor λ_r negativo com autovetor $\boldsymbol{v_r}$ associado. Assim, nós podemos definir o ponto:

$$z = \sum_{i=1}^{n} v_{ri} \Phi(\boldsymbol{x_i}) = \sqrt{\Lambda} \boldsymbol{V'} \boldsymbol{v_r}, \tag{4.43}$$

pertencente ao espaço de características, pois é uma combinação linear de pontos do espaço de características. Este ponto tem a seguinte norma elevada ao quadrado:

$$\|\boldsymbol{z}\| = \langle \boldsymbol{z}, \boldsymbol{z} \rangle = \boldsymbol{v_r'} \boldsymbol{V} \sqrt{\boldsymbol{\Lambda}} \sqrt{\boldsymbol{\Lambda}} \boldsymbol{V'} \boldsymbol{v_r} = \boldsymbol{v_r'} \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V'} \boldsymbol{v_r} = \boldsymbol{v_r'} \boldsymbol{K} \boldsymbol{v_r} = \lambda_r \leq 0, \tag{4.44}$$

o que contradiz o fato do espaço de características ser um espaço de Hilbert.

Assim, K deve ter autovalores não-negativos, o que significa que ela deve ser positiva semidefinida. Portanto, provamos a seguinte proposição:

Proposição 4.1 *Seja $\mathcal{L} = \{x_1, \ldots, x_n\}$ uma espaço origem finito com uma função simétrica K definida sobre ele. Então, $K(\boldsymbol{x}, \boldsymbol{z})$ será uma função kernel se e somente se a matriz*

$$\boldsymbol{K} = (K(x_i, x_j))_{i,j=1}^{n},$$

for semidefinida positiva.

Vamos agora estender este conceito para espaços de dimensão infinita. Adicionaremos um peso λ_i para cada dimensão do espaço, obtendo um novo produto interno:

$$\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle = \sum_{i=1}^{\infty} \lambda_i \Phi_i(\boldsymbol{x}) \Phi_i(\boldsymbol{z}) = K(\boldsymbol{x}, \boldsymbol{z}). \tag{4.45}$$

O teorema de Mercer dará as condições necessárias e suficientes para que uma função simétrica contínua K tenha um mapeamento Φ e um espaço de características $\mathcal{H}$ associados. Ou seja:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{i=1}^{\infty} \lambda_i \Phi_i(\boldsymbol{x}) \Phi_i(\boldsymbol{z}). \tag{4.46}$$

Teorema 4.2 (Mercer) *Seja K uma função simétrica contínua definida sobre uma subconjunto compacto $\boldsymbol{X}$. Se para toda função $g(\boldsymbol{x})$ definida sobre $\boldsymbol{X}$, tal que*

$$\int_X g(\boldsymbol{x})^2 \mathrm{d}x < \infty, \tag{4.47}$$

é válido que

$$\int_{X \times X} K(\boldsymbol{x}, \boldsymbol{z}) g(\boldsymbol{x}) g(\boldsymbol{z}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{z} \geq 0, \tag{4.48}$$

então há uma expansão de K numa série uniformemente convergente:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{i=1}^{\infty} \lambda_i \Phi(\boldsymbol{x}) \Phi(\boldsymbol{z}), \tag{4.49}$$

como nós queríamos.

Observemos que a condição (4.48) para espaços de dimensão infinita é equivalente à condição de $\boldsymbol{K}$ ser semidefinida positiva para espaços de dimensão finita na proposição (4.1).

Para alguns kernels, pode ser difícil verificar se o teorema de Mercer é válido, pois a desigualdade (4.48) deve ser satisfeita para toda função g que satisfaça a desigualdade (4.47) (tenha norma L_2 finita). Mas, para kernels da forma:

$$K(\boldsymbol{x}, \boldsymbol{z}) = (\langle \boldsymbol{x}, \boldsymbol{z} \rangle)^p, p \text{ inteiro positivo}, \tag{4.50}$$

nós mostraremos que é fácil provar que K satisfaz o teorema de Mercer.

Nós devemos mostrar que

$$\int_{X \times X} \left(\sum_{i=1}^{d} x_i z_i \right) g(\boldsymbol{x}) g(\boldsymbol{z}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{z} \geq 0, \tag{4.51}$$

para toda função g que satisfaça a condição (4.47).

Ao fazermos a expansão multinomial do lado esquerdo da desigualdade acima, teremos:

$$\frac{p!}{r_1! r_2! \ldots (p - r_1 - r_2 \ldots)!} \int_{X \times X} x_1^{r_1} x_2^{r_2} \ldots y_1^{r_1} y_2^{r_2} \ldots g(vetx) g(\boldsymbol{z}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \tag{4.52}$$

e ao fatorarmos, teremos:

$$\frac{p!}{r_1! r_2! \ldots (p - r_1 - r_2 \ldots)!} \left(\int_{X \times X} x_1^{r_1} x_2^{r_2} \ldots g(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \right)^2 \geq 0, \tag{4.53}$$

o que conclui a nossa prova.

Uma resultado importante da prova acima é que qualquer função kernel que possa ser expressa na forma:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{p=0}^{\infty} c_p \langle \boldsymbol{x}, \boldsymbol{z} \rangle^p, \ c_p \geq 0, \tag{4.54}$$

e a série seja uniformemente convergente, estará satisfazendo o teorema de Mercer.

4.5 Exemplos de kernels

Dentre os kernels mais estudados para reconhecimento de padrões, podemos destacar:

- **Kernels polinomiais:** a função kernel é da forma:

$$K(\boldsymbol{x}, \boldsymbol{z}) = (\langle \boldsymbol{x}, \boldsymbol{z} \rangle + c)^p, \tag{4.55}$$

  em que p é um inteiro positivo e c é um número real. Este tipo de kernel pode ser imaginado como uma generalização de hiperplanos separadores ($p = 1$) e uma de suas vantagens é o fato de que o cálculo de $(\langle \boldsymbol{x}, \boldsymbol{z} \rangle + c)^p$ através de $K(\boldsymbol{x}, \boldsymbol{z})$ requer $\mathcal{O}(pd)$ operações, enquanto que o cálculo explícito de $(\langle \boldsymbol{x}, \boldsymbol{z} \rangle + c)^p$ ("abrindo" a expressão) iria precisar de $\mathcal{O}(\binom{p+d}{p})$ operações[3].

- **Kernels de base Gaussiana radial:** a função kernel é da forma:

$$K(\boldsymbol{x}, \boldsymbol{z}) = e^{-\|\boldsymbol{x}-\boldsymbol{z}\|^2/2\sigma^2}. \tag{4.56}$$

- **Kernels baseados em redes neurais sigmoidais de duas camadas:** a função kernel é da forma:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \tanh(\kappa \langle \boldsymbol{x}, \boldsymbol{z} \rangle - \delta), \tag{4.57}$$

  em que a primeira camada consiste de $|S|$ (quantidade de vetores suporte) conjuntos de pesos, sendo cada conjunto composto por d elementos, e a segunda camada consiste de $|S|$ pesos (os α_i associados aos vetores suporte). Assim, para determinar-se a classe de um novo dado, é feita a soma ponderada de sigmóides, cada um calculado como um produto interno entre um vetor suporte e o dado sendo testado.

Durante o restante do texto, nós nos concentraremos em kernels polinomiais, devido à sua simples representação e à rapidez para se determinar a classe de um dado de teste.

[3]Lembre-se de que d é a dimensão dos dados no espaço de origem.

4.6 Soluções de máquinas de suporte vetorial

Analisaremos se as soluções dos problemas (4.4) e (4.19) da fase de treinamento de SVMs são globais e se são únicas. Estaremos interessados em dois tipos de unicidade das soluções $\{\boldsymbol{w}, b\}$: se o par $\{\boldsymbol{w}, b\}$ é único e se a expansão de $\boldsymbol{w}$ em função de $\boldsymbol{\alpha}$ é única. O primeiro tipo tem um interesse principalmente numérico, pois caso haja mais de uma solução $\{\boldsymbol{w}, b\}$, pode ser que uma tenha menos erros de arrendamento que as demais. E se a expansão de $\boldsymbol{w}$ em função de $\boldsymbol{\alpha}$ não for única, pode ser que uma expansão requeira um número menor de vetores suporte e assim, a determinação da classe de um novo dado na fase de testes (equação (4.16)) seria mais rápida.

Conforme já mencionado anteriormente, temos que os problemas (4.4) e (4.19) são convexos. E, segundo [Fri05, Proposição 3.3], sabemos que toda solução local de um problema convexo é também global. Portanto, toda solução de um problema da fase de treinamento de SVMs é global. Além disso, se a função objetivo for estritamente convexa, ou seja, se a matriz Hessiana $H_{ij} = y_i y_j \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle$ for definida positiva, então a solução será única. Porém, mesmo se a matriz for semidefinida positiva, a solução pode ainda ser única: tomemos, por exemplo, dois pontos em $\mathbb{R}$: $x_1 = 1$ e $x_2 = 2$ e rótulos $y_1 = +1$ e $y_2 = -1$, respectivamente. A matriz Hessiana neste caso é semidefinida positiva, mas a solução ($\boldsymbol{w} = -2,\ b = 3,\ \xi_i = 0$) do problema (4.19) é única. Mesmo quando a solução $\{\boldsymbol{w}, b\}$ é única, a expansão de $\boldsymbol{w}$ em função de $\boldsymbol{\alpha}$ pode não ser única. Tomemos, por exemplo, quatro pontos no $\mathbb{R}^2$: $x_1 = (1, 1),\ x_2 = (-1, 1),\ x_3 = (-1, -1)$ e $x_4 = (1, -1)$. com rótulos $y_1 = +1,\ y_2 = -1,\ y_3 = -1$ e $y_4 = +1$, respectivamente. Uma solução do problema (4.4) é ($\boldsymbol{w} = (1, 0),\ b = 0$ e $\boldsymbol{\alpha} = (0.25, 0.25, 0.25, 0.25)$) e uma outra solução tem os mesmos valores de $\boldsymbol{w}$ e b, mas $\boldsymbol{\alpha} = (0.5, 0.5, 0, 0)$.

Observamos então que soluções $\{\boldsymbol{w}, b\}$ podem não ser únicas somente se a matriz Hessiana não for definida positiva. Apresentaremos a seguir um teorema que mostra uma consequência do fato de existirem soluções não-únicas para os problema (4.4) e (4.19).

Teorema 4.3 *Chamemos de $\boldsymbol{X}$ o par de variáveis $\{\boldsymbol{w}, b\}$. Suponhamos que a matriz Hessiana para o problema* (4.4) *(ou o problema* (4.19)*) seja semidefinida positiva e, assim, a função objetivo é convexa. Sejam $\boldsymbol{X_0}$ e $\boldsymbol{X_1}$ duas soluções do problema. Então $\boldsymbol{X}(\tau) = (1 - \tau)\boldsymbol{X_0} + \tau \boldsymbol{X_1}, \quad \tau \in [0, 1]$ também é uma solução do mesmo problema.*

Prova: *Seja F a função objetivo do problema e F_{min} seu valor mínimo.*

Pela convexidade de F, temos que $F(\boldsymbol{X}(\tau)) \leq (1-\tau)F(\boldsymbol{X_0}) + \tau F(\boldsymbol{X_1}) = F_{min}$, para todo $\tau \in [0,1]$. Mas então, $F(\boldsymbol{X}(\tau)) \leq F_{min}$, o que implica que $F(\boldsymbol{X}(\tau)) = F_{min}$, pois F_{min} é um ponto de mínimo global de F. E, portanto, $F(\boldsymbol{X}(\tau))$ é uma solução do problema também. □

Então, um problema da forma (4.4) (ou da forma (4.19)) com uma matriz Hessiana semidefinida positiva somente terá soluções não-únicas se for possível deslocar-se linearmente de uma solução para outra, de forma que todas os pontos no deslocamento também sejam soluções do problema.

4.6.1 Dimensão VC de máquinas de suporte vetorial

Mostraremos nesta seção que a dimensão VC de máquinas de suporte vetorial pode ser muito grande (até mesmo infinita), apenas delas apresentarem geralmente um bom desempenho[4].

Nós denominaremos um *kernel de Mercer (*ou *um kernel positivo)* qualquer kernel que satisfaça o teorema de Mercer, e o espaço $\mathcal{H}$ associado de dimensão mínima de *espaço associado minimal*. Temos também o seguinte teorema:

Teorema 4.4 *Seja K um kernel positivo com espaço associado minimal $\mathcal{H}$. Então a dimensão VC da máquina de suporte vetorial correspondente é $dim(\mathcal{H}) + 1$.*

Prova: *Seja $d_{\mathcal{H}}$ a dimensão do espaço associado minimal $\mathcal{H}$ e Φ um mapeamento de $\mathfrak{L}$ para $\mathcal{H}$ associado a K também. Então, há $d_{\mathcal{H}}$ pontos na imagem de Φ cujos vetores posições em relação à origem são linearmente independentes. Pelo teorema* (3.2)*, temos que estes $d_{\mathcal{H}}+1$ pontos (incluindo a origem) são dominados por hiperplanos em $\mathcal{H}$. Então, se considerarmos apenas dados linearmente separáveis (seção* (4.1)*) ou se permitirmos que a penalidade C para dados não-linearmente separáveis (seção* (4.2)*) admita qualquer valor, de forma que se os dados forem realmente separáveis haverá um solução que os separa (mesmo com margem menor que $2/\|\boldsymbol{w}\|$), então a família de máquinas de suporte vetorial com kernel K também domina estes pontos e, portanto, tem dimensão VC igual a $d_{\mathcal{H}} + 1$.* □

Vejamos agora a dimensão VC de um tipo de kernel bem conhecido.

[4]Observe, entretanto, que ainda não há nenhuma prova que garanta o bom desempenho de uma certa família de SVM's para um dado problema

4.7 A dimensão VC de kernels polinomiais

Consideremos uma SVM com um kernel homogêneo polinomial ($c = 0$) de grau p, sendo $\mathbb{R}^{d_{\mathcal{L}}}$ o espaço origem:

$$K(\boldsymbol{x_1}, \boldsymbol{x_2}) = (\langle \boldsymbol{x_1}, \boldsymbol{x_2} \rangle)^p, \quad \boldsymbol{x_1}, \boldsymbol{x_2} \in \mathbb{R}^{d_{\mathcal{L}}}. \tag{4.58}$$

Como nós fizemos no exemplo (4.1), podemos construir explicitamente um mapeamento Φ de $\mathcal{L}$ para um espaço $\mathcal{H}$ tal que $K(\boldsymbol{x_1}, \boldsymbol{x_2}) = \langle \Phi(\boldsymbol{x_1}), \Phi(\boldsymbol{x_2}) \rangle = (\langle \boldsymbol{x_1}, \boldsymbol{x_2} \rangle)^p$. Seja $z_i = x_{1i}x_{2i}$ tal que $K(\boldsymbol{x_1}, \boldsymbol{x_2}) = (z_1 + \cdots + z_{d_{\mathcal{L}}})^p$. Então, percebemos que cada coordenada de $\Phi(\boldsymbol{x})$ estará relacionada a um termo com certas de potências de $z_i,\ i = 1, \ldots, d_{\mathcal{L}}$. E se rotularmos as componentes de $\Phi(\boldsymbol{x})$ desta forma, poderemos explicitamente construir o mapeamento de $\mathcal{L}$ para $\mathcal{H}$:

$$\Phi_{r_1 r_2 \cdots, r_{d_{\mathcal{L}}}}(\boldsymbol{x}) = \sqrt{\left(\frac{p!}{r_1! r_2! \cdots r_{d_{\mathcal{L}}}!}\right)} x_1^{r_1} x_2^{r_2} \cdots x_{d_{\mathcal{L}}}^{r_{\mathcal{L}}}, \quad \sum_{i=1}^{d_{\mathcal{L}}} r_i = p,\ r_i \geq 0. \tag{4.59}$$

Seguindo este raciocínio descrito acima, temos então o seguinte teorema:

Teorema 4.5 *Se o espaço origem tem dimensão $d_{\mathcal{L}}$, então a dimensão do espaço de características minimal para kernels polinomiais homogêneos de grau p ($K(\boldsymbol{x_1}, \boldsymbol{x_2}) = (\langle \boldsymbol{x_1}, \boldsymbol{x_2} \rangle)^p, \quad \boldsymbol{x_1}, \boldsymbol{x_2} \in \mathbb{R}^{d_{\mathcal{L}}}$) é $\binom{d_{\mathcal{L}}+p-1}{p}$.*

Prova: *A prova encontra-se em [Bur98, Apêndice].*

4.8 Implementações

Após termos apresentado os principais conceitos envolvidos com a teoria de máquinas de suporte vetorial, falaremos agora sobre algumas formas de resolução do problema.

Conforme já discutido anteriormente, os problemas (4.4) e (4.19) são convexos e, particularmente, são quadráticos. Assim, podemos usar algum algoritmo geral para problemas deste tipo, como o método de Newton ou de gradientes conjugados. Entretanto, podemos aproveitar a estrutura particular do problema de SVM's e usar algum algoritmo específico, para melhorar o desempenho. Um exemplo é o algoritmo SMO (*Sequential Minimal Optimization*). Para mais detalhes sobre estes algoritmos, sugerimos a consulta de [Bur98, Seção 5] e [CST02, Capítulo 7].

Um implementação de máquinas de suporte vetorial muito usada em experimentos para avaliações de desempenho é a SVM*light* ([Joa99]), criada e mantida por Thorsten Joachims e disponível em [Svml05]. A implementação foi feita na linguagem C e realiza várias otimizações, além de permitir algumas personalizações pelo usuário como, por exemplo, a escolha de um próprio kernel. Utilizamos esta implementação em nossos experimentos para detecção de spam que serão apresentados no próximo capítulo.

Capítulo 5

Aplicação de máquinas de suporte vetorial na detecção de spam

Para aprofundarmos os conceitos estudados durante a iniciação e verificarmos a eficiência de máquinas de suporte vetorial em exemplos reais, decidimos realizar alguns experimentos para a detecção de spam. Tal estudo foi baseado principalmente no artigo *Support Vector Machines for Spam Categorization* [HWV99], em que é feita uma comparação de desempenho de SVM's com outros métodos de reconhecimento de padrões na execução desta tarefa.

5.1 Apresentação do problema

Neste problema, o universo de interesse será formado por mensagens eletrônicas, os *e-mails*, sendo cada mensagem composta por palavras de um idioma. Estaremos interessados em classificar as mensagens em dois tipos:

- spams
- normais (não-spams ou hams).

Como os elementos originais do problema são cadeias de caracteres (palavras), nós precisamos encontrar um representação numérica para eles, de forma a facilitar o manuseio tanto durante a fase de testes quanto de treinamento. Tal representação deve ser reduzida, para evitarmos redudâncias

mas, ao mesmo tempo, deve manter todas as informações importantes para a classificação sobre cada elemento.

Definiremos uma *característica* como uma palavra de um e-mail. Nós criaremos um *dicionário* formado por palavras existentes nas mensagens e teremos associado a cada mensagem um *vetor de características* $\boldsymbol{x}$, que relacionará cada palavra do dicionário com a mensagem.

Para a criação do vetor de características, foram escolhidas duas abordagens:

- **Representação binária:** a i-ésima coordenada do vetor de características associado a uma mensagem indica se a i-ésima palavra do dicionário está presente ($x_i = 1$) ou não ($x_i = 0$) na mensagem.

- **Freqüência da palavra:** a i-ésima coordenada do vetor de características associado a uma mensagem indica quantas vezes a i-ésima palavra do dicionário ocorre na mensagem.

Foram usadas todas as características (palavras) ao invés de selecionarmos as mais significativas, pois a seleção tem consumo de tempo $\omega(dn)$, em que d é o tamanho do dicionário e n é a quantidade de mensagens analisadas. Além disso, a seleção de características não tem apresentado uma melhora substancial na eficiência de SVM's para o tratamento de spams.

Para a geração do dicionário, só foram consideradas palavras que aparecessem em pelo menos 3 mensagens diferentes. Este enfoque foi adotado para reduzir-se o tamanho do dicionário e eliminar palavras digitadas incorretamente ou pouco usadas, sem prejudicar substancialmente o desempenho da classificação.

E na criação do dicionário, todas as palavras tiveram suas letras convertidas para minúsculas, de forma que palavras como "love" e "LoVE" foram representadas pela mesma característica.

Dois enfoques foram adotados para analisarmos as mensagens:

- Somente o corpo (*body*) da mensagem.

- O assunto/título (*subject*) e o corpo da mensagem.

Foram usadas mensagens pertencentes a somente um idioma, sendo este o inglês. Tal escolha deveu-se principalmente à grande quantidade de e-mails existentes e por ser mais fácil realizar a análise léxica de textos (*parser*) neste idioma.

5.2 Medidas de desempenho

Para medir-se o desempenho de máquinas de suporte vetorial aplicadas à detecção de spam, as seguintes taxas foram adotadas:

- **Taxa de erro (TE):** dada por:

$$\frac{\text{número de e-mals classificados incorretamente}}{\text{total de e-mails analisados}} \tag{5.1}$$

- **Taxa de falsos-positivos (TFP):** dada por:

$$\frac{\text{quantidades de e-mails normais classificados como spams}}{\text{total de e-mails normais analisados}} \tag{5.2}$$

- **Taxa de falsos-negativos (TFN):** dada por:

$$\frac{\text{quantidade de spams classificados como normais}}{\text{total de spams analisados}} \tag{5.3}$$

5.3 Algoritmos e implementações utilizadas

Para fazermos a análise das palavras das mensagens, a geração do dicionário e dos vetores de características, foram usados scripts desenvolvidos por mim na linguagem de programação Python [Pyt05], versão 2.3.5.

Para testarmos os dados, usamos a implementação SVM*light*, apresentada no capítulo anterior, devido ao seu bom desempenho, ao grande uso pela comunidade acadêmica para experimentos científicos e por ser grátis para estes fins e de código-fonte aberto, o que permitiu que entendêssemos os algoritmos.

Como função kernel, usamos um hiperplano no espaço de origem dos dados[1], devido à simplicidade de escolha, pois não é preciso especificar parâmetros, além de ser muita rápida na fase de testes.

5.4 Ambiente computacional utilizado

O ambiente computacional utilizado nos experimentos foi a rede GNU/Linux do IME - USP. Utilizou-se um computador AMD Athlon XP+1800, com 1 GB de memória RAM e o sistema operacional usado foi o Gentoo/Linux versão 2005.1.

[1] Ou seja, não fizemos o mapeamento dos dados do espaço de origem para outro espaço.

5.5 Dados usados

Para nossos experimentos, consideramos mensagens em inglês obtidas no site do projeto SpamAssassin [Sa06]. Tal site disponibiliza uma coleção de hams e spams para testes de eficiência de filtros de e-mails [Sapc06]. A escolha deste conjunto de dados deveu-se ao fato de seu grande uso tanto pela comunidade acadêmica quanto por empresas, além de ser uma base pública, o que permite que qualquer pessoa tenha acesso aos mesmos dados que nós utilizamos.

Na fase de treinamento, foram usados os arquivos 20030228_easy_ham_2.tar.bz2 e 20030228_spam.tar.bz2, que continham 1400 mensagens normais e 500 spams, respectivamente. Uma mensagem do arquivo 20030228_easy_ham_2.tar.bz2 não foi considerada devido ao fato de ter havido um erro durante a análise léxica de seu contéudo. Assim, a base de treinamento foi formada por 1899 mensagens, sendo 1399 hams e 500 spams.

Na fase de testes, foram considerados os arquivos 20030228_easy_ham.tar.bz2, 20050311_spam_2.tar.bz2 e 20030228_hard_ham.tar.bz2, que continham 2500 hams, 1396 spams e 250 hams, respectivamente. Duas mensagens do arquivo 20030228_easy_ham_2.tar.bz2 não foram consideradas pois houve problemas durante a análise léxica de seus textos. Assim, a base de testes é formada por 4144 mensagens, sendo 2498 hams considerados fáceis de classificar[2], 1396 spam e 250 hams considerados difíceis (maior chance de serem classificados incorretamente como spams). Mediremos a taxa de falsos-positivos separadamente para as mensagens dos arquivos 20030228_easy_ham.tar.bz2 e 20030228_hard_ham.tar.bz2. Esta escolha foi feita devido ao fato do segundo conjunto (hard_ham) ser mais difícil de ser classificado corretamente do que o primeiro (easy_ham). Desta forma, poderemos analisar melhor o desempenho de SVM's tanto em mensagens consideradas fáceis de classificar-se quanto em mensagens consideradas difíceis, de uma forma independente.

No próximo capítulo, apresentaremos os resultados dos testes realizados. Para simplificar nossa notação, chamaremos de easy_ham, easy_ham_2, hard_ham, spam e spam_2 o conjunto de mensagems existentes em cada um dos arquivos 20030228_easy_ham.tar.bz2, 20030228_easy_ham_2.tar.bz2, 20030228_hard_ham.tar.bz2, 20030228_spam.tar.bz2 e 20050311_spam_2.tar.bz2, respectivamente.

[2]Note que a classificação de um ham como *easy*(fácil) ou *hard*(difícil) foi feita pela equipe do projeto SpamAssassin.

Capítulo 6

Resultados obtidos e conclusões

6.1 Resultados

A partir das abordagens e considerações descritas no capítulo anterior, apresentaremos agora os resultados e conclusões obtidos.

1. Usando somente o corpo da mensagem, foi gerado um dicionário com 11653 palavras e os resultados foram os seguintes:

vetor de características	**taxa de erro (TE)**	**TFN**		**TFP**
		easy_ham	**hard_ham**	**spam_2**
Repr. binária	0.1347	0.1097	0.8160	0.0573
Freq. palavras	0.1055	0.0060	0.8400	0.1519

2. Usando o corpo e o assunto da mensagem, obtivemos um dicionário com 11770 palavras e os resultados estão apresentados abaixo:

vetor de características	**taxa de erro (TE)**	**TFN**		**TFP**
		easy_ham	**hard_ham**	**spam_2**
Repr. binária	0.1286	0.1023	0.8040	0.0530
Freq. palavras	0.1021	0.0068	0.8280	0.1426

6.2 Conclusões

Como podemos observar pelas tabelas acima, as máquinas de suporte vetorial que utilizam vetores de características com frequência de palavras apresentaram um desempenho ligeiramente superior ao das máquinas de suporte vetorial que utilizam vetores de características com representação binária na classificação de todas as mensagens (uma diferença de aproximadamente 3% tanto no uso de somente o corpo da mensagem quanto no uso do corpo combinado com o assunto).

Além disso, percebemos que o uso do assunto da mensagem combinado com o corpo oferece um melhora muito pequena no desempenho dos dois tipos de representação dos vetores de características das máquinas de suporte vetorial, sendo a diferença de cerca de 0.35% quando se utiliza freqüencia de palavras e de 0.5% quando se usa representação binária.

Entretanto, ao analisarmos especificamente cada grupo de mensagens (easy_ham, hard_ham e spam_2), observamos que SVM's que utilizam vetores de características com representação binária têm um desempenho muito superior para classificar corretamente mensagens como spams quando comparadas com SVM's que utilizam vetores de características com frequência de palavras. Este fato ocorre tanto quando se utiliza apenas o corpo da mensagem quando se utiliza o corpo combinado com o título, sendo uma diferença de aproxidamente 9% no primeiro caso e de 9.65% no segundo. Além disso, SVM's com vetores de características de representação binária também tiveram melhor desempenho na classificação de mensagens normais, mas que eram mais difíceis de serem classificadas corretamente (hard_ham). Tanto no uso somente do corpo da mensagem quanto no uso combinado do corpo com o título, a diferença foi de aproximadamente 2.4%. O baixo desempenho de todas as SVM's para este grupo de mensagens deveu-se ao fato deste conjunto ter pouca similaridade com o conjunto de treinamento, o que, claramente, torna mais difícil a classificação correta das mensagens. Por outro lado, SVM's com vetores de características que usam frequência de palavras resultou em um desempenho muito melhor na classificação correta de hams (easy_hams) do que SVM's com vetores de características representando freqüencias de palavras (cerca de 9.65% quando se usou apenas o corpo das mensagem e cerca de 10.35%

quando combinou o corpo com o assunto).

Portanto, máquinas de suporte vetorial que adotam freqüência de palavras nos vetores de características e usam tanto o corpo quanto o assunto das mensagens obtiveram o melhor desempenho geral e para o grupo easy_ham nos testes realizados. Por outro lado, máquinas de suporte vetorial que usaram vetores de características com representação binária, tiveram um desempenho bem melhor para classificar spams e hams "difíceis". Como, para o problema em questão, é mais importante ter uma baixa taxa de falsos positivos, o uso de máquinas de suporte vetorial com vetores de características que usam freqüencia de palavras talvez seja o mais adequado.Além disso, apesar do fato de muitas mensagens consideradas spams terem um extenso assunto e um corpo vazio ou com muito poucas palavras, o uso combinado dos dois campos não gerou uma melhoria siginificativa aos analisarmos as mensagens.

Como possíveis rumos das pesquisas destacamos: aumentar o tamanho da base de testes para confirmamos os resultados obtidos, modificarmos a largura de uma das margens do hiperplano separador (a do lado dos hams) ou aumentarmos a penalização para dados que violarem esta margem, de forma a reduzir a taxa de falsos positivos.

Parte II

Experiência pessoal

Capítulo 7

Experiência pessoal

7.1 Escolha do tema da iniciação

A escolha da área e do projeto de iniciação científica aconteceu em março deste ano. Tinha vontade de fazer uma iniciação científica desde o terceiro ano do BCC pois achava importante na complementação da formação oferecida no curso. Mas a grande quantidade de matérias do curso, meu gosto por várias áreas da computação, aliados às atividades extras que realizo e alguns problemas pessoais fizeram-me sempre adiar esta escolha.

Durante fevereiro e começo de março conversei com alguns professores do BCC para saber mais sobre as áreas que estudavam e os projetos de iniciação que ofereciam. Como fiz colégio técnico em Informática, gosto bastante da área de sistemas, mas desde o primeiro ano, também comecei a me interessar cada vez mais por áreas mais teóricas da computação. Assim, conversei com professores e alunos de pós-graduação de várias áreas.

Finalmente, escolhi a área de Otimização Contínua porque tinha gostado da área de Otimização (Contínua e Combinatória) quando cursei matérias relacionadas, como Programação Linear, Análise de Algoritmos e Algoritmos em Grafos. Também me interessei bastante pelo projeto que o professor Paulo J. S. Silva me apresentou porque ele tinha alguns aspectos de que eu gostava: uma grande base teórica para estudar, assuntos de duas áreas (Otimização Contínua e Aprendizado Computacional) e algumas aplicações reais importantes.

7.2 Desafios e frustrações encontrados

Inicialmente, o professor Paulo me indicou o livro de Cristianini e Shawe-Taylor [CST02] para aprender os conceitos básicos de máquinas de suporte vetorial. Os primeiros conceitos eram simples e de rápido entendimento, apresentando a teoria de aprendizado computacional. Fiz a implementação do algoritmo perceptron para entender algumas idéias envolvendo este tipo de classificador.

Em seguida, seguindo ainda [CST02], começamos a estudar kernels e mapeamento dos dados de um espaço para outro. Foi quando comecei a perceber o quanto eram poderosas as idéias por trás de máquinas de suporte vetorial (SVM), pois envolviam teorias de várias áreas da Matemática e da Computação, como Álgebra Linear (Computacional), Análise Funcional e Análise de Algoritmos. Quando meu orientador me apresentou o teorema de Mercer pela primeira vez, percebi que teria que revisar e aprender vários conceitos de Álgebra Linear e Análise Funcional para poder compreender bem tal teorema.

Depois de terminarmos o estudo inicial de kernels, ele sugeriu que utilizasse o tutorial de Burges [Bur98] como referência. Ao ler este tutorial, comecei a perceber como a diferença de estilos de escrever textos científicos pode influenciar muito na sua compreensão. O livro de Burges apresentava as idéias de máquinas de suporte vetorial com um outro enfoque, mais voltado à área de Otimização e com uma linguagem mais simples, o que facilitou o entendimento. Entretanto, meu orientador e eu percebemos que eu precisava aprofundar meus conhecimentos na área de Programação Não-Linear (PNL), pois boa parte da modelagem dos problemas de máquinas de suporte vetorial envolvia assuntos desta área. Como não tinha cursado a disciplina MAC0427 - Programação Não-linear, oferecida como optativa eletiva no primeiro semestre, tive que estudar os conceitos durante a iniciação.

Assim, durante as três semanas seguintes, estudei pelo livro de Friedlander [Fri05] para aprender os conceitos básicos de PNL. Logo após, para dirigir o estudo para problemas de Otimização Quadrática e Otimização Restrita, o professor Paulo sugeriu que passasse a estudar o livro de Nocedal e Wright [NW99], no qual me concentrei durante outras três semanas. Durante este período, meu interesse por Otimização Contínua aumentou bastante, pois apreciei bastante a teoria envolvida em otimização restrita. Para completar o estudo sobre PNL, estudei o livro de Bertsekas [Ber95] para entender a idéia de multiplicadores de Lagrange, que são usados para formular o problema

dual de SVM's.

Após este período estudando alguns conceitos de Otimização Contínua, voltei a ler o tutorial de Burges. Nesta releitura, entendi muitas idéias que não havia compreendido muito bem antes por falta de conceitos da área de Otimização e de Aprendizado Computacional. O fato de estar cursando a disciplina MAC5832 - Aprendizado Computacional como aluno especial da pós-graduação também foi fundamental para entender melhor vários conceitos envolvidos com máquinas de suporte vetorial, complementando assuntos vistos durante a iniciação.

A partir de setembro, comecei a me dedicar mais a escrever a monografia do trabalho de formatura, que era baseado nesta iniciação científica. Foi quando percebi o quanto é difícil escrever um bom texto científico. Toda semana, enviava uma versão para meu orientador ler e a cada reunião semanal que tínhamos, ele fazia correções em meu texto e me dava dicas de como poderia melhorá-lo. No início, tive certa dificuldade para organizar a estrutura do texto, mas conforme fui escrevendo e com o auxílio do meu orientador, a tarefa foi-se tornando mais fácil. Ao mesmo tempo, ao escrever a monografia pude aprofundar meus conhecimentos ou corrigir idéias equivocadas sobre o assunto, pois surgiram novas dúvidas que não haviam aparecido durante a fase anterior de estudos.

Paralelamente, continuei pesquisando sobre máquinas de suporte vetorial, estudando melhor os livros de Burges e Cristianini e Shawe-Taylor. A partir do final de outubro, começamos a pensar em realizar experimentos com máquinas de suporte vetorial para a detecção de spam. A base para os experimentos foi um artigo de Vapnik [HWV99], em que ele comparava SVM's com outras técnicas de reconhecimento de padrões para realizar esta tarefa. A descrição da metodologia usada por ele foi importante para sabermos como representaríamos um texto de maneira numérica para que pudesse ser tratado por uma máquina de suporte vetorial.

Os experimentos tinham 4 fases: montar o conjunto de dados (selecionar os e-mails), criar um analisador léxico (*parser*) para analisar cada mensagem, gerando o dicionário de palavras e os vetores de características, aprender como funciona a implementação SVM*light* e, por fim, fazer os testes e analisar a eficiência. Creio que as fases mais trabalhosas foram a criação do conjunto de dados e do analisador léxico, pois a primeira demandava muito tempo, analisando mensagens e a segunda tinham vários detalhes para serem ajustados.

Mas os resultados dos experimentos foram bem animadores. Nos pri-

meiros testes com um conjunto de treinamento de 371 mensagens normais e 360 spams e um conjunto de testes de 99 spams e 82 mensagens normais, obtive uma eficiência de 100% usando o corpo e o cabeçalho na análise e de aproximadamente 98% usando apenas o corpo, utilizando a representação binária de características. Isso me estimulou, pois percebi que as máquinas de suporte vetorial realmente tinham uma boa eficiência para o problema estudado.

Acredito que uma das maiores dificuldades encontradas durante a iniciação foi o pouco tempo existente para estudar uma grande quantidade de assuntos de várias áreas: Matemática Pura, Estatística, Análise de Algoritmos e, principalmente, de Otimização Contínua e Aprendizado Computacional, muitos deles bastante complexos, como Análise Funcional. Outra dificuldade foi conciliar a iniciação com o BCC e as demais atividades que realizo, como trabalho e curso de francês. Percebi que na iniciação é importante ter uma grande disciplina e organização, para manter um bom ritmo regular de estudo e, ao mesmo, ter tempo para realizar outras atividades.

Acho que deveria ter começado a iniciação científica antes, no início do terceiro ano do bacharelado em Ciência da Computação. Desta forma, eu poderia ter estudado melhor vários assuntos vistos, principalmente da área de algoritmos de Otimização Contínua e técnicas de reconhecimentos de padrões. Além disso, poderia ter realizado um número maior de experimentos e estudado outros assuntos relacionados.

7.3 Disciplinas do BCC mais relevantes

Diversas disciplinas do bacharelado em Ciência da Computação foram importantes tanto na iniciação quanto na minha formação profissional.

Destaco MAC0122 - Princípios de Desenvolvimento de Algoritmos, porque foi a primeira disciplina em que realmente há uma preocupação com a "qualidade" de um algoritmo (seu consumo de tempo e memória) e serve de base para todas as outras disciplinas da área de Computação. A partir desta disciplina, comecei a perceber (algumas vezes nas provas) a grande diferença entre um algoritmo funcionar e um algoritmo funcionar bem (de forma eficiente). Quando tinha feito colégio técnico em Informática, geralmente a preocupação maior era que o algoritmo funcionasse corretamente, sem avaliar seu desempenho. Quando cursei MAC0122, percebi o quanto é importante fazer um bom algoritmo. Além disso, MAC0338 - Análise de Algoritmos foi

fundamental no curso, pois estende e aprofunda muitos conceitos vistos em disciplinas anteriores, como MAC0122 e MAC0323 - Estrutura de Dados, e também apresenta muitas idéias usadas em várias áreas da Computação.

Além das disciplinas citadas acima, as seguintes disciplinas foram muito importantes para entender os conceitos vistos durante a iniciação:

- **MAC0315 - Programação Linear:** esta disciplina introduziu as idéias básicas importantes tanto em Otimização Contínua quanto Combinatória e permitiu que eu entendesse os conceitos estudados de Programação Não-Linear.

- **MAC5832 - Aprendizado Computacional:** forneceu ou aprofundou alguns dos conceitos importantes vistos durante a iniciação.

- **MAC0300 - Métodos Numéricos de Álgebra Linear:** esta disciplina apresentou vários conceitos e técnicas usados em Álgebra Linear Computacional, os quais foram úteis para entender as formas de resolução de problemas de Programação Não-linear envolvidos com máquinas de suporte vetorial.

- **MAC0242 - Laboratório de Programação II:** foi nesta disciplina que tive meu primeiro contato com a linguagem Python, quando aprendi linguagens de scripts. Isto foi muito importante para desenvolver os programas que faziam as análises da mensagens e geravam o dicionário e os vetores de características.

- **MAT0121 - Cálculo Diferencial e Integral II:** esta disciplina apresentou muitas das idéias importantes na área de Otimização Contínua, como, por exemplo, máximos e mínimos de funções e matrizes Hessianas.

- **MAT0139 - Álgebra Linear para Computação:** introduziu muitos conceitos importantes não apenas na área de Otimização, mas em várias áreas da Computação.

- **MAE0121 - Introdução à Probabilidade e à Estatística I e MAE0212 - Introdução à Probabilidade e à Estatística II:** tais disciplinas foram importantes para entender alguns conceitos existentes na área de Aprendizado Estatístico, envolvendo as suposições feitas sobre a distribuição dos dados analisados.

Além destas disciplinas, considero muito importantes para minha formação geral como cientista da Computação e para a minha decisão de fazer um mestrado no próximo ano, as seguintes disciplinas:

- MAC0328 - Algoritmos em Grafos
- MAC0438 - Programação Concorrente
- MAC5781 - Otimização Combinatória (feita como aluno especial do mestrado)
- MAC0322 - Desafios de Programação
- MAC0453 - Princípios de Pesquisa Operacional e Logística
- MAC0413 - Tópicos de Programação Orientada a Objetos
- MAC0422 - Sistemas Operacionais
- MAC0426 - Sistemas de Banco de Dados
- MAT0213 - Álgebra II
- MAT0221 - Cálculo Diferencial e Integral IV
- MAE0228 - Noções de Probabilidade e Processos Estocásticos

De forma geral, considero que todas as disciplinas obrigatórias do currículo foram úteis para tornar minha formação bastante ampla. Procurei cursar disciplinas optativas de diversas áreas da Computação para diversificar minha formação e também para ajudar a conhecê-las melhor e decidir em quais eu tinha mais interesse. Apenas acho que não aproveitei completamente algumas disciplinas cursadas, principalmente devido ao pouco tempo que podia dedicar a algumas delas e também à forma que algumas foram oferecidas.

Além disso, o fato de ter participado da Olímpiada Brasileira de Informática quando estava no primeiro ano da graduação e das maratonas de programação do IME e da SBC durante todo o curso, melhorou muito minha capacidade de resolver problemas usando idéias vistas em algumas disciplinas.

7.4 Interação com o orientador e a forma de trabalho na iniciação

Durante quase toda a iniciação, tinha reuniões semanais com o meu orientador em dias e horários previamente marcados. Nestas reuniões, geralmente, o professor Paulo esclarecia algumas dúvidas que tinha sobre o assunto estudado durante a semana e, muitas vezes, mostrava uma outra interpretação ou demonstração dos mesmos. Além disso, fazia explicações mais detalhadas, semelhantes a aulas, para explicar assuntos mais complexos, como a demonstração do Teorema de Mercer e alguns conceitos da área de Otimização Restrita. Ele também se dedicava bastante à iniciação, estudando paralelamente os assuntos que estava vendo.

No início da iniciação, ele escolhia os assuntos que deveria estudar durante a semana e qual seria a melhor referência. Isto foi muito importante, pois cada livro ou artigo abordava o assunto de uma forma diferente, sendo mais adequado para certos tópicos. Com o desenvolvimento da iniciação, fui tendo mais autonomia para escolher quais assuntos estudar, o que foi importante para desenvolver minha habilidade de pesquisa. Neste período, minhas visitas à biblioteca do IME tornaram-se ainda mais freqüentes para escolher novas referências.

Cada reunião tinha uma duração de geralmente uma hora. Quando um tópico mais complexo havia sido estudado durante a semana, as reuniões chegavam a durar aproximadamente duas horas e meia. Nas reuniões, aprendi a não pensar somente de uma única forma sobre um assunto, procurando fazer outras interpretações e relacionar com outros conteúdos. Acredito que isto será fundamental durante o meu mestrado e também na minha vida profissional, tanto se eu seguir a carreira acadêmica quanto se eu for para o mercado.

Meu orientador também foi importante para escolher as disciplinas que estou cursando como aluno especial de mestrado no segundo semestre, explicando como elas poderiam ser úteis no mestrado. Também no segundo semestre, quando comecei a escrever esta monografia, ele teve papel fundamental, lendo cada versão e fazendo sugestões e correções. O seu rigor matemático foi útil para melhorar a qualidade do texto, eliminando erros e possíveis ambigüidades.

Ademais, ele teve influência em algumas decisões profissionais que tomei no segundo semestre como, por exemplo, quando decidi trabalhar fora

do IME. Ele acabou apresentando bons argumentos para que eu não trabalhasse tanto para não me sobrecarregar e, conseqüentemente, atrapalhar meu desempenho na graduação e na iniciação.

Por fim, a experiência do professor Paulo como programador também ajudou na fase de testes, através da sugestão de pacotes para o desenvolvimento dos programas que analisavam os e-mails.

Por tais motivos, acho que ele foi um bom orientador e, por isso, ele também será meu orientador durante o mestrado.

7.5 Aplicações dos conceitos vistos no curso e na iniciação em aplicações reais

Os conceitos vistos no curso foram muito importantes tanto na iniciação científica quanto em algumas atividades paralelas que desenvolvi durante a graduação. Desde maio de 2005, sou um dos administradores da rede CEC/Linux, e os conhecimentos adquiridos em diversas disciplinas do BCC possibilitaram-me adquirir um bom conhecimento da área de administração de sistemas. Além disso, trabalho atualmente com desenvolvimento de sistemas e utilizo alguns dos conceitos teóricos estudados.

Os conceitos vistos na iniciação foram aplicados em uma situação prática: a detecção de spam e os bons resultados animaram-me a continuar pesquisando.

7.6 O futuro

No próximo ano, devo iniciar o mestrado em Ciência da Computação, na área de Otimização Contínua, em um tema não relacionado diretamente com os tópicos vistos durante a iniciação, mas cuja escolha foi muito influenciada pelo estudo de máquinas de suporte vetorial.

Além disso, pretendo estudar paralelamente máquinas de suporte vetorial aplicadas na detecção de spam, dedicando algumas horas livres para melhorar os resultados obtidos e ampliar os testes para mensagens em outros idiomas.

7.7 Conclusões

Considero muito importante ter feito iniciação científica durante a graduação, pois acredito que complementou minha formação como cientista da Computação e foi muito importante para a escolha da minha área do mestrado. Durante a iniciação, aprendi alguns assuntos não vistos durante o curso e aprofundei outros, além de ter me acostumado a ler artigos científicos, procurando entender a idéia que os autores estão tentando transmitir. E, principalmente, desenvolvi minha habilidade de pesquisa, ao estudar diversas referências para entender um conteúdo maior, o que deverá ser muito importante durante o mestrado e na minha vida pessoal e profissional.

O bacharelado em Ciência da Computação me deu uma grande base sobre várias áreas fundamentais da Computação, possibilitando-me atuar tanto no meio acadêmico quanto no mercado. Como ainda tenho dúvidas sobre qual dos dois segmentos vou atuar, acho importante ter esta base que me permite analisar de forma crítica vários assuntos da área.

Por fim, agradeço aos meus pais, professores e amigos por todas as contribuições prestadas durante a graduação.

Referências Bibliográficas

[AB92] M. Anthony and N. Biggs. *Computational Learning Theory.* Cambridge University Press, 1992.

[Ber95] D. P. Bertsekas. *Nonlinear programming.* Athena Scientific, 1995.

[Bur98] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery,* 2:121-167, 1998.

[Cer05] CERT. http://www.nbso.nic.br, Outubro de 2005.

[CST02] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based methods.* Cambridge University Press, 2002.

[DHS01] R. O. Duda, P. E. Hart and D. G. Stork. *Pattern classification.* Wiley, 2001

[Fer05] Ferris Research. http://www.ferris.com, Outubro de 2005.

[Fri05] A. Friedlander. *Elementos de Programação Não-linear.* http://www.ime.unicamp.br/ friedlan/livro.htm

[Gma05] Gmail. http://www.gmail.com. Dezembro, 2005.

[HWV99] H. Druker, D. Wu, V. N. Vapnik. Support Vector Machines for Spam Categorization. *IEEE Transactions on Neural Networks,* 10(5):1048-1054, 1999.

[Joa99] T. Joachims. Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.).* MIT-Press, 1999.

[NW99] Jorge Nocedal, Stephen J. Wright. *Numerical optimization.* Springer, 1999

[Pyt05] Python. http://www.python.org. Dezembro de 2005.

[RLI05] Rede Linux IME - USP. http://www.linux.ime.usp.br. Dezembro de 2005.

[Sa06] Projeto SpamAssassin. http://spamassassin.apache.org. Fevereiro de 2006.

[Sapc06] Coleção de e-mails do projeto SpamAssassin. http://spamassassin.apache.org/publiccorpus. Fevereiro de 2006.

[Scna05] SpamCop.net. http://www.spamcop.net, Outubro de 2005.

[Sfr05] Spam Filter Review. http://spam-filter-review.toptenreviews.com, Outubro de 2005.

[Svml05] SVM*light*. http://svmlight.joachims.org, Dezembro de 2005.

[SVN05] Support Vector Machines - The Book. http://www.support-vector.net. Dezembro, 2005.

[Vap98] V. N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.

[Wat91] D. S. Watkins. *Fundamentals of Matrix Computations.* John Wiley & Sons, 1991.