Universidade Estadual de Campinas

Projeto Supervisionado

O Perceptron e as Máquinas de Vetores de Suporte

José Dié Viegas

Orientado por
Prof. Dr. Marcos Eduardo do Valle

27 de Junho de 2017

Resumo

Na atualidade, sistemas de automação e programas baseados em inteligência artificial vem ganhado bastante notoriedade devido à superação de antigos paradigmas e limitações da computação tradicional e também às novas possibilidades concebidas pelos mesmos. Da previsão de *default* em sistemas de risco de crédito, ao reconhecimento de padrões no comportamento dos consumidores de uma empresa, aplicações baseadas em *Machine learning* permeiam cada dia mais o cotidiano

Este trabalho expolora aspectos teóricos e computacionais de modelos relacionados à inteligência articial, em particular as redes neurais artificiais (RNAs) e as máquinas de vetores de suporte (do inglês, *support vector machines* ou SVMs apenas). Para tanto, analisam-se conceitos como o mecanismo de funcionamento de um neurônio biológico como inspiração para um modelo artificial.

Buscou-se aqui estabelecer uma análise detalhada de cada modelo em observância com suas possíveis aplicações e limitações. De forma a melhor entender os modelos sugeridos, implementou-se computacionalmente os algoritmos descritos nesta monografia[1] em exemplos concretos na Linguagem MATLAB.

[1] Códigos, parâmetros de entrada e resultados podem ser vistos nos apêndices da monografia.

Abstract

Nowadays, automation systems and programs based on artificial intelligence (AI) have gained a lot of notoriety due to the overcoming of old paradigms and limitations of traditional computing and also the new possibilities conceived by them. From the prediction of default in credit risk systems, to the recognition of patterns in the behavior of the consumers of a company, applications based on Machine learning permeate every day more our everyday life.

This paper explores theoretical and computational aspects of models related to artificial intelligence, in particular artificial neural networks (ANNs) and support vector machines (SVMs only). In order to do so, we analyze concepts such as the mechanism of functioning of a biological neuron as an inspiration for an artificial model.

We sought to establish a detailed analysis of each model in compliance with its possible applications and limitations. In order to better understand the suggested models, the algorithms described in this monograph have been implemented computationally[2] in concrete examples in the MATLAB Language.

[2]Codes, input parameters and results can be seen in the appendices of the monograph.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Motivação para modelos artificais baseados no sistema biológico

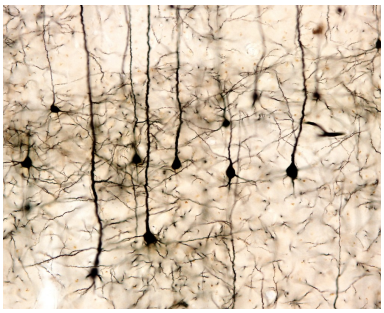
1.1 O neurônio biológico

A ideia de se conceber um modelo computacional inspirado no sistema nervoso não é recente. Redes neurais biológicas são de grande complexidade e possuem alta capacidade de reconhecimento de padrões, além de desempenhar outras importantes funções, como representação do ambiente, memorização e o acionamento de atividades motoras. Neste sentido, é esperado que pesquisadores se voltem para o entendimento deste mecanismo biológico e suas possiblidades de replicação em sistemas artificiais. A unidade biológica que torna todo este potencial possível é o neurônio (vide Figura 1.1). Estima-se que o cerébro humano contenha a ordem de 10^{10} destas unidades básicas. Cada um destes neurônios opera em paralelo comunicando-se com- aproximadamente- 10^4 outros neurônios.

O processamento em paralelo é uma das grandes vantagens de um sistema nervoso biológico, uma vez que o mesmo permite um modelo de computação- relativamente- veloz e a realização de tarefas de alta complexidade.

1.1.1 Estrutura do Neurônio

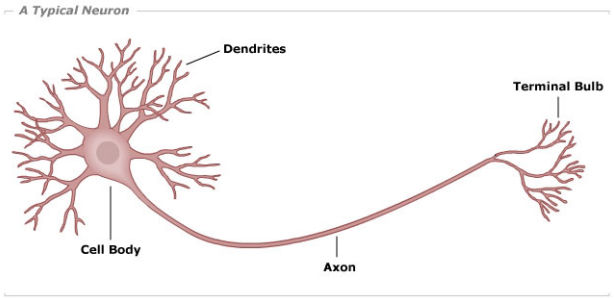
A Figura (1.2) representa um neurônio. Pode-se observar que o mesmo é composto de três principais segmentos: os dendritos, o corpo celular e o axônio. A principal função dos dendritos é captar os sinais de outros neurônios atrravés de junções denominadas sinapses neurais. É no corpo celular que ocorre- de fato- o processamento dos sinais recebidos dos dendritos. Como veremos,



Fonte: Foto de José Luis Calvo/Shutterstock.com

Figura 1.1: Neurônios do cortex cerebral visualizados por microscópio.

um potencial de ativação indicará se ocorre, ou não, a propagação de um impulso elétrico em direção ao axônio, o qual irá transmitir os tais impulsos para outros neurônios. Existem ainda os neurônios efetuadores, os quais possuem conexão direta com o tecido muscular.



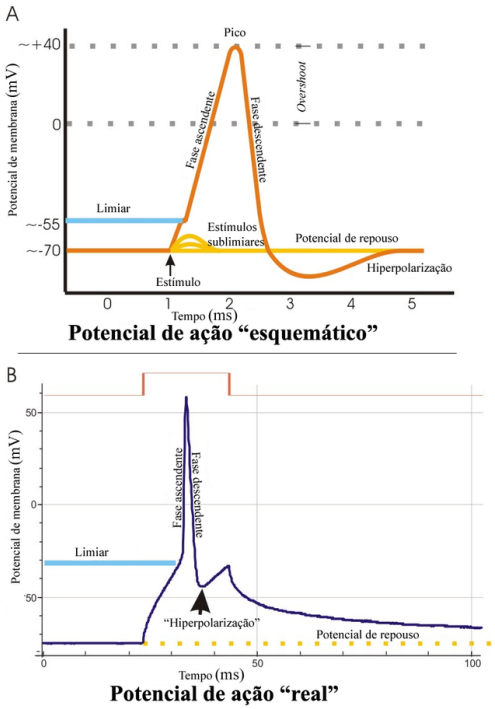
Fonte: https://online.science.psu.edu/bisc004_activewd001/node/1907

Figura 1.2: Gravura de um neurônio com indicação de seus principais componentes.

Sabe-se que o número de sinapses no ser humano não é constante. Pesquisas indicam que a densidade das mesmas é relativamente maior durante o período que compreende a fase embrionária até o início da puberdade, momento no qual começa a se notar a diminuição de tais conexões. Ainda sobre sinapses, destaca-se que o contato físico entre cada neurônio inexiste. De fato, a propagação de sinais é devida a elementos denominados de **neurotransmissores**. Desta forma, inferimos que os neurotransmissores desempenhma um papel de intermédio entre diferentes unidades nervosas.

1.1.2 Excitação de um Neurônio Biológico

Quando em repouso, a membrana neural está polarizada negativamente em sua região interna, ou seja, existe uma maior quantidade de íons negativos nesta região quando comparada ao meio exterior. O estímulo de uma célula nervosa (despolarização) ocorre por meio da transferência de íons positivos (Na^+ e K^+) para o interior da membrana. Tal mecanismo se dá pela atuação da Bomba de Sódio e Potássio. Caso tal despolarização ultrapasse um limiar, ocorre um disparo de um impulso elétrico em direção ao axônio. Tal limiar de ativação é aproximadamente $-55 \quad mV$ e o potencial de ação máximo é aproximadamente $35 \quad mV$. Na Figura (1.3) podemos ver as fases envolvidas durante a excitação do neurônio considerando as variações de potencial elétrico.

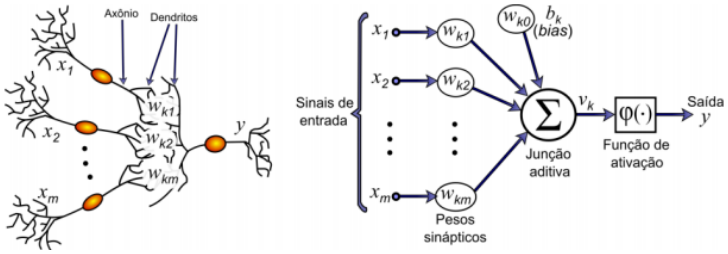


Fonte: https://pt.wikipedia.org/wiki/Potencial_de_a%C3%A7%C3%A3o

Figura 1.3: Etapas de Variação de Potencial de Ação.

1.2 O neurônio Artificial

Baseado em análises anteriores como, por exemplo, as realizadas por Hodgkin&Huxley (1952), o modelo percursor no que tange às RNAs foi proposto por McCulloch&Pits (1943). No funcionamento deste neurônio, o conjunto de informações advindas do meio externo (entradas) são somadas e ponderadas



Fonte: [Haykin, 1999, Oliveira, 2014]

Figura 1.4: Neurônio Artificial.

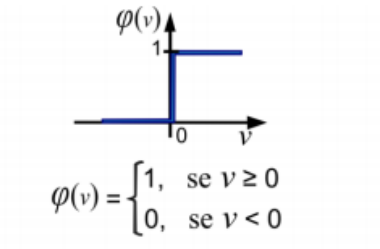
por pesos sinápticos. Após o processamento desta soma por uma função de ativação, caso o resultado supere um limiar (específico de cada neurônio), a saída resultante é não nula. A figura a seguir exemplifica a ideia por trás de um neurônio artificial. Observe que o combinador linear é representado na mesma pelo sinal de somatório $\sum$.

1.2.1 Tipos de Função de Ativação

Em geral, considarmos dois principais tipos de função de ativação. São elas as parcialmente diferenciáveis e as funções totalmente diferenciáveis. As funções de ativação dependem de qual regra de aprendizado está sendo considerada para o treinamento da rede. Neste sentido, nos modelos cuja regra de aprendizado depende de algum tipo de otmização, é comum o uso de funções totalmente diferenciáveis, uma vez que descontinuidades poderiam ser incovenientes para o cálculo de derivadas. Em outros casos, a regra de aprendizado não demanda de de nenhum tipo de otimização. Para estes, comumente vê-se o uso de funções com descontinuidades (parcialmente diferenciáveis). Este é o caso do Perceptron, por exemplo (função degrau).

Dos exemplos abaixo, apenas o primeiro caso (função de Heavyside) consiste numa função parcialmente diferenciável.

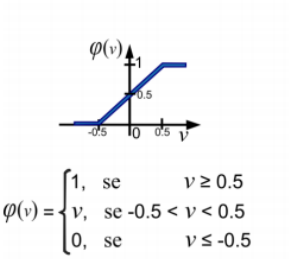
1. **Função de Limiar ou Degrau(Heavyside)**



Fonte: [Haykin, 1999, Oliveira, 2014]

Figura 1.5: Função Degrau.

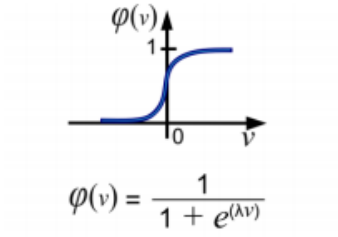
2. **Função Limiar por partes**



Fonte: [Haykin, 1999, Oliveira, 2014]

Figura 1.6: Função Limiar por partes.

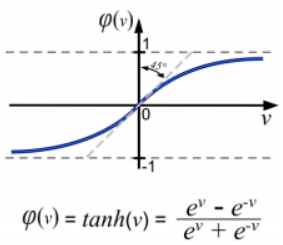
3. **Função Sigmóide**



Fonte: [Haykin, 1999, Oliveira, 2014]

Figura 1.7: Função sigmóide.

4. **Função Tangente Hiperbólica**



Fonte: [Haykin, 1999, Oliveira, 2014]

Figura 1.8: Função tangente hiperbólica.

1.2.2 Arquitetura de Redes Neurais

As redes neurais podem estar arranjadas de múltiplas maneiras. Denominamos de arquitetura a maneira em que os neurônios estãos dispostos uns em relação aos outros. Tais conformações são determinantes para se extrair o comportamento dos outputs da rede.

As redes neurais apresentam 3 principais partes:

1. Camada de entrada: Consiste na camada na qual são fornecidos as entradas externos, ou seja, trata-se da camada que recebe primariamente as medições. Denominamos cada conjunto de entrada como amostra.

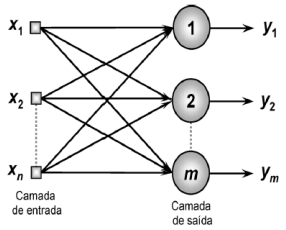
2. Camadas Escondida (*Hidden Layers*): Tais camadas aparecem em modelos com mais de um neurônio. As mesmas atuam extraindo características do sistema analisado.

3. Camadas de Saída: As camadas de saída são também constituídas por neurônios. É nestas onde se verfica a apresentação dos resultados produzidos pela rede. Tais outputs são geralmente associados a classes.

Quando os resultados de uma camada dependem apenas de entradas de camadas anteriores, dizemos que a rede é ***feedfoward*** ("alimentação à frente"), as quais podem ser de uma única camada ou de múltiplas camadas.

Arquitetura feedfoward de camada única:

Neste tipo de arquitetura verifica-se apenas uma camada de entrada, assim como uma camada de neurônios, a qual é a própria camada de saída.

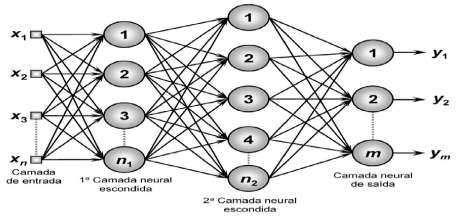


Fonte: [Silva et al., 2010]

Figura 1.9: Rede feedfoward de camada única.

Arquitetura feedfoward de múltiplas camadas:

Neste tipo de rede neural existem camadas de neurônios escondidas. Cada camada pode possuir uma quantidade distinta de neurônios. De fato, é comum que a quantidade de neurônio por camada não seja exatamente a mesma. Os principais tipos de redes com arquitetura *feedfoward* de múltilpas camadas são o Perceptron multicamadas (*multlayer Perceptron-MLP*) e as redes de base radial (*radial basis function -RBF*).

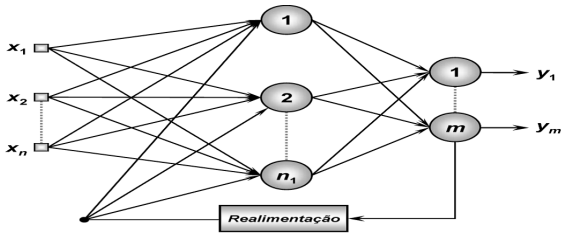


Fonte: [Silva et al., 2010]

Figura 1.10: Rede feedfoward Multicamdas.

Arquitetura recorrente ou realimentada:

Nestes tipos de rede, as saídas de uma camada podem servir como entradas para camadas anteriores. Como principal representante desta arquitura, tem-se a rede Hopfield e a rede Perceptron com realimentação. O interessante deste tipo de arquitetura é que saídas atuais dependem das saídas anteriores. Sua aplicação é notada em séries temporais, uma vez que tal tipo de arquitetura é bastante usado para estudar sistemas com variação temporal.

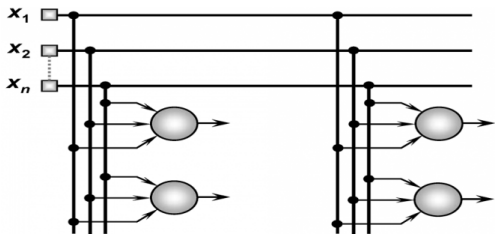


Fonte: [Silva et al., 2010]

Figura 1.11: Rede recorrente.

Arquitetura Reticulada:

Neste tipo de rede, sinais de entrada podem servir como entradas para diversos neurônios de outras camadas da rede. O exemplo mais famoso desta arquitetura é a de rede de Kohoen. O uso das mesmas é notável em problemas com grafos e otimização de sistemas.



Fonte: [Silva et al., 2010]

Figura 1.12: Rede reticulada.

1.3 Paradigma de Aprendizagem

A uma rede neural, a capacidade de aprender é fundamental. No âmbito da mesma, o "aprendizado"se dá no sentido de realizar uma tarefa de forma a sempre produzir melhoria em seu desempenho. A mesma o faz provocando mudanças em seus parâmetros internos a medida em que a assertividade em suas tarefas aumenta. Denominamos de **Algoritmos de Treinamento** um conjunto bem definido de regras usadas por uma rede neural para encontrar os parâmetros internos que satisfaçam de maneira correta (ou mais otimizada o possível) as tarefas propostas à rede.

Definimos como **Paradigma de Aprendizagem** a maneira pela qual se dá a influência no aprendizado pelo ambiente externo. Os principais tipos de Paradigmas são:

1. **Aprendizagem Supervisionada:**

   Neste tipo de aprendizagem, existe um supervisor que compara os resultados da rede com aqueles previstos de acordo com entradas fornecidas a mesma.

2. **Aprendizagem não Supervisionada ou auto-organizada:**

   Neste tipo de aprendizagem, não existe um supervisor que compare os resultados da rede com os previstos de acordo com as entradas. Neste caso, dizemos que os dados são *não-rotulados*, uma vez que as classes das entradas são previamente desconhecidas. É, portanto, função do algoritmo a descoberta de padrões e regularidades. Redes que seguem esta arquitetura são usadas, por exemplo, na construção de agrupamentos (*Clusters*).

Existem ainda outros paradigmas de aprendizagem, tal qual a aprendizagem por reforço, entretanto, como nos modelos a seguir será apenas explorado a aprendizagem supervisionada, ficando-se restrito aqui apenas à menção da existência das mesmas.

Capítulo 2

O Perceptron de Rosenblatt

Para iniciar o estudo do modelo do Perceptron, vamos usar como referência o diagrama apresentado na Figura (2.1). É possível notar que existem m entradas nesta rede e um limiar (*bias*) designado por b ou como no diagrama: w_0. Ocorre que o sinal de ativação u decorre da soma ponderada entre os pesos sinápticos w_i pelas entradas x_i adicionada ao valor do limiar da rede Perceptron, tipicamente negativo. Ou seja, a saída do neurônio é obtida calculando-se a função degrau bipolar na ativação. Em termos matemáticos, temos que:

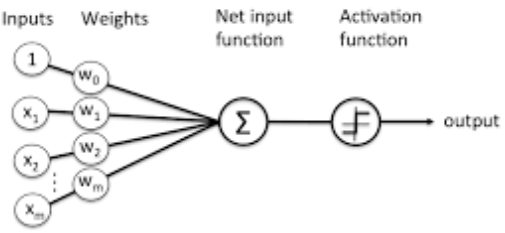
$$u = \sum_{i=1}^{m} x_i.w_i - \quad b \tag{2.1}$$

$$y = \varphi(u) \tag{2.2}$$

$$y = \begin{cases} -1, & \text{se} \quad u \leq 0, \\ 1, & \text{se} \quad u > 0 \end{cases} \tag{2.3}$$

A função de ativação $\varphi(u)$ usada na saída acima é a degrau bipolar[1]. O que o resultado de (3) nos diz é que, dado um conjunto de m entradas, o Perceptron irá gerar apenas uma saída com duas possibilidades. Isto nos permite inferir que tal modelo- ainda que com vasta aplicabilidade- limita-se a classificar somente 2 classes distintas. De fato, ver-se-á que o funcionamento correto do modelo Perceptron de Rosenblatt pressupõe a existência de duas classes linearmente seperáveis.

[1] A função degrau canônica também é usada no modelo do Perceptron simples



Schematic of Rosenblatt's perceptron.

Fonte: http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Figura 2.1: Neurônio Artificial Perceptron de Rosenblatt.

É conveniente usarmos a notação matricial para representarmos os parâmetros deste modelo. Considere- portanto- as seguintes matrizes:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & ... & x_{1\mathrm{m}} \\ 1 & x_{21} & x_{22} & ... & x_{2\mathrm{m}} \\ \vdots & \vdots & & ... & \vdots \\ 1 & x_{\mathrm{p}1} & x_{\mathrm{p}2} & ... & x_{\mathrm{pm}} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

O elemento x_{ij} representa a entrada j da i-ésima amostra. Desta forma, as linhas da matriz $\mathbf{X}$ designam os conjuntos de entradas da rede, ou- de maneira equivalente- as amostras.

Daqui por diante, representaremos a k-ésima linha de $\mathbf{X}$ em sua forma transposta como $\mathbf{x}^k$.

Note que, para algum k fixo, $\mathbf{w}^T.\mathbf{x} = 0$ define um hiperplano m-dimensional como uma fronteira de decisão para duas classes distintas $(C1, C2)$, acessadas de acordo com as entradas fornecidas à rede. Evidente que tal classificação está intimamente ligada aos parâmetros da rede, isto é, aos elementos de $\mathbf{w} = (w_0, w_1, \ldots, w_m)$.

Definição 1. *A classe de um vetor arbitrário $\boldsymbol{x}$ com m colunas é dada por:*

$$Classe(\boldsymbol{x}) = \begin{cases} C_1, & se \quad u > 0 \\ C_2, & se \quad u \leq 0, \end{cases}$$

$u = \boldsymbol{w}^T.\boldsymbol{x}$

Definição 2. *Seja $\boldsymbol{T}$ uma matriz denominada Matriz de Treinamento com estrutura análoga à matriz $\boldsymbol{X}$ e com a seguinte propriedade: para $\boldsymbol{t}$ vetor linha transposto da*

matriz de treinamento, tem-se que N linhas de $\boldsymbol{T}$ *satisfazem* $\boldsymbol{w}^T.\boldsymbol{t} > 0$ *e as* $P - N$ *linhas restantes satisfazem* $\boldsymbol{w}^T.\boldsymbol{t} \leq 0$.

Definição 3. *H_1 é o conjunto dos vetores de treinamento $\boldsymbol{t} \in C_1$. H_2 é o conjunto dos vetores de treinamento $\boldsymbol{t} \in C_2$.*

2.1 O treinamento para classificação

2.1.1 Problemas de classificação

Dado um conjunto χ $\{ (x_i, y_i) \in \mathbf{X} \times \mathbf{Y} : i = 1, \ldots, P\}$, onde $\mathbf{Y}$ é um conjunto finito de rótulos de classe dado por $\mathbf{Y} = \{l_1, l_2, \ldots, l_L\}$ e $\mathbf{X}$ um universo arbitrário, o objetivo de um problema de classificação é determinar uma função $f : \mathbf{X} \rightarrow \mathbf{Y}$, tal que $f(x_i) = y_i$, para $\forall i = 1, \ldots, P$.

2.1.2 Classificação no contexto do Perceptron

As linhas da matriz de treinamento $\mathbf{T}$ são vetores, cujas classes são **previamente conhecidas**. A ideia por trás de se treinar uma rede é justamente encontrar os parâmetros necessários (vetor de pesos), de forma que as amostras de treinamento sejam classificadas da maneira esperada.

Assumindo $\mathbf{X}$ agora como uma matriz de treinamento, nosso objetivo é encontrar um vetor de pesos $\mathbf{w}$ que satisfaça simultaneamente as duas equações:

$$\mathbf{w}^T.\mathbf{x} > 0, \text{ para os } N \text{ vetores } \mathbf{x} \in H_1. \tag{2.4}$$

$$\mathbf{w}^T.\mathbf{x} \leq 0, \text{ para os } P - N \text{ vetores } \mathbf{x} \in H_2. \tag{2.5}$$

Desta forma, é intuitivo que deva haver algum processo de atualização em $\mathbf{w}$ até que todas as amostras sejam classificadas corretamente. As etapas de treinamento da rede são denominadas *épocas*.

O algoritmo abaixo[2] nos dá uma ideia do tipo de comportamento que esperamos em relação à atualização do vetor de pesos em realação a uma amostra específica $\mathbf{x}^k$:

[2]Não será usada tal estruturação ao apresentarmos o algoritmo definitivo e sua implementação computacional. A mesma foi aqui descrita apenas no sentido de explicitar de maneira mais didática qual o comportamento esparado para atualização do vetor de pesos num problema de classificação durante a fase de treinamento. De fato, veremos que é possível integrar diferentes condições deste algoritmo numa mesma etapa.

Begin
float X; /*matriz de treinamento*/
float $\mathbf{w}_k$; /* Vetor de pesos*/
if $(((\boldsymbol{x}^k \in H_1) \quad e \quad (\boldsymbol{w}^T{}_k.\boldsymbol{x}^k > 0)) \quad ou \quad ((\boldsymbol{x}^k \in H_2) \quad e \quad (\boldsymbol{w}^T{}_k.\boldsymbol{x}^k \leq 0)))$ **then**
  $\mathbf{w}_{k+1} = \mathbf{w}_k$;
else
  if $((\boldsymbol{x}^k \in H_2) \quad e \quad (\boldsymbol{w}^T{}_k.\boldsymbol{x}^k > 0))$ **then**
    $\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \quad \mathbf{x}^k$; /*onde $\eta \in [0,1]$*/
  else
    if $((\boldsymbol{x}^k \in H_1) \quad e \quad (\boldsymbol{w}^T{}_k.\boldsymbol{x}^k \leq 0))$ **then**
      $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta \quad \mathbf{x}^k$; /*onde $\eta \in [0,1]$*/
    end
  end
end

A ideia por trás do algoritmo é a seguinte: caso as entradas da rede (linhas da matriz de treinamento $\mathbf{x}$) gerem uma saída identificada corretamente com a classe esperada- então, não se deve pertubar o vetor de pesos $\mathbf{w}_k$.

Caso contrário, existem duas possiblidades: o vetor de teste $\mathbf{x}^k \in H_1$, isto é, para algum $\mathbf{w}^*$[3], $\mathbf{w}^{*T}.\mathbf{x}^k > 0$ e a rede- entretanto- produz uma saída negativa, classificada- portanto- erroneamente em C_2. Neste caso, deve-se atualizar o vetor de pesos aumentando a contribuição do mesmo no produto $\mathbf{w}^T{}_k.\mathbf{x}^k$ proporcionalmente ao vetor de teste $\mathbf{x}^k$.

A outra possibilidade é $\mathbf{x}^k \in H_2$, ou seja, para algum $\mathbf{w}^*$, $\mathbf{w}^*.\mathbf{x}^k \leq 0$ e a rede- contudo- prover uma saída positiva. Neste outro caso, deve-se atualizar o vetor de pesos diminuindo contribuição do mesmo no produto $\mathbf{w}^T{}_k.\mathbf{x}^k$ proporcionalmente ao vetor de teste $\mathbf{x}^k$.

Em suma, o Perceptron deve testar (atualizando o vetor de pesos quando necessário) todas as amostras da matriz de treinamento de maneira que todas estas passem a estar identificadas corretamente em suas classes.

2.2 O algoritmo Perceptron

Definição 4. *Seja um conjunto $H^\dagger$ dado por:*

$$H^\dagger = H_1 \cup (-H_2) \tag{2.6}$$

Tomando os vetores de teste pertencentes ao conjunto $H^\dagger$, torna-se possível simplificar o algoritmo de atualização do vetores de pesos: ora, como estamos considerando todos $\mathbf{x}^k \in H^\dagger$, basta verificar se $((\mathbf{x}^k \in H^\dagger) \quad e \quad (\mathbf{w}_k.\mathbf{x}^k > 0))$. Portanto, o algoritmo simplificado do Perceptron de Rosenblatt é dado por:

[3] $\mathbf{w}^*$ corresponde ao vetor de pesos que satisfaz simultaneamente as equações (2.5) e (2.6).

Algoritmo Simplificado do Perceptron

Begin
float X; /* matriz de treinamento*/
float $\mathbf{w}_k$; /* Vetor de pesos*/
Bool $erro = True$;
while *(erro = True)* **do**
  $erro = False$;
  for *(k = 1; k <= P; k++)* **do**
    if $((\boldsymbol{x}^k \in H^{\dagger}) \quad e \quad (\boldsymbol{w}^T{}_k.\boldsymbol{x}^k > 0))$ **then**
      $\mathbf{w}_{k+1} = \mathbf{w}_k$;
    else
      $erro = True$;
      $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta \quad \mathbf{x}^k$; /*onde $\eta \in [0,1]$*/
    end
  end
end

Algorithm 1: Algoritmo de Treinamento do Perceptron

Evidentemente não há qualquer tipo de perda de generalidade em se proceder da maneira acima. Se o hiperplano ótimo fosse definido por uma reta atravessando a origem, ao tomar todos os $\mathbf{x}^k \in H^{\dagger}$, estaríamos considerando as amostras negativas refletidas em relação ao hiperplano que cruza a origem.

2.3 Convergência do Perceptron

Definição 5. *Seja χ_P uma dicotomia (partição binária) de P amostras, tal que cada amostras se relacione exclusivamente com uma classe:*

$$\chi_P = \{(\boldsymbol{x}^1, l_1), (\boldsymbol{x}^2, l_2), (\boldsymbol{x}^3, l_3), \quad . \quad . \quad . \quad , (\boldsymbol{x}^i, l_i), \quad . \quad . \quad . \quad , (\boldsymbol{x}^P, l_P)\} \tag{2.7}$$

$\boldsymbol{x}^i \in \mathfrak{Re}^m, i = 1, ..., P$
$l_i \in \{-1, 1\}$, $l_i = sinal(\boldsymbol{w}^T.\boldsymbol{x}^i)$.

Teorema 1. *Se χ_P descreve um dicotomia linearmente separável, então o algoritmo do Perceptron converge após um número finito de atualizações no vetor de pesos.*

A seguir, apresenta-se a prova teorema acima:

Demonstração. Vamos tomar os vetores de treinamento $\mathbf{x}^{\mathrm{k}} \in H^{\dagger}$. Como vimos anteriormente, isso implica que é possível usar o algoritmo de treinamento simplificado. Consideremos agora um vetor $\mathbf{w}^*$ unitário, o qual satisfaz as equações(2.5) e (2.6), isto é, o mesmo produz um hiperplano que separa os outputs das amostras de acordo com os valores de saída esperados. Além disto, tomemos um $\delta \in \mathfrak{Re}^+$ que satisfaz a seginte desigualdade:

$$\mathbf{w}^{*T}.\mathbf{x} > \delta \tag{2.8}$$

Seja a seguinte igualdade:

$$g(\mathbf{w}) = \frac{\mathbf{w}^*.\mathbf{w}}{|\mathbf{w}|} \tag{2.9}$$

Como supomos $|\mathbf{w}^*| = 1$, então $g(\mathbf{w}) = cos(\mathbf{w}^*, \mathbf{w})$. Disto, podemos afirmar que:

$$g(\mathbf{w}) \leq 1 \tag{2.10}$$

Seguindo os passo do algoritmo de atualização de pesos para um $\eta = 1$ durante a t-ésima época, vale a igualdade:

$$\mathbf{w}_{t+1} = (\mathbf{w}_t + \mathbf{x}^t) \tag{2.11}$$

Multiplicando escalarmente a expressão acima por $\mathbf{w}^*$, obtemos:

$$\begin{aligned} \mathbf{w}^{*T}.\mathbf{w}_{t+1} &= \mathbf{w}^{*T}.(\mathbf{w}_t + \mathbf{x}^t) && (2.12)\\ &= \mathbf{w}^{*T}.\mathbf{w}_t + \mathbf{w}^{*T}.\mathbf{x}^t && (2.13)\\ &\geq \mathbf{w}^{*T}.\mathbf{w}_t + \delta && (2.14) \end{aligned}$$

Observe que (2.14) se justifica devido a equação (2.10). Ora, após n iterações, uma vez que $\mathbf{x}^k \in H^+$ e vale o algoritmo simplificado de treinamento, deve valer a seguinte desigualdade:

$$\mathbf{w}^{*T}.\mathbf{w}_n \geq n\delta \tag{2.15}$$

A inequação acima é signficativa, pois nos permite extrair uma ideia do comportamento do numerador de $\frac{\mathbf{w}^*.\mathbf{w}}{|\mathbf{w}|}$. De maneira análoga, seria interessante se pudéssemos inferir algum tipo de ideia sobre o denominador desta razão. De fato;

$$\begin{aligned} |\mathbf{w}_{t+1}|^2 &= (\mathbf{w}_t + \mathbf{x}^t) \quad . \quad (\mathbf{w}_t + \mathbf{x}^t) && (2.16)\\ &= |\mathbf{w}_t|^2 + |\mathbf{x}^t|^2 + 2.\mathbf{w}_t.\mathbf{x}^t && (2.17) \end{aligned}$$

O produto $\mathbf{w}^T.\mathbf{x}$ deve ser negativo (vide conjunto de vetores de treinamento assumido e algoritmo implementado). Para um conjunto de amostras de treinamento normalizadas[4], tem-se então:

$$|\mathbf{w}_{t+1}|^2 < 1 \tag{2.18}$$

[4]É comum normalizamos as amostras a serem fornecidas à rede, pois efeitos de magnitude podem ocasionar falhas durante o treinamento, uma vez que as entradas podem ter ordens de grandeza expressivamente distintas.

Após a *n-ésima* atualização do vetor de pesos, teremos portanto:

$$|\mathbf{w}_n|^2 < n \tag{2.19}$$

Concluímos de (2.15) e (2.19) portanto que:

$$g(\mathbf{w}_n) = \frac{\mathbf{w}^*.\mathbf{w}_n}{|\mathbf{w}_n|} \tag{2.20}$$

$$\frac{n.\delta}{\sqrt{n}} \tag{2.21}$$

Entretanto, vimos que $g(\mathbf{w}) \leq 1$. Com esta fato, por fim podemos escrever:

$$\sqrt{n}.\delta \leq 1 \tag{2.22}$$

$$n \leq \frac{1}{\delta^2} \tag{2.23}$$

A equação acima diz que é possível tomarmos $\delta > 0$ tão pequeno quanto queiramos e- ainda assim- após n etapas de atualização, o algoritmo de treinamento irá convergir. Ainda mais, o número de atualizações terá a limitação $n \leq \frac{1}{\delta^2}$ □

2.4 O Problema do ou-exclusivo (XOR)

Como já havíamos discutido, o Perceptron é um algoritmo extremamente útil no caso em que as classes são linearmente separáveis. Uma das causas que desencorajou a pesquisa em redes neurais causando um hiato histórico neste campo do conhecimento foi tal limitação[5] . O exemplo canônico para ilustrar tal fato é o chamado problema do "ou-exclusivo". Para entender a natureza do problema, vamos inicialmente considerar a figura abaixo:

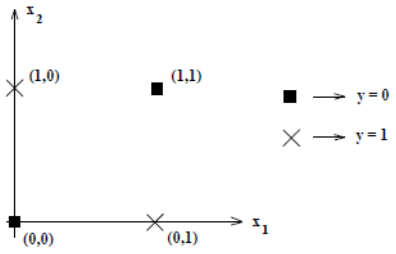


Figura 2.2: Classes não-linearmente separáveis

[5]O problema do ou-exclusivo foi resolvido a partir de redes neurais Multicamadas. Tal abordagem despertou novamente interesse no estudo de redes neurais artificiais, promovendo assim grande avanço neste campo de pesquisa.

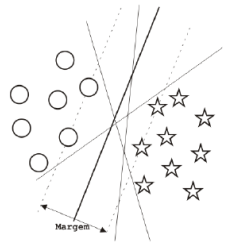
É fácil perceber a partir da figura acima a impossibilidade de se traçar um hiperplano que separe as duas classes existentes. Com isto, uma rede neural como o Perceptron não consegue separar as classes. Jamais haverá convergência do algoritmo para as amostras de classes não-linearmente separáveis.

Capítulo 3

Máquinas de Vetores de Suporte

3.1 Introdução às SVM

Para iniciar o estudo das máquinas de vetores de suporte (*Support Vector Machines* ou simplesmente *SVM*), faz-se a priori o uso da motivação gráfica por trás do método. Considere a figura abaixo:



Fonte: [Rufino, 2011]

Figura 3.1: Conjunto de hiperplanos separadores

Observando o gráfico, podemos perceber que- apesar de todas as linhas cheias separarem corretamente pontos de classes diferentes, apenas a mais escura das mesmas tem a propriedade de estar mais o distante dos pontos mais próximos de cada classe. Desta maneira, é de ser se esperar que um hiperplano ótimo seja um separador linear que maximize a distância entre si e os pontos mais próximos de cada classe.

Consideremos uma dicotomia binária com P amostras. Além disto, diferente da metodologia empregada anteriormente nas redes neurais artificiais, na qual era conveniente escrevermos o vetor de pesos incorporando o poten-

cial de ativação da rede b (*bias*), vamos reconsiderar $\mathbf{w}$ como:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

Como havíamos descrito anteriormente, a equação $\mathbf{w}^T.\mathbf{x}^k + b = 0$ descreve um hiperplano. Um fato importante de se observar é que a equação anterior é invariante por multiplicação por um escalar α qualquer. Isto nos permite reescalonar w e b de forma que a seguinte equação seja válida:

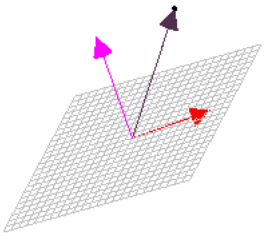
$$min_{(\mathbf{x},\mathrm{d})} \quad |\mathbf{w}^T.\mathbf{x} + b| = 1 \tag{3.1}$$

Um dos conceitos chaves no entendimento das máquinas de vetores de suporte é o conceito da distância de um ponto a um hiperlano. Em $\Re\mathfrak{e}^3$ para um ponto P_0 contido num plano Π e um ponto aleatório P_1, temos a conhecida equação:

$$dist(P_1, \Pi) = \frac{\overrightarrow{P_0P_1} \quad . \quad \hat{\mathbf{n}}}{|\hat{n}|} \tag{3.2}$$

$$= \frac{|ax + by + cz + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{3.3}$$

, onde a, b e c são as coordenadas do vetor normal ($\hat{\mathbf{n}}$).



Fonte: http://www.mat.ufmg.br/gaal/aulas_online/at4_06.html

Figura 3.2: Distância entre plano e ponto

Generalizando este conceito para $\Re\mathfrak{e}^m$, podemos perceber que a distância de um vetor arbitrário $\mathbf{x}$ a um hiperplano é dado por:

$$dist(\mathbf{x}, hiperplano) = \frac{|\mathbf{w}^T.\mathbf{x} + b|}{||\mathbf{w}||} \tag{3.4}$$

Definição 6. *Denotamos por **margem** (ρ) a menor distância entre os pontos mais próximos de cada classe ao hiperplano separador.*

Dada a definição acima, considerando a suposiçao de que $\min_{\mathbf{x},d} \quad |\mathbf{w}^T.\mathbf{x}+b| = 1$ e a equação da distância entre um ponto e um hiperplano, concluímos que:

$$\rho = \min_{\mathbf{x},d} \quad \frac{|\mathbf{w}^T.\mathbf{x}+b|}{||\mathbf{w}||} \tag{3.5}$$

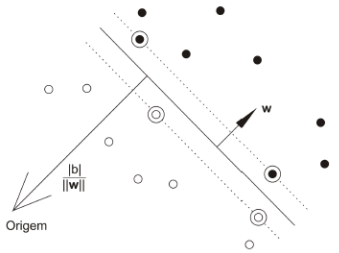
$$= \frac{1}{||\mathbf{w}||} \tag{3.6}$$

Observe que ρ implica que $\mathbf{w}^T.\mathbf{x}+b$ admita apenas duas soluções:

$$\mathbf{w}^T.\mathbf{x}+b = \begin{cases} -1 \\ 1 \end{cases} \tag{3.7}$$

Assim, pode-se concluir que um hiperplano denifido por $(\bar{\mathbf{w}}, b)$ classifica corretamente as amostras de treinamento $\mathbf{x}_i$ quando $\mathbf{w}^T.\mathbf{x}+b$ possui o mesmo sinal de d_i. Como, por coveniência, havíamos reescalonado o par $(\bar{\mathbf{w}}, b)$, temos então que toda amostra deve satisfazer a seguinte inequação:

$$d_i.(\mathbf{w}^T.\mathbf{x}+b) \geq 1 \tag{3.8}$$



Fonte: [Rufino, 2011]

Figura 3.3: Hiperplano de Separação

Intuitivamente, a ideia de usar o conceito de distância no problema de classificação aparece. Observando a figura acima, poderíamos pensar numa regra de decisão que mensura-se a possiblidade da amostra $\mathbf{x}_i$ estar à direita (ou não) do hiperplano separador e- caso esta primeira opção fosse satsifeita- classificase-ia tal amostra como pertencente a uma classe específica, digamos $\mathbf{C}_1$. Em linguagem matemática, isto se traduz em verificar se $\mathbf{w}^T.\mathbf{x} \geq K$, com $k \in \mathfrak{Re}$. Em

particular, podemos tomar $K = -b$, de forma que $\mathbf{w}^T.\mathbf{x} + b \geq 0$. Não tendo esta última condição satisfeita, automaticamente teríamos que a amostra pertenceria à $\mathbf{C}_2$. Vamos considerar que se $\mathbf{x}_i \in \mathbf{C}_1$, então $d_i = +1$. Em contrapartida, se $\mathbf{x}_i \in \mathbf{C}_2$, então $d_i = -1$

Definição 7. *Defini-se com vetor de suporte o ponto que que dista ρ do hiperplano separador e que está contido em um hiperplano marginal com vetor normal paralelo a aquele que caracteriza o hiperplano separador de classes.*

Para os vetores de suporte da figura (3.3), temos que a inequação (3.8) é satisfeita por igualdade. Como vimos no início do capítulo, a nossa ideia é maximizar a margem de forma a obter uma maior generalização. Vamos agora tomar dois vetores de suporte de classes distintas. Isto implica que cada um destes está sobre um hiperplano marginal: um à direita do hiperplano separador, outro à esquerda (vide figura 3.3). Ora, dados estes dois vetores de suporte (digamos, $\overrightarrow{\mathbf{x}_1} \in \mathbf{C}_2$ e $\overrightarrow{\mathbf{x}_2} \in \mathbf{C}_1$), uma vez que cada hiperplano marginal dista de ρ do hiperplano separador, maximizar a margem implica em maximizar a componente paralela ao vetor normal do hiperplano separador da diferença entre $\overrightarrow{\mathbf{x}_2}$ e $\overrightarrow{\mathbf{x}_1}$. Isto é;

$$max \quad 2.\rho = max \quad (\overrightarrow{\mathbf{x}_2} - \overrightarrow{\mathbf{x}_1}).\frac{\mathbf{w}}{\|\mathbf{w}\|} \tag{3.9}$$

$$max \quad 2.\rho = max \quad \frac{(\overrightarrow{\mathbf{x}_2}.\mathbf{w} - \overrightarrow{\mathbf{x}_1}.\mathbf{w})}{\|\mathbf{w}\|} \tag{3.10}$$

Sabemos, entretanto, que para os vetores de suporte vale $d_i.(\mathbf{w}^T.\mathbf{x} + b) = 1$. Além disto, usando a convenção de sinal de d_i em observância com as classes destes mesmos vetores, temos que:

$$+1.(\mathbf{w}^T.\mathbf{x}_2 + b) = 1 \tag{3.11}$$

$$\mathbf{w}^T.\mathbf{x}_2 = 1 - b \tag{3.12}$$

$$\mathbf{x}_2.\mathbf{w} = 1 - b \tag{3.13}$$

Também;

$$-1.(\mathbf{w}^T.\mathbf{x}_1 + b) = 1 \tag{3.14}$$

$$(-1).(\mathbf{w}^T.\mathbf{x}_1) = 1 + b \tag{3.15}$$

$$(-1).(\mathbf{x}_1.\mathbf{w}) = 1 + b \tag{3.16}$$

Usando (3.13) e (3.16) em (3.10), obtemos:

$$max \quad 2.\rho = max \quad \frac{(1-b)+(1+b)}{||\mathbf{w}||} \tag{3.17}$$

$$max \quad 2.\rho = max \quad \frac{2}{||\mathbf{w}||} \tag{3.18}$$

3.2 Otimização Quadrática

A útima equação sugere que para encontrar o hiperplano ótimo devemos maximizar a seguinte função objetivo: $\frac{2}{||\mathbf{w}||}$. Sem perda de generalidade, visando a conveniência nos cálculos, maximizar a margem é análogo a minimizar a seguinte quantidade:

$$\begin{aligned} &\min_{\mathbf{w}} && \Phi = \frac{1}{2}.\mathbf{w}^T.\mathbf{w} \\ &\text{sujeito a} && d_i.(\mathbf{w}^T.\mathbf{x}_i + b) = 1, \; para \quad i = 1, \ldots, P. \end{aligned}$$

Como é sabido acerca de otimzação[1], podemos escrever uma função lagrangiana da seguinte forma:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\alpha}, b) = \frac{1}{2}\mathbf{w}^T.\mathbf{w} - \sum_{i=1}^{P} \alpha_i [d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1] \tag{3.19}$$

, onde os α_i são os multiplicadores de Lagrange.

As condições de primeira ordem afirmam que $\nabla\mathcal{L} = 0$. Portanto;

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0, i = 1, ..., P \tag{3.20}$$

$$0 = \frac{\partial}{\partial \mathbf{w}}\{\frac{1}{2}\mathbf{w}^T.\mathbf{w} - \sum_{i=1}^{P} \alpha_i [d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1]\} \tag{3.21}$$

$$0 = \mathbf{w} - \frac{\partial}{\partial \mathbf{w}}\{\sum_{i=1}^{P} \alpha_i [d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1]\} \tag{3.22}$$

$$0 = \mathbf{w} - \sum_{i=1}^{P} \alpha_i d_i \mathbf{x}_i \tag{3.23}$$

Portanto:

$$\mathbf{w} = \sum_{i=1}^{P} \alpha_i d_i \mathbf{x}_i \tag{3.24}$$

[1] Para detalhes acerca do tema, ver Bertsekas [1999].

Observe que- na prática- podemos derivar em relação a um vetor de forma similar a um escalar.

Além de $\mathbf{w}$, temos que b também varia. Portanto, devemos calcular a derivada parcial do lagrangiano também em relação a esta variável. Com isto:

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \tag{3.25}$$

$$0 = \frac{\partial}{\partial b}\{\frac{1}{2}\mathbf{w}^T.\mathbf{w} - \sum_{i=1}^{P} \alpha_i[d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1]\} \tag{3.26}$$

$$0 = -1\frac{\partial}{\partial b}\sum_{i=1}^{P} \alpha_i[d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1] \tag{3.27}$$

$$\sum_{i=1}^{P} \alpha_i d_i = 0 \tag{3.28}$$

As equações (3.24) e (3.28) representam- portanto- as condições necessárias para se computar os parâmetros do hiperplano ótimo definido por $(\mathbf{w}, b)$.

3.2.1 O problema Dual

Uma vez que obtivemos uma expressão para $\mathbf{w}$, é possível usar o resultado de (3.24) na expressão do lagrangiano. Proceder desta maneira nos leva ao problema de otimização dual.

$$\mathcal{L}(\mathbf{w},\boldsymbol{\alpha},b) = \frac{1}{2}\mathbf{w}^T.\mathbf{w} - \sum_{i=1}^{P}\alpha_i[d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1] \tag{3.29}$$

$$\mathcal{L}(\mathbf{w},\boldsymbol{\alpha},b) = \frac{1}{2}\|\sum_{i=1}^{P} d_i\mathbf{x}_i\alpha_i\|^2 - \{\sum_{i=1}^{P}\alpha_i d_i\mathbf{w}^T.\mathbf{x}_i + \sum_{i=1}^{P}\alpha_i d_i b - \sum_{i=1}^{P}\alpha_i\} \tag{3.30}$$

$$\mathcal{L}(\mathbf{w},\boldsymbol{\alpha},b) = \frac{1}{2}\|\sum_{i=1}^{P} d_i\mathbf{x}_i\alpha_i\|^2 - \{\sum_{i=1}^{P}\alpha_i d_i\mathbf{w}^T.\mathbf{x}_i + b\underbrace{\sum_{i=1}^{P}\alpha_i d_i}_{0} - \sum_{i=1}^{P}\alpha_i\} \tag{3.31}$$

$$\mathcal{L}(\mathbf{w},\boldsymbol{\alpha},b) = \frac{1}{2}\|\sum_{i=1}^{P} d_i\mathbf{x}_i\alpha_i\|^2 - \{\sum_{i=1}^{P}\alpha_i d_i\mathbf{w}^T.\mathbf{x}_i - \sum_{i=1}^{P}\alpha_i\} \tag{3.32}$$

$$= \frac{1}{2}\{\sum_{i=1}^{P} d_i\mathbf{x}_i\alpha_i \sum_{j=1}^{P} d_j\mathbf{x}_j\alpha_j\} - \sum_{i=1}^{P}\alpha_i d_i\mathbf{w}^T.\mathbf{x}_i + \sum_{i=1}^{P}\alpha_i \tag{3.33}$$

$$= \frac{1}{2}\{\sum_{i=1}^{P} d_i\mathbf{x}_i\alpha_i \sum_{j=1}^{P} d_j\mathbf{x}_j\alpha_j\} - \sum_{i=1}^{P}\alpha_i d_i.\mathbf{x}_i \sum_{j=1}^{P} d_j\mathbf{x}_j\alpha_j + \sum_{i=1}^{P}\alpha_i \tag{3.34}$$

$$= \frac{1}{2}\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j\alpha_i\alpha_j(\mathbf{x}_i.\mathbf{x}_j)\} - 1\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j\alpha_i\alpha_j(\mathbf{x}_i.\mathbf{x}_j)\} + \sum_{i=1}^{P}\alpha_i \tag{3.35}$$

Chegamos finalmente ao seguinte problema de otimização quadrática:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & Q(\boldsymbol{\alpha}) = \sum_{i=1}^{P}\alpha_i - \frac{1}{2}\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j\alpha_i\alpha_j(\mathbf{x}_i.\mathbf{x}_j)\} \\ \text{sujeito a} \quad & \alpha_i \geq 0, \quad para \quad i = 1,2,3,\ldots \\ & \sum_{i=1}^{P}\alpha_i.y_i = 0 \end{aligned}$$

Ora, obtendo os multiplicadores de Lagrange (valores denotados por $\alpha_{0,i}$), é possível determinar vetor normal do hiperplano ótimo, aqui denominado de $\mathbf{w}_0$. Portanto:

$$\mathbf{w}_0 = \sum_{i=1}^{P} d_i.\alpha_{0,i}\mathbf{x}_i \tag{3.36}$$

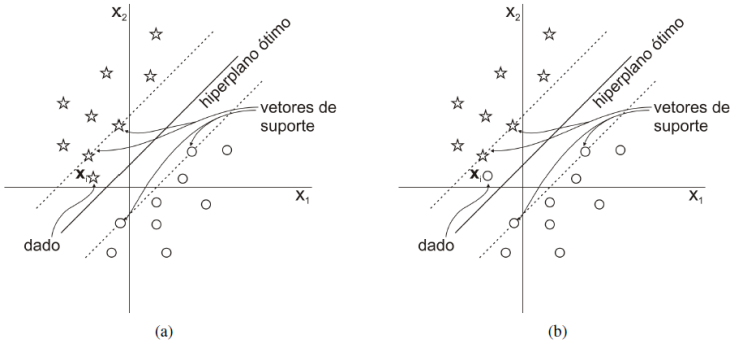
Uma vez encontrado $\mathbf{w}_0$, basta tomar um vetor de suporte $\mathbf{x}_i \in \mathbf{C}_1$ para encontrarmos o parâmetro "b"do hiperplano ótimo:

$$b_0 = 1 - \mathbf{w}_0{}^T.\mathbf{x}_i \tag{3.37}$$

3.3 Classes não-linearmente Separáveis

Até então, os métodos explorados nesta monografia (Perceptron e SVMs) apresentaram-se como soluções para obtenção classificadores lineares. Como já havíamos destacado, entretanto, apeasar destes possuírem grande aplicabilidade em vários contextos, nem sempre os probelmas estudados apresentam como característica intrísceca a separabilidade linear das amostras.

Neste contexto de classes não linearmente separáveis, seria interessante a busca de um hiperplano que minimiza-se a probabilidade de erro de classificação. Para tanto, vamos no âmbito das SVM considerar quando tais problemas de classifcação podem ocorrer. Para exemplificar, observemos as imagens a seguir:



Fonte: [Haykin, 1999, Rufino, 2011]

Figura 3.4: Violações da Margem

1. Na figura (3.4 - a), temos que uma das amostras está classificada na região correta de classificação. Entretanto, a mesma está na região interna, isto é, entre os hiperplanos marginais, a qual- por construção- não deveria conter qualquer vetor.

2. Já na figura (3.4 - b), temos que uma das amostras está classificada na região incorreta.

De outra forma, poderíamos dizer que tais pontos satisfazem a relação abaixo:

$$d_i(\mathbf{w}.\mathbf{x} + b) \ngeq 1 \tag{3.38}$$

A solução proposta por Cortes e Vapnik (1995) foi considerar variáveis de folga ξ_i não nulas para se mensurar os erros em pontos mal classificados. Assim, a nova condição a ser satisfeita é:

$$d_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, P \tag{3.39}$$

Observe ainda que:

1. Para $0 < \xi_i \leq 1$, temos que os pontos são classificados corretamente, isto é, de acordo com seus rótulos. Entretanto, tais vetores localizam-se dentro da região de separação.
2. Para $\xi_i = 0$, temos que não há qualquer violação do tipo (3.39).
3. Para $\xi_i > 1$, temos que os pontos são classficados na classe errada.

3.3.1 Problema Primal

É matematicamente conveniente definirmos uma função $\Phi^{'}$ dada por:

$$\Phi^{'} = C.\sum_{i=1}^{P} \xi_i \tag{3.40}$$

O parâmetro C representa o quanto estamos penalizando o erro de classificação. Em geral, tal parâmetro é definido experimentalmente pelo usuário ou de acordo com alguma heurística.

Desta maneira, é razoável que se queira agora minimizar a quantidade $\Phi^{'} + \Phi$. Isto nos leva ao seguinte problema de otimização:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}} \quad & \Phi + \Phi^{'} = \frac{1}{2}.\mathbf{w}^T.\mathbf{w} + C.\sum_{i=1}^{P} \xi_i \\ \text{sujeito a} \quad & d_i.(\mathbf{w}^T.\mathbf{x}_i + b) = 1 - \xi_i; \quad para \quad i = 1,\ldots,P \\ & e \quad \xi_i \geq 0; para \quad i = 1,\ldots,P. \end{aligned}$$

Podemos finalmente definir uma nova função lagrangiana. Observe:

$$\mathcal{L}(\mathbf{w},\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{\mu}) = \frac{1}{2}\mathbf{w}^T.\mathbf{w} + C.\sum_{i=1}^{P} \xi_i - \sum_{i=1}^{P} \alpha_i[d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{P} \mu_i\xi_i \tag{3.41}$$

O último termo do lagrangiano acima se relaciona com a não negatividade dos ξ_i.

Procedendo como nas vezes anteriores, vamos usar as condições de KKT[2] para o lagrangiano. Entretanto, vale notar que:

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Longrightarrow \quad \sum_{i=1}^{P} \alpha_i d_i = 0 \tag{3.42}$$

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0, i = 1,...,P \Longrightarrow \quad 0 = \mathbf{w} - \sum_{i=1}^{P} \alpha_i d_i \mathbf{x}_i \tag{3.43}$$

[2]Condições de Karuch-Kuhn-Tucker.

Resta- portanto- computarmos $\frac{\partial \mathcal{L}}{\partial \xi_k} = 0$:

$$\frac{\partial \mathcal{L}}{\partial \xi_k} = \frac{\partial}{\partial \xi_k}[\frac{1}{2}\mathbf{w}^T.\mathbf{w} + C\sum_{i=1}^{P}\xi_i - \sum_{i=1}^{P}\alpha_i[d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{P}\mu_i\xi_i] \quad (3.44)$$

$$= -\sum_{i=1}^{P}\alpha_i\frac{\partial}{\partial \xi_k}[d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{P}\frac{\partial}{\partial \xi_k}(\mu_i\xi_i) + C\sum_{i=1}^{P}\frac{\partial}{\partial \xi_k}(\xi_i) \quad (3.45)$$

$$= -\sum_{i=1}^{P}\delta_{i,k}\alpha_i - \sum_{i=1}^{P}\mu_i\delta_{i,k} + \delta_{i,k}C \quad (3.46)$$

$$0 = \alpha_k - \mu_k + C \quad (3.47)$$

Todas as condições do problema primal são dadas então por:

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Longrightarrow \quad \sum_{i=1}^{P}\alpha_i d_i = 0 \quad (3.48)$$

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0, i = 1,...,P \Longrightarrow \quad 0 = \mathbf{w} - \sum_{i=1}^{P}\alpha_i d_i \mathbf{x}_i \quad (3.49)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_k} = 0 \Longrightarrow 0 = \alpha_i - \mu_i + C \quad (3.50)$$

$$d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1 + \xi_i \geq 0, \quad para \quad i = 1,\ldots,P \quad (3.51)$$

$$\alpha_i[d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1 + \xi_i] = 0, \quad para \quad i = 1,\ldots,P \quad (3.52)$$

$$\xi_i \geq 0, \quad para \quad i = 1,\ldots,P \quad (3.53)$$

$$\mu_i \geq 0, \quad para \quad i = 1,\ldots,P \quad (3.54)$$

$$\alpha_i \geq 0, \quad para \quad i = 1,\ldots,P \quad (3.55)$$

$$\mu_i.\xi_i \geq 0, \quad para \quad i = 1,\ldots,P \quad (3.56)$$

3.3.2 O Problema Dual para classes não-linearmente separáveis

À semlhança do problema das SVMs para classes linearmente separáveis, vamos usar (3.49) (3.50) na função lagrangiana (3.44). Isto nos leva ao problema dual:

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & Q(\boldsymbol{\alpha}) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2}\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j \alpha_i \alpha_j (\mathbf{x}_i.\mathbf{x}_j)\} \\
\text{sujeito a} \quad & 0 \leq \alpha_i \leq C, \quad para \quad i = 1, 2, 3, \ldots \\
& \sum_{i=1}^{P} \alpha_i . d_i = 0
\end{aligned}$$

Para deduzir a restrição adicional sobre os valores dos multiplicadores de Lagrange α_i, consideremos a igualdade (3.54) e a desigualdade (3.50):

$$C = \mu_i + \alpha_i \tag{3.57}$$
$$\mu_i \geq 0 \tag{3.58}$$
$$C - \alpha_i = \mu_i \tag{3.59}$$
$$C - \alpha_i \geq 0 \tag{3.60}$$

Portanto;

$$0 \leq \alpha_i \leq C \tag{3.61}$$

Capítulo 4

Métodos Kernel

4.1 Motivação

Relembrando o problema dual das máquinas de vetores de suporte, tínhamos que:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & Q(\boldsymbol{\alpha}) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2}\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j \alpha_i \alpha_j (\mathbf{x}_i.\mathbf{x}_j)\} \\ \text{sujeito a} \quad & \alpha_i \geq 0, \quad para \quad i = 1, 2, 3, \ldots \\ & \sum_{i=1}^{P} \alpha_i.d_i = 0 \end{aligned}$$

Poderíamos nos perguntar sobre o que ocorre para as SVM's cujos os vetores estejam contidos no espaço imagem (chamado também de espaço de características) de uma tranformação $\boldsymbol{\varphi}$ definida da maneira a seguir:

$$z = \boldsymbol{\varphi}(\mathbf{x}) : \Re\mathfrak{e}^m \rightarrow \Re\mathfrak{e}^n \tag{4.1}$$

$$\boldsymbol{\varphi} = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_m(\mathbf{x})]^T \tag{4.2}$$

, com $\varphi_0(\mathbf{x}) = 1$ para todo $\mathbf{x}$.

Ou seja, estamos buscando o hiperplano ótimo no espaço de características defnido por:

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0 \tag{4.3}$$

Desta forma, devemos esperar que nossa função lagrangiana fique definida

por:

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2}\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j \alpha_i \alpha_j (\boldsymbol{\varphi}(\mathbf{x}_i).\boldsymbol{\varphi}(\mathbf{x}_j))\} \tag{4.4}$$

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2}\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j \alpha_i \alpha_j < \mathbf{z}_i, \mathbf{z}_j >\} \tag{4.5}$$

Por outro lado, adaptando (3.36) para o novo espaço, temos que:

$$\mathbf{w} = \sum_{i=1}^{P} \alpha_i d_i \boldsymbol{\varphi}(\mathbf{x}_i) \tag{4.6}$$

Substituindo (4.6) em (4.3), vê-se que:

$$\sum_{i=1}^{P} \alpha_i d_i \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}_i) = 0 \tag{4.7}$$

Observe que tanto na função objetivo, quanto nas restrições do problema, o produto interno $< \mathbf{z}_i, \mathbf{z}_j >$ aparece. Assim, vê-se que não é necessário o conhecimento da transformação $\boldsymbol{\varphi}(\mathbf{x})$, com $\mathbf{x} \in \Re\varepsilon^m$, basta termos uma equação $\kappa(\mathbf{x}_i, \mathbf{x}_j) =< \varphi(\mathbf{x}_i).\varphi(\mathbf{x}_j) >$ para resolvermos o problema de otimização do lagrangiano para vetores do espaço de características.

4.1.1 Redução da complexidade Computacional

Consideremos agora uma função $\kappa^{'}$ definida por:

$$\kappa^{'} = (1+ < (\varphi(\mathbf{x}), \varphi(\mathbf{x}^{'}) >)2 \tag{4.8}$$

, com $\mathbf{x}, \mathbf{x}^{'} \in \Re\varepsilon^2$. Então;

$$\kappa^{'} = (1 + x_1 x_1^{'} + x_2 x_2^{'})^2 \tag{4.9}$$

$$= 1 + x_1 x_1^{'} + x_2 x_2^{'} + 2x_1^2 {x_1^{'}}^2 + 2x_2^2 {x_2^{'}}^2 + 2x_1 x_1^{'} x_2 x_2^{'} \tag{4.10}$$

Ora, se considerarmos $\varphi(\mathbf{x}) = [1, x_1^2, \sqrt{2}x_1 x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2]$, vemos que:

$$\kappa^{'}(\mathbf{x}, \mathbf{x}^{'}) =< (\varphi(\mathbf{x}), \varphi(\mathbf{x}^{'}) > \tag{4.11}$$

Pode parecer- a primeira vista- desnecessário se encontrar uma função $\kappa^{'}$ que represente o produto interno $< (\varphi(\mathbf{x}), \varphi(\mathbf{x}^{'}) >$, caso se conheça a função $\varphi(\mathbf{x})$. Entretanto, do ponto de vista computacional isto pode ser bastante interessante. Para ilustrar o fato, vamos considerar um expoente n "arbitrariamente"grande para $\kappa^{'}(\mathbf{x}, \mathbf{x}^{'})$, com $\mathbf{x}, \mathbf{x}^{'} \in \Re\varepsilon^{\beta}$. Assim, teríamos que:

$$\kappa^{'} = (1 + x_1 x_1^{'} + x_2 x_2^{'} + \cdots + x_\beta x_\beta^{'})^n \tag{4.12}$$

É fácil notar que número de *flops* (operações) para se computar $\kappa^{'}$ dependerá exclusivamente da ordem de β, ou seja, independe do n considerado.

4.2 Caracterização de um Kernel

4.2.1 Espaços com produto interno e Espaço de Hilbert

Definição 8. *Um espaço vetorial $\boldsymbol{X}$ sobre o corpo dos reais $\Re$ é dito um espaço vetorial com produto interno se existir uma relação simétrica bilinear $< . \quad , \quad . >$ que satisfaça*

$$< \boldsymbol{x}, \boldsymbol{x} > \geq 0 \tag{4.13}$$

Além disto, dizemos que um produto interno é *estrito* se:

$$< \mathbf{x}, \mathbf{x} > = 0 \tag{4.14}$$

, se, e somente se, $\mathbf{x} = 0$. O produto escalar $< (\boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}^{'}) >$ é denomindo de *Kernel*(κ) e é definido da seguinte maneira:

$$\kappa : \boldsymbol{X} \times \boldsymbol{X} \rightarrow \Re \tag{4.15}$$

Definição 9. *Um Espaço de Hilbert $\mathcal{F}$ é um espaço munido com produto interno e com as propriedades adicionais de ser separável e completo. Ser completo relaciona-se com o fato de que toda sequência de Cauchy $(h_n)_{n \geq 1}$ converge para um elemento $h \in \mathcal{F}$, onde uma sequência de Cauchy é definida por:*

$$\sup_{m,n} \quad \|h_n - h_m\| = 0$$

, quando $n \rightarrow \infty$.

Por outro lado, dizemos que $\mathcal{F}$ é separável se para todo $\epsilon > 0$ existe um conjunto finito de elementos $h_1, h_2, h_3 ...)$ de $\mathcal{F}$ tal que:

$$\min_i \quad \|h_i - h\| < \epsilon$$

Vamos agora enunciar[1] um teorema fundamental para a construção da teoria de Kernel.

Teorema 2. *Uma função*

$$\kappa : \boldsymbol{X} \times \boldsymbol{X} \rightarrow \Re \tag{4.16}$$

que ou é contínua ou possui domínio finito, pode ser ser decomposta em $< (\boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}^{'}) >$, onde $\boldsymbol{\varphi}(\boldsymbol{x})$ é uma transformação sobre um Espaço de Hilbert. $\mathcal{F}$ [Taylor and Cristianini, 2004].

[1] Para a demonstração completa deste teorema, do teorema de Mercer e das propriedades de Kernels, sugere-se ver Taylor and Cristianini [2004].

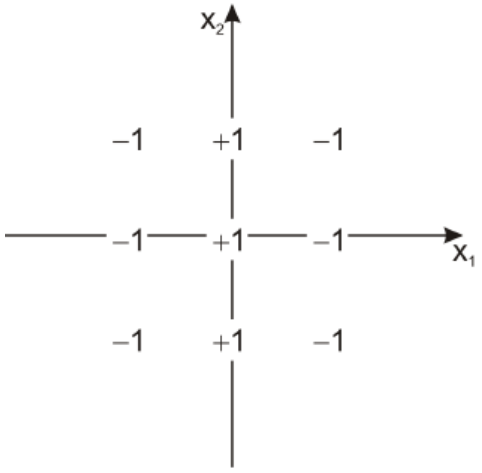
4.2.2 Tipos de Kernels

Tabela 4.1: Exemplos de Kernels

Kernel	$\kappa(\mathbf{x},\mathbf{x}^{'}) =< (\boldsymbol{\varphi}(\mathbf{x}),\boldsymbol{\varphi}(\mathbf{x}^{'}) >$
Polinomial	$(1+ < \mathbf{x},\mathbf{x}^{'} >)^n$
RBF function	$exp(\frac{-1}{2\sigma}\|\mathbf{x}-\mathbf{x}^{'}\|^2)$
sigmóidal	$tangh[\beta_0 < \mathbf{x},\mathbf{x}^{'} > +\beta_1]$

4.3 Transformação de classes não-linearmente Separáveis

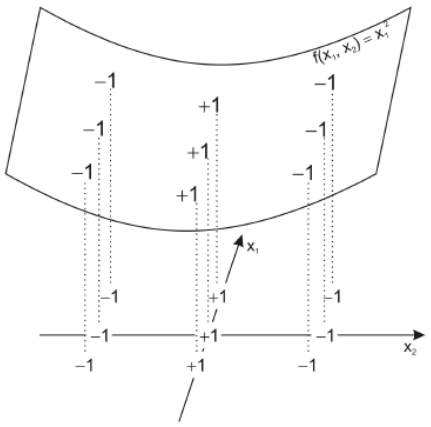
Apesar já termos descrito algumas das vantagens no uso de Kernels, para o objeto de estudo desta monografia, a grande qualidade dos Kernels é transformar pontos não-linearmente separáveis de um Espaço de Hilbert em linearmente separáveis num espaço de características, isto é, no espaço imagem de uma tranformação $\boldsymbol{\varphi}(\mathbf{x})$. O motivo teórico pelo qual isto é possível relaciona-se com o fato que conjuntos de dados não linearmente separáveis possuem maior probabilidade de serem linearmente separáveis num espaço de características de dimensão maior. Tal relação foi provada num conjunto de teoremas por Cover [1964]. Abaixo, algumas imagens ilustram o fato:



Fonte: [Rufino, 2011]

Figura 4.1: Conjuntos de dados não-linearmente separáveis

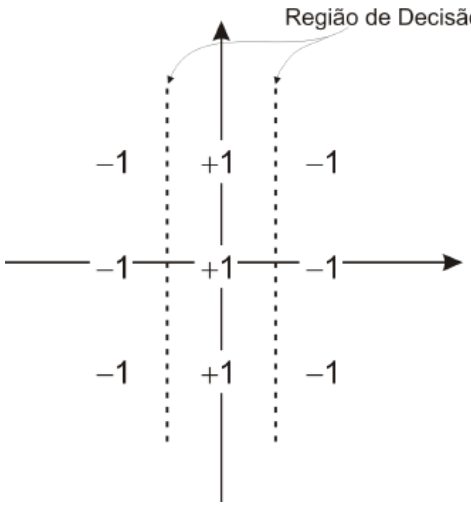
Considermos agora uma transformação $f(x_1,x_2) = x_1^2$. Então, graficamente nosso espaço de características é apresentado graficamente como:



Fonte: [Rufino, 2011]

Figura 4.2: Visualização do Espaço de características

Conseguimos então enxergar a seguinte fronteira de decisão:



Fonte: [Rufino, 2011]

Figura 4.3: Fronteira de Decisão no espaço de Características

Outro exemplo clássico para se demonstrar tal qualidade do Kernel, é o caso do "ou-exclusivo"(XOR). Neste caso, usamos o Kernel descrito por (4.9) para transportar os dados iniciais para um espaço de características, no qual estes passam a ser linearmente separáveis.

4.4 SVMs e Kernel

Retomando agora a motivação inicial com as Máquinas de Vetores de suporte, vamos definir o problema de otimização a ser resolvido. Observe que, uma vez escolhido convenientemente o Kernel e usando o formalismo lagrangiano desenvolvido para as máquinas de vetores de suporte com classes não-linearmente separáveis, temos uma poderosa ferramenta no âmbito da classificação de pontos de classes binárias.

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & Q(\boldsymbol{\alpha}) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2}\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j \alpha_i \alpha_j (\boldsymbol{\varphi}(\mathbf{x}_i).\boldsymbol{\varphi}(\mathbf{x}_j))\} \\ \text{sujeito a} \quad & 0 \leq \alpha_i \leq C, \quad para \quad i = 1, 2, 3, \ldots \\ & \sum_{i=1}^{P} \alpha_i . y_i = 0 \end{aligned}$$

, donde vem $\mathbf{w}_0 = \sum_{i=1}^{P} \alpha_{i0} \boldsymbol{\varphi}(\mathbf{x}_i)$ como solução do vetor[2] de pesos ótimo, com os α_{i0} sendo os multiplicadores de Lagrange do problema otimizado.

[2] Veja que estamos incorporando o bias ao vetor.

Capítulo 5

Conclusão

Este trabalho explorou aspectos teóricos e computacionais sobre alguns métodos de inteligência Artificial, em particular as Máquinas Vetores de Suporte e a Rede Neural Perceptron.

No âmbito das redes neurais, estoudou-se a motivação biológica fazendo-se, quando possível, a menção histórica do desenvovimento dos modelos. Foi feito também um apanhado geral acerca das principais arquiteturas de rede, assim como a discussão de paradigmas de aprendizado.

No que tange ao modelo do perceptron, foi possível inferir que apesar de vasta aplicabilidade, tal modelo consegue apenas classificar corretamente classes linearmente separáveis. Para afirmar tal fato, foi demonstrado o teorema de convergência do Perceptron para estas respectivas classes.

Vale destacar ainda a implementação do algoritmo do Perceptron no matlab em suas duas instâncias: fase de teste e fase de treinamento da rede neural com uma camada.

Em relação às Máquinas de Vetores de Suporte, buscou-se aqui construir a teoria usando forte apelo geométrico. Vimos que a construção de tais modelos leva naturalmente a um problema de otmização quadrática, o qual pode ser convenientemente transformado num problema de otimização dual. Tal teoria foi estendida no sentido de incorporar à função a ser otimizada os erros dos pontos que violavam a margem. Com isto, foi obtido outro problema de otimização para classes não-linearmente separáveis.

Para resolver tal problema, usamos a rotina interna do matlab para programação quadrática "*quadprog*". Verifcamos, como previsto, que a solução apresentada, isto é, pesos $\mathbf{w}_i$ definidores do hiperplano ótimo, possuem maior assertividae do que aqueles obtidos via algoritmo do Percptron.

Por fim, fez-se um breve estudo acerca de métodos Kernel. Demonstramos através de exemplos que os mesmos podem reduzir consideravelmente a complexidade no momento de se computar produtos internos, além de possuirem a propriedade de tornar pontos não-linearmente separáveis de um Espaço de Hilbert em pontos linearmente separáveis.

Apêndice A

Código da Fase de Treinamento do Perceptron:

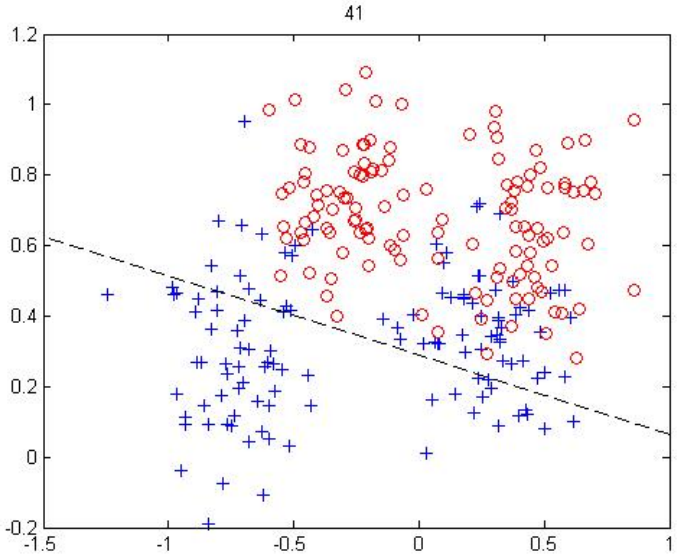
Como conjunto de dados de treinamento, vamos usar o recurso digital disponível em Ripley [2001] para o conteúdo abordado.

Código:

```
%Definindo funcao de Treino
function [w] =training(Xtr,dtr)
%Definindo numero maximo de iteracoes
it_max=100;
%Lendo a taxa de aprendizado
disp('Entre com a taxa de aprendizagem eta')
eta=input('')
%lendo dimensoes da matriz de treinamento
[m,n] =size(Xtr)
%Inserindo coluna relativa ao bias
Xtr(:,n+1)=1
disp(Xtr)
disp('New matrix')
%Encontrando indices dos vetores de cada classe
indA=find(dtr==-1);
indB=find(dtr==1);
x=linspace(-1.5,1);
%Inicializando vetor de pesos. Defnindo booleana verficadora de
    erros e contador.
w=zeros(n+1,1);
errorcheck =1 ;
it=0;
%Definindo contador de erros
```

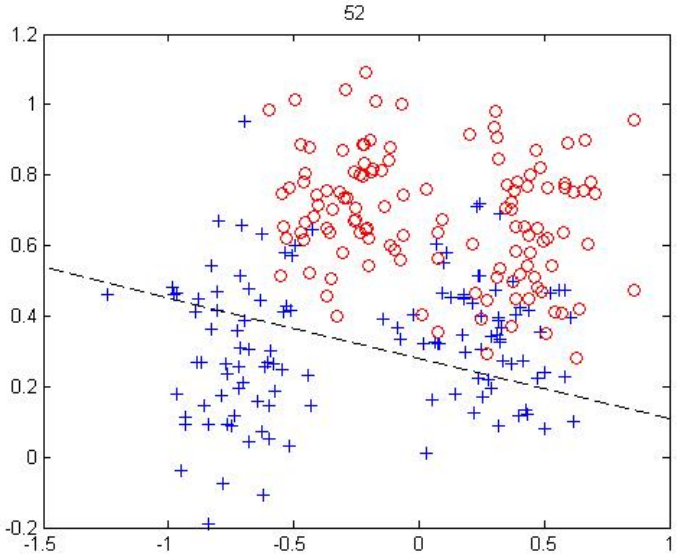
```
count=0;
%Etapa de treinamento
while ((errorcheck>0)&(it<it_max))
  it=it+1;
 errorcheck = 0;
    for i=1:m
%Verificando o sinal
     sinal=2*(Xtr(i,:)*w > 0)-1;
     if (sinal~=dtr(i))
%Atualizando contador de erros
         count = count +1
%Atualizando vetor de pesos
        w=w+eta*dtr(i)*Xtr(i,:)'
        errorcheck= 1;
      end
     end
    if mod(it,1)==0
      wt=w;
      x=linspace(-1.5,1,5)
%Definindo a reta- Hiperplano.
      y=-(wt(1)/wt(2))*x-(wt(3)/wt(2));
%Plotando a reta.
      plot(Xtr(indA,1),Xtr(indA,2),'+b',Xtr(indB,1),Xtr(indB,2),'or
          ',x,y,'--k');
      title(it);
      pause(0.2);
    end
    disp([it,errorcheck,count])
end
[w]
end
```

Podemos ver como o hiperplano muda durante o treinamento do Perceptron:



Fonte: Produção Própria

Figura A.1: Classificador linear numa determinada iteração



Fonte: Produção Própria

Figura A.2: Classificador linear após 3 iterações do treinamento

Apêndice B

Código da Fase de Teste do Perceptron

De maneira análoga[1] ao conjunto de treinamento usado durante fase homônima, usaremos para o teste do hiperplano obtido com o perceptron o conjunto de amostras disponível como recurso digital em Ripley [2001].

Código:

```
data=[%    xs           ys       yc
  0.05100797  0.16086164    0
 -0.74807425  0.08904024    0
 -0.77293371  0.26317168    0
  0.21837360  0.12706142    0
  0.37268336  0.49656200    0
 -0.62931544  0.63202159    0
 -0.43307167  0.14479166    0
 -0.84151970 -0.19131316    0
  0.47525648  0.22483671    0
  0.32082976  0.32721288    0
  0.32061253  0.33407547    0
 -0.89077472  0.41168783    0
  0.17850119  0.44691359    0
  0.31558002  0.38853383    0
  0.55777224  0.47272748    0
  0.03191877  0.01222964    0
  0.25090585  0.30716705    0
  0.23571547  0.22493837    0
 -0.07236203  0.33376524    0
  0.50440241  0.08054579    0
```

[1] Para o cálculo dos parâmetros ótimos da máquina de vetor de suportes também foram usados os conjuntos de treinamento e teste disponível em Ripley [2001].

```
−0.63223351  0.44552458    0
−0.76784656  0.23614689    0
−0.70017557  0.21038848    0
−0.64713491  0.15921366    0
−0.76739248  0.09259038    0
−0.51788734  0.03288107    0
 0.17516644  0.34534871    0
−0.68031190  0.47612156    0
 0.01595199  0.32167526    0
−0.71481078  0.51421443    0
 0.07837946  0.32284981    0
−0.80872251  0.47036593    0
−0.84211234  0.09294232    0
−0.98591577  0.48309267    0
 0.29104081  0.34275967    0
 0.24321541  0.51488295    0
−0.60104419  0.05060116    0
−1.24652451  0.45923165    0
−0.82769016  0.36187460    0
−0.62117301 −0.10912158    0
−0.70584105  0.65907662    0
 0.06718867  0.60574850    0
 0.30505147  0.47417973    0
 0.60788138  0.39361588    0
−0.78937483  0.17591675    0
−0.53123209  0.42652809    0
 0.25202071  0.17029707    0
−0.57880357  0.26553665    0
−0.83176749  0.54447377    0
−0.69859164  0.38566851    0
−0.73642607  0.11857527    0
−0.93496195  0.11370707    0
 0.43959309  0.41430638    0
−0.54690854  0.24956276    0
−0.08405550  0.36521058    0
 0.32211458  0.69087105    0
 0.10764739  0.57946932    0
−0.71864030  0.25645757    0
−0.87877752  0.45064757    0
−0.69846046  0.95053870    0
 0.39757434  0.11810207    0
−0.50451354  0.57196376    0
 0.25023622  0.39783889    0
 0.61709156  0.10185808    0
 0.31832860  0.08790562    0
−0.57453363  0.18624195    0
```

```
 0.09761865  0.55176786    0
 0.48449339  0.35372973    0
 0.52400684  0.46616851    0
-0.78138463 -0.07534713    0
-0.49704591  0.59948077    0
-0.96984525  0.46624927    0
 0.43541407  0.12192386    0
-0.67942462  0.30753942    0
-0.62529036  0.07099046    0
-0.02318116  0.40442601    0
 0.23200141  0.71066846    0
 0.09384354  0.46674396    0
 0.14234301  0.17898711    0
-0.61686357  0.25507763    0
 0.23636288  0.51543839    0
 0.38914177  0.40429568    0
-0.95178678 -0.03772239    0
 0.24087822  0.71948890    0
 0.12446266  0.45178849    0
-0.60566430  0.26906478    0
-0.71397188  0.30871780    0
 0.31008428  0.34675335    0
 0.18018786  0.46204643    0
-0.42663885  0.64723225    0
 0.06143230  0.32491150    0
 0.07736952  0.32183287    0
 0.42814970  0.13445957    0
-0.80250753  0.66878999    0
 0.40142623  0.42516398    0
 0.37084776  0.26407123    0
-0.80774748  0.41485899    0
 0.50163585  0.23934856    0
 0.58238323  0.22842741    0
-0.59136100  0.30230321    0
-0.87037236  0.26941446    0
-0.72086765  0.19676678    0
 0.27778443  0.21792253    0
 0.33240813  0.27349865    0
-0.14092068  0.39247351    0
-0.59759518  0.14790267    0
-0.85581534  0.14513961    0
-0.88912232  0.26896001    0
 0.21345680  0.43611756    0
-0.53467949  0.57901229    0
 0.31686848  0.39705856    0
-0.68121733  0.04209840    0
```

```
−0.97586127  0.45964811    0
 0.41457183  0.27141230    0
 0.32751292  0.36780137    0
−0.93209192  0.09362034    0
 0.58395341  0.47147282    0
−0.44437309  0.23010142    0
 0.29109441  0.19365556    0
−0.51080722  0.41496003    0
−0.96597511  0.17931052    0
 0.18741315  0.29747132    0
 0.17965417  0.45175449    0
−0.72689602  0.35728387    0
−0.54339877  0.41012013    0
−0.59823393  0.98701425    1
−0.20194736  0.62101680    1
 0.47146103  0.48221146    1
−0.09821987  0.58755577    1
−0.35657658  0.63709705    1
 0.63881392  0.42112135    1
 0.62980614  0.28146085    1
−0.46223286  0.61661031    1
−0.07331555  0.55821736    1
−0.55405533  0.51253129    1
−0.43761773  0.87811781    1
−0.22237814  0.88850773    1
 0.09346162  0.67310494    1
 0.53174745  0.54372650    1
 0.40207539  0.51638462    1
 0.47555171  0.65056336    1
−0.23383266  0.63642580    1
−0.31579316  0.75031340    1
−0.47351720  0.63854125    1
 0.59239464  0.89256953    1
−0.22605324  0.79789454    1
−0.43995011  0.52099256    1
−0.54645044  0.74577198    1
 0.46404306  0.51065152    1
−0.15194296  0.81218439    1
 0.48536395  0.82018093    1
 0.34725649  0.70813773    1
 0.43897015  0.62817158    1
−0.21415914  0.64363951    1
 0.57380231  0.63713466    1
 0.38717361  0.58578395    1
 0.32038322  0.53529127    1
−0.20781491  0.65132467    1
```

```
−0.18651283  0.81754816    1
 0.24752692  0.39081936    1
 0.66049881  0.89919213    1
−0.28658801  0.73375946    1
−0.32588080  0.39865509    1
−0.25204565  0.67358326    1
 0.37259022  0.49785904    1
−0.29096564  1.04372060    1
−0.30469807  0.86858292    1
−0.21389978  1.09317811    1
−0.36830015  0.75639546    1
−0.46928218  0.88775091    1
 0.39350146  0.77975197    1
−0.45639966  0.80523454    1
 0.51128242  0.76606136    1
 0.22550468  0.46451215    1
 0.01462984  0.40190926    1
−0.19172785  0.80943313    1
 0.38323479  0.75601744    1
 0.49791612  0.61334375    1
 0.35335230  0.77324337    1
−0.34722575  0.70177856    1
 0.58380468  0.76357539    1
−0.13727764  0.71246351    1
 0.38827268  0.44977123    1
−0.53172709  0.61934293    1
−0.11684624  0.87851210    1
 0.54335864  0.41174865    1
−0.45399302  0.66512988    1
−0.21913200  0.83484947    1
 0.30485742  0.98028760    1
 0.65676798  0.75766017    1
 0.61420447  0.75039019    1
−0.45809964  0.77968606    1
−0.21617465  0.88626305    1
−0.26016108  0.81008591    1
 0.31884531  0.84517725    1
−0.23727415  0.80178784    1
 0.58310323  0.77709806    1
 0.02841337  0.75792620    1
−0.41840136  0.68041440    1
 0.67412880  0.60245461    1
−0.25278281  0.70526103    1
 0.51609843  0.62092390    1
 0.20392294  0.91641482    1
−0.17207124  1.00884096    1
```

```
 0.27274507  0.29346977    1
 0.07634798  0.56222204    1
-0.36653499  0.64831007    1
 0.44290673  0.80087721    1
-0.19976385  0.54295162    1
-0.54075738  0.65293033    1
-0.07060266  1.00296912    1
 0.50715054  0.35045758    1
-0.06048611  0.62982713    1
 0.21532928  0.60260249    1
 0.46809108  0.87182416    1
-0.29888511  0.73669866    1
 0.86129620  0.47289330    1
 0.70120877  0.74572893    1
-0.11342797  0.60067099    1
 0.31234354  0.90756345    1
-0.12172541  0.84112851    1
 0.36867857  0.37052586    1
 0.57311489  0.40949740    1
-0.25841225  0.67192335    1
 0.30937186  0.50823318    1
 0.43319338  0.77016967    1
-0.30448035  0.57820106    1
 0.44276338  0.58023403    1
-0.19442057  0.89876808    1
-0.06105237  0.74184946    1
 0.07619347  0.35386246    1
 0.85826993  0.95819523    1
 0.37039200  0.72342401    1
 0.51481515  0.76203996    1
 0.43127521  0.54259166    1
 0.42286091  0.65242185    1
 0.29815001  0.93453682    1
 0.37128253  0.70089181    1
-0.51528729  0.76473490    1
 0.38525783  0.65528189    1
-0.34825368  0.50529981    1
 0.68510504  0.78067440    1
-0.36528923  0.45703265    1
-0.40903577  0.74230433    1
 0.43574387  0.44689789    1
 0.26887846  0.44559230    1
-0.49254862  1.01443372    1
 0.07615960  0.63795180    1
 0.49226224  0.46876241    1
-0.40249641  0.71301084    1 ] ;
```

```
252 
253 teste = [%    xs            ys       yc
254  −0.970990139  0.429424950    0
255  −0.631997027  0.251952852    0
256  −0.773605760  0.690750778    0
257  −0.606211523  0.175677956    0
258  −0.539409005  0.376744239    0
259  −0.960325850  0.110040710    0
260  −1.041375608  0.328508085    0
261  −0.822600536  0.175874200    0
262  −0.943714771 −0.180633309    0
263  −0.968763299  0.296070217    0
264  −0.853637980  0.644010559    0
265  −0.771994930  0.476344773    0
266  −0.718952712  0.090457675    0
267  −0.539520701  0.447837856    0
268  −0.540093447  0.551067215    0
269  −0.792923186  0.531235891    0
270  −0.861472850  0.287352652    0
271  −0.470131571  0.544251260    0
272  −0.770683778  0.482733051    0
273  −0.803031230  0.228632039    0
274  −0.962520756  0.367759881    0
275  −0.681960494  0.495354977    0
276  −0.433007837  0.213645636    0
277  −0.336831640  0.293614869    0
278  −0.696425307  0.315194495    0
279  −0.355766886  0.269794553    0
280  −0.547898136  0.277054714    0
281  −0.799663889  0.292931173    0
282  −0.780012402  0.038437662    0
283  −0.853938355  0.198423604    0
284  −0.896295454  0.286916469    0
285  −0.824028270  0.295231859    0
286  −0.901075546  0.321018371    0
287  −0.556718720  0.358145252    0
288  −0.871004652  0.258992681    0
289  −0.800820459  0.363123198    0
290  −0.699003238  0.417050087    0
291  −0.759409251  0.366156047    0
292  −0.775268090  0.306716684    0
293  −0.893576947 −0.096908084    0
294  −0.284857192  0.307321395    0
295  −0.665571750  0.365820514    0
296  −0.741374392  0.298498149    0
297  −0.767733049  0.245811163    0
```

```
−0.779306345  0.319092986    0
−0.892190952  0.201459901    0
−0.122811626  0.516497113    0
−0.731730651  0.055992550    0
−1.011976425  0.344692082    0
−0.573762197  0.059676643    0
−0.641425285  0.333730563    0
−0.985902178  0.162020997    0
−0.661140507  0.136840396    0
−0.749218489  0.185148533    0
−0.540329548  0.387396621    0
−0.592092859  0.447510299    0
−0.860077357  0.218917745    0
−0.867516891 −0.137491677    0
−0.590055695  0.466004783    0
−0.775966325  0.403399745    0
−0.849687489  0.315466589    0
−0.746283040  0.256242513    0
−0.700854929  0.518361424    0
−0.923680439  0.449453255    0
−0.912092992  0.407980138    0
−0.650765709  0.412200546    0
−0.980330108  0.299281948    0
−0.744408938  0.203087089    0
−0.604170665  0.326156917    0
−0.735903002  0.655288145    0
−0.643607616  0.513819006    0
−0.963376987  0.249000843    0
−0.426980732  0.282178155    0
−0.654762824  0.562181098    0
−0.843491783  0.345421521    0
−0.553968009  0.538960351    0
−0.716946447  0.122102049    0
−0.775328790  0.498892271    0
−0.640289822  0.435762487    0
−0.516878864  0.182337108    0
−0.952125366  0.298280511    0
−0.723017513  0.256182935    0
−0.658805240  0.269147489    0
−0.464552773  0.218324319    0
−0.564517221  0.196511498    0
−0.814096964  0.228304066    0
−0.396184143  0.511765539    0
−0.996637001  0.209223029    0
−0.815950989  0.235966820    0
−0.526626592  0.418687316    0
```

```
−0.667763995  0.428833798    0
−0.658898181  0.031828081    0
−0.923935948  0.530254142    0
−0.909973792  0.451785093    0
−0.410551229  0.252159645    0
−0.462064440  0.230673805    0
−0.366146922 −0.036140226    0
−0.595861370  0.400288539    0
−0.704392096  0.238984335    0
−0.841225771  0.577095745    0
−0.969828933  0.155360193    0
−0.557037265  0.314190393    0
−0.671104208  0.361767035    0
−0.503286446  0.566417412    0
−0.950325858  0.078493347    0
−0.675813120  0.319308250    0
−0.831561973  0.143581661    0
−0.435074090  0.492855894    0
−0.793021028  0.118140919    0
−0.848627588  0.082762982    0
−0.820269797  0.395714263    0
−0.422092727  0.477760711    0
−0.408676218  0.374918252    0
−0.546953839  0.473748255    0
−0.735444130  0.266138774    0
−0.582205470  0.271991191    0
−0.338346632  0.242426860    0
−0.535045557  0.118043648    0
−0.493743519  0.717856305    0
−0.760932705  0.416245530    0
−0.515677444  0.184242721    0
−0.673504588  0.296239478    0
−0.459705697  0.186931282    0
−0.694881314  0.381840980    0
−0.387447545  0.080890693    0
−0.596036129  0.184974829    0
−0.664372536  0.423940859    0
−0.883742635  0.614943083    0
−0.509344933  0.290033636    0
−0.925124882  0.604748154    0
−0.841007867  0.290327096    0
−0.894120137  0.157169952    0
−0.646573229  0.609447746    0
−1.017873059  0.148721295    0
−0.582528753  0.184940557    0
−0.897329196  0.532091737    0
```

```
−0.465016860  0.285520226    0
−0.726508681  0.181867205    0
−0.514352969  0.156961029    0
−0.739246011  0.408845252    0
−0.537049319  0.307417180    0
−0.923407832  0.492249753    0
−0.663217181  0.241275721    0
−0.871900824  0.191786697    0
−0.574764695  0.216699985    0
−0.778723382  0.417127421    0
−0.717491428  0.169911784    0
−0.293985190  0.341692708    0
−0.732183039  0.611673182    0
−0.672451661  0.290330390    0
−0.392906014  0.314507904    0
−0.821496561  0.383502471    0
−0.441649840  0.131552989    0
−0.734149425  0.138366727    0
−0.353467324  0.403725989    0
−0.756729286  0.140926608    0
−0.985271855  0.307051129    0
−0.734362749  0.131915653    0
−0.843814454  0.508797861    0
−0.871470989  0.409534472    0
−0.643774042  0.386072579    0
−0.617659001  0.067340392    0
−0.282068649  0.693923139    0
−0.402555368  0.204385656    0
−0.458583969  0.420739380    0
−0.846296983  0.277152491    0
−1.048542317  0.338822747    0
−0.799795307  0.309430762    0
−0.852040552  0.307281614    0
−0.616474678  0.252952510    0
−0.691690351  0.272750414    0
−0.809142202  0.441901584    0
−0.837139722  0.269171931    0
−0.743520251  0.247417602    0
−0.660650230 −0.028489077    0
−0.594815839  0.109164679    0
−0.597128033 −0.037465241    0
−0.921420258 −0.069844290    0
−0.877566913  0.304297059    0
−0.765371773  0.596974416    0
−0.699840550  0.167126769    0
−0.523434825 −0.064742897    0
```

```
−0.656387744  0.012460495    0
−1.036967640  0.141450813    0
−0.715165192  0.217239838    0
−0.747858131  0.569994813    0
−0.625684541  0.320122450    0
−0.756699924  0.174518616    0
−0.679690670  0.438410861    0
−0.612004202 −0.134269826    0
−0.647906789  0.239638558    0
−0.691066413  0.255635309    0
−0.675112764  0.550169559    0
−0.851072790  0.474955936    0
−0.837051482  0.408050507    0
−0.961405831  0.588207922    0
−0.642774716  0.163487304    0
−0.892075711  0.064132978    0
−0.927798777  0.072240031    0
−0.751800726  0.409258566    0
−0.805341030  0.064157327    0
−0.692838235  0.171715163    0
−0.703943931  0.476730183    0
−0.694804098  0.268655402    0
−0.567758798  0.207116645    0
−0.822380000  0.268404036    0
−0.565082539  0.327015498    0
−0.724181702  0.625763803    0
−0.916357511  0.236124996    0
−0.430182548  0.268033748    0
−0.632645741  0.522382761    0
−0.850972862  0.345168936    0
−0.609691020  0.501872186    0
−0.705661024  0.220694983    0
−0.693161871  0.100244402    0
−0.633922642  0.390701059    0
−0.710406768  0.015180240    0
−1.055052036  0.517833140    0
−0.621276063  0.167382599    0
−0.613423246  0.266134950    0
−0.989565379  0.166693580    0
−0.923580375  0.412606504    0
−0.889581095  0.426760653    0
−0.930040388  0.240533824    0
−0.691421356  0.006339557    0
−1.031412255  0.482277646    0
−0.701394895  0.462356010    0
−0.627721178  0.243813111    0
```

```
−0.829380326  0.487867261    0
−0.612200851  0.121715064    0
−0.528139634  0.449962538    0
−0.616674472  0.058254182    0
−0.649202842  0.263909873    0
−0.655384302  0.225793561    0
−0.750085240  0.119545244    0
−0.471920626  0.278830975    0
−0.219905912  0.315052974    0
−0.871701260  0.240570129    0
−0.730197977  0.295504781    0
−0.620676222  0.046383576    0
−0.657830687  0.265899761    0
−0.475352116  0.279850946    0
−0.734794644  0.365235616    0
−0.772673638  0.355477724    0
−0.620710470  0.770796635    0
−0.529626406  0.091067609    0
−0.730846476  0.642803364    0
−0.938694493  0.324275071    0
−0.723706354 −0.017999841    0
−0.979569099 −0.003034376    0
 0.448754392  0.015050386    0
−0.077907282  0.245842052    0
 0.316786631  0.252917817    0
 0.229597046  0.067681573    0
 0.197949376  0.310003887    0
 0.048404642 −0.037865268    0
 0.270601003  0.260199166    0
 0.516192043  0.258256258    0
 0.154718993  0.040306842    0
−0.005611276  0.223658499    0
 0.365076313 −0.001956641    0
 0.086615547  0.138482814    0
 0.198645891  0.047611642    0
 0.131870660  0.402255360    0
 0.585894768  0.433203159    0
−0.023498655  0.379919943    0
 0.394174061  0.533936878    0
 0.595983773  0.680516952    0
 0.388419733  0.321931614    0
 0.270452263  0.360309566    0
 0.336909893  0.176262915    0
 0.481432232  0.326027716    0
 0.246865240  0.532700400    0
−0.020439631  0.132155124    0
```

```
 0.389941424  0.309223343    0
 0.048115168  0.104763308    0
 0.284816331 −0.048775617    0
 0.529166911  0.285314795    0
 0.349208427  0.063167392    0
 0.323888259  0.192358455    0
 0.321213977  0.101190083    0
 0.303365953  0.286689359    0
−0.075979803  0.312196126    0
 0.317894059  0.110578558    0
 0.136145272  0.223509762    0
 0.086777443  0.397316175    0
 0.330555298 −0.018831347    0
 0.202260475  0.212061643    0
 0.276704436  0.541792424    0
 0.244814590 −0.033434890    0
 0.429043775  0.183967494    0
 0.340412789  0.237474210    0
 0.382064022  0.123295299    0
 0.381833239  0.085809636    0
 0.424417864  0.321954582    0
 0.206306313  0.348957865    0
 0.091614953  0.309132098    0
 0.627597689  0.472188745    0
 0.270244718  0.361936451    0
 0.127928396  0.368238186    0
 0.399192895  0.120050819    0
 0.450618123  0.452328633    0
 0.254900382  0.410220018    0
 0.259523390  0.124427489    0
 0.417004689  0.300805900    0
 0.346581338  0.283479475    0
 0.748854615  0.246812787    0
 0.428530072  0.636260298    0
 0.127369504  0.321732050    0
 0.528722462  0.227075837    0
 0.618168220  0.327309276    0
 0.286029472  0.215643450    0
 0.142578461  0.112955825    0
 0.282764909  0.091628143    0
 0.788220007  0.464545152    0
 0.119165220  0.239567886    0
 0.244772936  0.014906673    0
 0.160442893  0.455259044    0
 0.454067300  0.332582882    0
−0.057868287  0.498675578    0
```

```
−0.111365306  0.079756044    0
 0.198824819  0.476017542    0
 0.595468169  0.162120124    0
 0.085627364  0.315262031    0
 0.465261497  0.123331422    0
 0.359673625  0.364504393    0
 0.111822093  0.296370162    0
 0.509269078  0.464037322    0
 0.470888018  0.285556829    0
 0.393262912  0.093782124    0
 0.311897634  0.286626364    0
 0.151594554  0.268411495    0
 0.084423498  0.319282396    0
 0.208641564  0.230226362    0
 0.361230606  0.506867239    0
 0.425667999  0.239049251    0
 0.399549324  0.136827304    0
 0.279615939  0.310402719    0
 0.109049911  0.630255432    0
 0.102929855  0.446152743    0
 0.551085316  0.313983603    0
 0.579201159  0.179353765    0
 0.356514867  0.178396614    0
 0.259861364  0.096917764    0
 0.545480531  0.272730569    0
 0.398789597  0.149343536    0
 0.383441254  0.243298247    0
 0.405415302  0.351024129    0
 0.249091946  0.423059272    0
 0.293535767  0.133960638    0
 0.149869213  0.305675082    0
 0.224986842  0.464864831    0
 0.240826479  0.233973445    0
 0.122917552  0.406179372    0
 0.301231733  0.178773911    0
 0.257698819  0.537312141    0
 0.446288764  0.206483371    0
 0.511214849  0.156330717    0
 0.474675267  0.454212426    0
 0.373402327  0.107531816    0
 0.453575217  0.013564367    0
 0.363708989  0.324209899    0
 0.323172397  0.308234424    0
 0.263568182  0.097321560    0
 0.375989273  0.511128488    0
 0.483416817 −0.027606822    0
```

```
 0.412708967  0.353260156    0
 0.294590710  0.338631607    0
 0.148425126  0.313998286    0
 0.476236614  0.009138517    0
 0.051021769  0.518229423    0
 0.488029582  0.492206314    0
 0.193703118  0.356127440    0
 0.390385684  0.402548715    0
 0.166515062  0.077486533    0
 0.378346001  0.205554127    0
 0.059890677  0.615481812    0
-0.077252668  0.325973024    0
 0.519325984  0.352901733    0
 0.271955420  0.031010063    0
 0.027254987  0.289394991    0
 0.437437673 -0.027210937    0
 0.028370640  0.166304765    0
 0.433657082  0.604909277    0
 0.280505393  0.022916023    0
 0.300735977  0.188023897    0
 0.182031568  0.292354741    0
 0.316158641  0.423973591    0
 0.530601146  0.287109075    0
 0.210237556  0.384357431    0
 0.399444521  0.496882692    0
 0.272113433  0.437262474    0
 0.418146305  0.145521656    0
 0.504825239  0.154106314    0
 0.166974207  0.180641380    0
 0.106527356  0.500370591    0
 0.607348514  0.184680121    0
 0.517847638  0.396858357    0
 0.231553652  0.403086636    0
 0.255029497  0.430592319    0
 0.287511011  0.219412906    0
 0.200852107  0.272097495    0
 0.226547849  0.244596483    0
 0.011878373  0.352803074    0
 0.380569910  0.434089493    0
 0.519215428  0.072764703    0
 0.623854880  0.338983888    0
 0.183173455  0.255322403    0
 0.226420389  0.075341621    0
 0.455356509  0.367957232    0
 0.332301375 -0.011058516    0
 0.376306021  0.188460770    0
```

```
 0.428169526  0.054583036    0
 0.145829529  0.368253163    0
 0.493757540  0.376063674    0
 0.529391969  0.074698658    0
 0.409826160  0.280322788    0
 0.612354746  0.120926664    0
 0.221568084  0.273458368    0
 0.427545649  0.106200846    0
 0.533325611  0.591671136    0
 0.462109537  0.357955560    0
 0.182362120  0.298520960    0
 0.310107790  0.301510248    0
 0.159799550  0.257640193    0
 0.254288145  0.374308080    0
 0.316374077  0.029411804    0
 0.285942260  0.338773678    0
 0.552541865 -0.016858031    0
-0.004090460  0.399012387    0
 0.060484031  0.277592649    0
 0.545097739  0.218461339    0
 0.268284924  0.267903340    0
 0.159022649  0.531382417    0
 0.492658208  0.486286052    0
-0.128240252  0.533333926    0
 0.447760080  0.284865402    0
 0.239374886  0.462386877    0
 0.138634894  0.395550274    0
 0.417284343  0.200022118    0
 0.178303979  0.306720386    0
 0.221552636  0.396534895    0
-0.009120409  0.724738825    0
 0.292748806  0.414432640    0
 0.300563713  0.214325496    0
 0.242506812  0.232690286    0
 0.234494302  0.247006083    0
 0.352550448  0.351581175    0
 0.185994378  0.269914887    0
 0.409680307  0.212370722    0
 0.163919950  0.026130185    0
 0.169756191  0.104358886    0
 0.354398935  0.227524046    0
 0.388870060  0.042378087    0
 0.344788486  0.246053805    0
 0.193145216  0.271352787    0
 0.430800164  0.263193765    0
 0.232808591  0.445516712    0
```

```
 0.326059317  0.563886858    0
 0.330837091  0.256040145    0
 0.323691216  0.356872920    0
 0.367737090 −0.088857683    0
 0.530750561  0.327389964    0
 0.089596372  0.338423910    0
 0.432192982  0.394261493    0
 0.186694048  0.438187113    0
 0.458275145  0.324647633    0
 0.480078071  0.374810492    0
 0.582758378  0.390433695    0
 0.437808065  0.389265557    0
 0.208830936  0.010096493    0
 0.377797466  0.474572076    0
 0.183803076 −0.090083970    0
 0.155682547  0.537563127    0
 0.071926861  0.572783083    0
 0.364435618 −0.123841713    0
 0.408213991  0.254483065    0
 0.466073956  0.398618252    0
 0.614281743  0.283302172    0
−0.047151673  0.214579449    0
 0.326917150  0.468066389    0
 0.458840582  0.443470083    0
 0.109537926  0.189505910    0
 0.161895892  0.123705078    0
 0.450055408  0.501518844    0
 0.368869484  0.557190529    0
 0.334209119  0.413960488    0
−0.031121068  0.228014456    0
 0.176753850  0.430199990    0
 0.552527788  0.224902508    0
 0.304266409  0.220287210    0
 0.210462653  0.415336683    0
 0.063953710  0.045543235    0
−0.063149684  0.351389125    0
 0.073535710  0.252143534    0
 0.665453703  0.203720086    0
 0.539642761  0.279986737    0
 0.250981585  0.069569958    0
 0.392679888  0.090261998    0
 0.431409216  0.288456378    0
−0.516451834  0.501256111    1
−0.116775286  0.483404773    1
−0.327960793  0.546240228    1
−0.394572192  0.755243715    1
```

```
−0.110201988  0.553402230    1
−0.160538577  0.579525838    1
−0.124742465  0.323661757    1
−0.109742769  0.696514698    1
−0.687328305  0.807033124    1
−0.358374262  0.807265743    1
−0.335836520  0.392482381    1
−0.321604223  0.591913273    1
−0.091546228  0.562483354    1
−0.660890881  0.611049023    1
−0.561938441  0.907495412    1
−0.244433911  0.451367292    1
−0.392885460  0.550604753    1
−0.429608736  0.644152661    1
−0.090462865  0.522251590    1
−0.436484641  0.520039359    1
−0.519966218  0.940830736    1
−0.418391404  1.011277424    1
−0.405807798  0.738999068    1
−0.085688384  0.847932361    1
−0.210347223  0.416696729    1
−0.531896660  0.452618557    1
−0.294588066  0.846012850    1
−0.092753982  0.693082777    1
−0.314549926  0.797236706    1
−0.262918395  0.787474678    1
−0.389819133  0.579880509    1
−0.162163174  0.315021403    1
−0.418250429  0.684349895    1
−0.356533257  0.896022491    1
−0.461800168  0.782142975    1
−0.149067005  0.837864969    1
−0.376621128  0.553207248    1
−0.235807559  0.642937572    1
−0.433816383  0.568682995    1
 0.003602461  0.804352974    1
−0.286855152  0.710632583    1
−0.424066790  0.994872459    1
−0.270030002  0.833427152    1
−0.239212386  0.378268423    1
−0.255304685  0.822105360    1
−0.196569409  0.703182679    1
−0.125203354  0.844725933    1
−0.338351441  0.680964321    1
−0.383184405  0.839383812    1
−0.398513962  0.750284450    1
```

```
 0.027844709  0.537770177    1
−0.295483256  0.846722230    1
−0.552989277  0.794817114    1
−0.004901838  0.608282407    1
−0.029384352  0.614072912    1
−0.444694587  0.779042878    1
−0.338928122  0.789725990    1
 0.122195503  0.784475027    1
−0.186584991  0.560614872    1
−0.295015658  0.840559001    1
−0.102630670  0.675938267    1
−0.430785693  0.645617846    1
−0.099297566  0.894434898    1
−0.009264193  1.012595196    1
−0.560973647  0.807423104    1
−0.536294204  0.529432752    1
−0.563297476  0.646381268    1
−0.292902091  0.620924549    1
−0.107464304  0.615869773    1
−0.261216307  0.699646352    1
−0.105100716  0.868085863    1
−0.362473095  0.683245848    1
−0.548222187  0.726739882    1
−0.522717054  0.636324411    1
−0.406753361  0.858975870    1
−0.272149948  1.009788333    1
−0.058505372  0.722037722    1
−0.286284031  0.564831018    1
−0.145641743  0.527786275    1
−0.254951568  0.909735133    1
−0.200910922  0.911648155    1
−0.397769966  0.398117280    1
−0.547436085  0.779495789    1
−0.231129177  0.491139768    1
−0.473894736  0.682466158    1
−0.231075189  0.453157246    1
−0.268776826  0.676814477    1
−0.180889587  0.880462410    1
−0.326237906  0.599734095    1
−0.252657163  0.575832499    1
−0.294967226  0.707617098    1
−0.441714737  0.649258390    1
−0.434336942  0.859634714    1
−0.080950672  0.608362742    1
−0.256056671  0.465280126    1
−0.767972482  0.818894418    1
```

```
−0.250929687  0.807765177    1
−0.233531508  0.536107452    1
−0.166252171  0.578022234    1
−0.399389870  0.961981117    1
−0.383257048  0.918196737    1
−0.246208261  0.728269018    1
−0.112873567  0.825689335    1
−0.096666032  0.707306804    1
−0.457949369  0.704015342    1
−0.255003562  0.504258034    1
−0.073434667  0.722783609    1
−0.409375468  0.526062925    1
−0.363348126  0.881713044    1
−0.257217769  0.607597755    1
−0.349331300  0.703112332    1
−0.151880213  0.492886000    1
−0.404171363  0.737139545    1
−0.462320910  0.423673110    1
−0.546143281  0.835222198    1
−0.229962943  0.611218821    1
−0.246561278  0.550748181    1
−0.392635644  0.396901704    1
−0.175983074  0.659236133    1
−0.160444346  0.856989440    1
−0.341235994  0.536421185    1
−0.333233675  0.558945553    1
−0.274226030  0.677337101    1
−0.394217634  1.084965709    1
−0.177110920  1.174990894    1
−0.403972304  0.705580257    1
−0.387046408  0.654499407    1
−0.044038573  0.753839485    1
−0.278389636  0.349432166    1
−0.272249470  0.234622985    1
−0.191592271  0.380898603    1
−0.590368203  0.698331693    1
−0.374188840  0.819242381    1
−0.351703587  0.730361507    1
−0.281959049  0.469288157    1
−0.751945036  0.885219702    1
−0.306929899  0.574182522    1
−0.762727447  0.890352701    1
−0.564448380  0.729602705    1
 0.040323664  0.779572618    1
−0.462188702  0.998868915    1
−0.447915766  0.843500207    1
```

```
−0.217001799  0.796623800    1
−0.112509220  0.611900551    1
−0.131149777  0.948975611    1
−0.403054671  0.786868546    1
 0.008848708  0.652933806    1
 0.090647590  0.654317764    1
−0.358620932  0.936462477    1
−0.441265488  0.326283245    1
−0.479842420  0.788087594    1
−0.588843824  0.648214630    1
−0.562606783  0.754763105    1
−0.514270007  0.324312047    1
−0.392905106  0.821041597    1
−0.075132059  0.685702990    1
−0.196830870  0.714112820    1
−0.301481674  0.552313534    1
−0.181585205  0.659988770    1
−0.114373131  0.736877415    1
−0.331936585  0.440209520    1
−0.266807581  0.545085006    1
−0.475109818  0.947483833    1
−0.557037972  0.778719573    1
−0.193240214  0.574512048    1
−0.029348731  0.829601881    1
−0.383376526  0.624385592    1
−0.035071125  0.812800625    1
−0.060506093  0.772166835    1
−0.160710931  0.530042141    1
−0.210362275  0.567446850    1
−0.283272444  0.798839816    1
−0.520613526  0.837372559    1
−0.263870495  0.687937002    1
−0.060226406  0.688228649    1
−0.429473669  0.654717940    1
−0.325250467  0.791105596    1
 0.094837102  0.750572909    1
−0.326848641  0.823553280    1
−0.537630937  0.827068887    1
−0.589458171  0.897096209    1
−0.255109811  0.737443245    1
−0.350722503  0.739648314    1
−0.111745167  0.705987527    1
−0.213435551  0.466547665    1
−0.272518877  0.683481004    1
−0.440414101  0.974317798    1
−0.303362790  0.576264653    1
```

```
−0.221200040  0.987888085    1
−0.286914561  0.619578181    1
 0.096845361  0.511673423    1
−0.363110834  0.661562448    1
−0.211246704  0.813171823    1
−0.222052903  0.686080299    1
−0.321828330  0.624357510    1
−0.473737950  0.506318972    1
−0.212793549  0.774693470    1
 0.008463870  0.614591369    1
−0.205693420  0.644919563    1
−0.378486601  0.778361218    1
−0.229442899  0.594732866    1
−0.162703081  0.930991126    1
−0.321296905  0.828610911    1
−0.400332594  0.688297191    1
−0.312050685  0.618494750    1
−0.039349153  0.959790721    1
−0.273914659  0.599403497    1
−0.348565665  0.612606769    1
−0.413758325  0.696448995    1
−0.098831839  0.854519409    1
−0.287690535  0.883301183    1
−0.383124103  0.672367628    1
−0.561271474  1.067278573    1
−0.166431846  0.897151624    1
−0.635114720  0.688087392    1
−0.332175204  0.501477407    1
−0.474805835  0.711218005    1
−0.116004389  0.708363990    1
−0.477937453  0.702949001    1
−0.126810442  0.971409951    1
−0.156822576  0.457687275    1
−0.293523863  0.856486819    1
−0.129615545  0.891819146    1
−0.108242313  0.644814421    1
−0.501979824  0.370050434    1
−0.138108021  0.612928438    1
−0.179322731  0.366517387    1
−0.458093963  0.571370985    1
−0.028565637  0.486501211    1
−0.426175577  0.461765467    1
−0.310680953  0.544905689    1
−0.180247439  0.876336671    1
−0.217870537  0.390856979    1
−0.315992257  0.736172703    1
```

```
 0.236276902  0.714179743    1
−0.185456072  0.702294953    1
−0.203065705  0.317910002    1
−0.296142711  0.648026589    1
−0.448939545  0.650603998    1
 0.077064746  0.797884087    1
 0.034024500  0.788213418    1
−0.439519067  0.946446539    1
−0.471452461  0.708540945    1
−0.263821096  0.565778110    1
−0.676333519  1.064998541    1
−0.394630195  0.732544473    1
−0.334698783  0.638313660    1
 0.043828297  0.782970773    1
 0.073254562  0.639405607    1
−0.358305948  0.638878595    1
 0.289824646  0.645297701    1
 0.479141353  0.769272264    1
 0.180670084  0.518893193    1
 0.199825830  0.747216818    1
 0.735249202  0.833027044    1
 0.249991814  0.350660256    1
 0.413137889  0.854044549    1
 0.518581462  0.386362750    1
 0.465359263  0.854392557    1
 0.348309276  0.680024754    1
 0.174782318  0.544423218    1
 0.549911988  0.472172493    1
 0.203934276  0.410263392    1
 0.338644108  1.028370469    1
 0.161322119  0.950855699    1
 0.350961307  0.686427652    1
 0.090257414  0.846995122    1
 0.764373743  0.615571296    1
 0.414756998  0.893306725    1
 0.679361421  0.659759084    1
 0.640285978  0.804268545    1
 0.630876040  0.710028594    1
 0.366370214  0.772543364    1
 0.314611449  0.755070836    1
 0.745924055  0.706345767    1
 0.489768059  0.684198041    1
 0.075247977  0.621422345    1
 0.499573139  0.679632119    1
 0.350405143  0.443980792    1
 0.636928363  0.603842916    1
```

```
 0.224908918  0.840917922    1
−0.032261912  0.655726651    1
 0.627052189  0.808688697    1
 0.263348975  0.455434849    1
 0.520257017  0.762965338    1
 0.151882522  0.966544141    1
 0.098482589  0.517323437    1
 0.201212077  0.549826846    1
 0.371298202  0.761389940    1
 0.497766489  0.769076360    1
 0.409493154  0.305118700    1
 0.340849813  0.766677739    1
 0.391675543  0.489773920    1
 0.516131854  0.412661585    1
 0.522760611  0.520845425    1
 0.446358722  0.869775036    1
 0.224400728  0.559199836    1
 0.583149627  0.871728559    1
 0.420184227  0.768544337    1
 0.340883764  0.582414682    1
 0.407626346  1.016274588    1
 0.226804848  0.997357208    1
 0.461550030  0.728402685    1
 0.275762111  0.773039119    1
 0.304760108  0.405069957    1
 0.636786149  0.521153930    1
 0.544820787  0.902598154    1
 0.816098957  0.643244361    1
 0.454637082  0.627059827    1
 0.416886517  0.498139441    1
 0.585814059  0.472857968    1
 0.158972903  0.877325952    1
 0.218197123  0.791103192    1
 0.436713777  0.582375556    1
 0.465359340  0.619108530    1
 0.346901746  0.776639489    1
 0.599207277  0.605698565    1
 0.463002935  0.972725613    1
 0.694263789  0.550710864    1
 1.000277812  0.669240364    1
 0.503660224  0.451743317    1
 0.609419010  0.560098000    1
 0.352923549  0.639530833    1
 0.313797682  0.428469344    1
 0.275593847  0.624510853    1
 0.310310776  0.757815199    1
```

```
0.200769573  1.068014129    1
0.393611386  0.489922085    1
0.293284180  0.564537846    1
0.150904334  0.874953285    1
0.359648477  0.984800311    1
0.425437016  0.605205704    1
0.550057275  0.953322346    1
0.369377777  0.717383758    1
0.483823544  0.776401643    1
0.665201554  0.609337149    1
0.367662676  0.432857589    1
0.603654120  0.439204275    1
0.361992913  0.607744455    1
0.365320313  0.193465958    1
0.565587013  0.766374185    1
0.459978544  0.421990201    1
0.389662454  0.697573566    1
0.662029374  0.545080251    1
0.193287037  0.660104813    1
0.770581129  0.678276952    1
0.517729293  0.709447233    1
0.666759179  0.738395921    1
0.507357601  0.504291821    1
0.074897782  0.726624656    1
0.267419803  0.669125800    1
0.570998498  0.905961669    1
0.234076185  0.680851488    1
0.204728441  0.915150466    1
0.463600872  0.831022543    1
0.551695270  0.877530083    1
0.375064997  0.706265086    1
0.548113044  0.683542273    1
0.436411367  0.523946916    1
0.171669265  0.706402907    1
0.228628170  0.696358973    1
0.258176000  0.750019031    1
0.427636052  0.726640752    1
0.551129128  1.041844415    1
0.382357212  0.485587245    1
0.627187520  0.857796470    1
0.759430378  0.897903714    1
0.385966401  0.649098802    1
0.216206061  0.886147391    1
0.107421934  0.525437056    1
0.466619974  0.649300564    1
0.483552867  0.519368234    1
```

```
0.188288155  0.704849311    1
0.123111648  0.618943465    1
0.149201404  0.674098357    1
0.541125439  0.641048950    1
0.707584972  1.048980926    1
0.250259605  0.738434506    1
0.388929309  0.980538827    1
0.163559795  0.768820434    1
0.290938989  0.858416660    1
0.671326658  0.887569891    1
0.419646183  0.833301601    1
0.297576300  0.815635781    1
0.488205349  0.928912516    1
0.274956333  0.622947292    1
0.364636103  0.552039161    1
0.020765563  0.400801476    1
0.503582267  0.462402974    1
0.129743512  0.478205376    1
0.205737679  0.652800375    1
0.491663362  0.919029482    1
0.541928820  0.592238748    1
0.352448258  0.438954474    1
0.340546986  0.610581184    1
0.087362845  0.722352081    1
0.544510425  0.310570940    1
0.426834451  0.697519317    1
0.505026501  0.203961507    1
0.393952243  0.701709243    1
0.341212359  0.487823226    1
0.443882109  0.515215865    1
0.216623801  0.641423278    1
0.325421774  0.565006133    1
0.339954219  0.500219969    1
0.757953402  0.646113630    1
0.166511560  0.675639720    1
0.394924171  0.795156547    1
0.581373272  0.769434777    1
0.469451043  0.686613394    1
0.180074959  0.917903510    1
0.314960733  0.919406796    1
0.781475499  1.074871466    1
0.261043992  0.883671133    1
0.149151175  0.475484999    1
0.236371870  0.975832107    1
0.646323770  0.522312176    1
0.518347874  0.876936157    1
```

```
 0.089471338  0.658664051  1
 0.498070451  0.902620720  1
 0.248059552  0.746906831  1
 0.550195316  0.737298487  1
 0.280602842  0.603132684  1
 0.431834416  0.533887741  1
 0.267799611  0.603699345  1
 0.507750995  0.826989974  1
-0.064478127  0.834070122  1
 0.342112413  0.661643764  1
 0.332313982  0.509083774  1
 0.665012582  0.878512787  1
 0.382910589  0.749228951  1
 0.361027556  0.645111929  1
 0.571981147  0.794214002  1
 0.536918322  0.898472992  1
 0.331872670  0.570367930  1
 0.044037168  0.476641964  1
 0.410716663  0.798924771  1
 0.455083777  0.551831167  1
 0.474594596  0.889946347  1
 0.413672127  0.867650039  1
 0.682171442  0.972182362  1
 0.425353451  0.535316350  1
 0.262277420  0.637457666  1
 0.007860344  0.806598462  1
 0.380999590  0.653580787  1
 0.538437280  0.907997360  1
 0.180415465  0.914334885  1
 0.237060285  0.752505492  1
 0.829663295  0.697894513  1
 0.307664951  1.074702414  1
 0.239849381  0.753987444  1
 0.275375404  0.806554305  1
 0.416984789  0.452953422  1
 0.476493007  0.858473259  1
 0.564497576  0.915314697  1
 0.198295169  0.534934547  1
 0.294198911  0.374100529  1
 0.684760671  0.892746414  1
 0.168075136  0.794230658  1
 0.502763522  0.712129784  1
 0.129722603  0.697110450  1
 0.285983065  0.796121883  1
 0.097239329  0.681159777  1
 0.210574775  0.792652629  1
```

```
  0.593896992  0.530407106    1
  0.358836790  0.671400853    1
  0.197591638  0.710584968    1
  0.540587182  0.774780451    1
  0.175106338  0.609394118    1
  0.448304389  0.663333083    1
  0.289880687  0.204721503    1
  0.300130047  0.934825869    1
  0.152511070  0.851596486    1
  0.495317475  0.631046756    1
  0.072423805  0.678667079    1
  0.500846416  0.689706961    1
  0.159104712  0.628206422    1
  0.710308164  0.777809751    1
  0.750642087  0.828037270    1
  0.559868855  0.783081248    1
  0.400801648  0.786167018    1
  0.356480531  0.911823818    1
  0.844132265  0.561509712    1
  0.426337951  0.777438407    1
  0.461052514  0.615763585    1
  0.205997206  0.785369909    1
  0.118613656  0.832647177    1
  0.444428480  0.747145725    1
  0.278467451  0.755943870    1
  0.329683958  0.704522943    1
  0.338924385  0.739418880    1
  0.427674817  0.962589298    1
  0.324169980  0.808410845    1
  0.526486063  0.856427139    1
  0.664857776  0.773954077    1
  0.327675416  0.608013752    1
  0.247589562  0.279270348    1
  0.418514564  1.044157214    1
  0.232314519  0.819642835    1
  0.762040971  0.573218465    1 ]  ;
%Alterando rotulo 0 -> -1
data(:,3)=2*data(:,3)-1;
teste(:,3)=2*teste(:,3)-1;
%Lendo dimensoes da matriz de Treinamento
[m,n]=size(data);
%Defindo a matriz de treinamento
Xtr=data(:,1:(n-1));
%Definindo rotulos da matriz de Treinamento
dtr=data(:,3);
%Invocando a funcao de treinamento
```

```
wt= training(Xtr,dtr);
%Deixamos o w otimo abaixo para comparcao
%wt = [1.280404020416833;   8.376034520281552;
    -4.331239328055023];
%Lendo as dimensoes da matriz de teste
[p,q]=size(teste);
%Definindo matriz de teste
Xte=teste(:,1:(q-1));
%Definindo rotulos da matriz de teste
dte=teste(:,3);
%Lendo novamente as dimensoes da matriz de teste
[p,q]=size(Xte);
%Inserindo coluna relativa ao bias
Xte(:,q+1)=1
%Variaveis para matriz de confusao
    vp= 0 ;
    vn = 0 ;
    fp = 0 ;
    fn = 0 ;
%Criacao da matriz de confusao
    for i=1: p
%Verificao de sinal
        sinal = 2 * (Xte(i,:)*wt > 0 )-1

        if ((sinal== (dte(i))) && (dte(i)<0))
            vn =  vn +1 ;
        elseif ((sinal== (dte(i))) && (dte(i)>0))
            vp = vp +1 ;
        elseif ((sinal~= (dte(i))) && (dte(i)<0))
            fp = fp +1 ;
        else
            fn = fn +1 ;
        end
    end
%Matriz de confusao
    C=[vp, fp; fn, vn];
    disp(C)
%Indices na matriz dos vetores de cada classe
indA=find(dte==-1);
indB=find(dte==1);
x=linspace(-1.5,1,5)
%Definindo a reta
y=-(wt(1)/wt(2))*x-(wt(3)/wt(2));
%Plotagem
plot(Xte(indA,1),Xte(indA,2),'+b',Xte(indB,1),Xte(indB,2),'or',x,y,
    '--k');
```

Apêndice C

Problema Dual das SVMs no Matlab

Antes de apresentarmos o código, vamos apresentar o tipo de problema que a rotina *quadprog* do Matlab resolve.

A função *quadprog* do Matlab tenta resolver por diferentes métodos um problema de otimização quadrática definido como:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \boldsymbol{f}^T\mathbf{x} \\ \text{sujeito a} \quad & \mathbf{A}\mathbf{x} < b \\ & \mathbf{A}_{eq}\mathbf{x} = b_{eq} \\ & lb \leq \mathbf{x} \leq ub \end{aligned}$$

Por outro lado, o problema que queremos resolver é:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & Q(\boldsymbol{\alpha}) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2}\{\sum_{i=1}^{P}\sum_{j=1}^{P} d_i d_j \alpha_i \alpha_j (\mathbf{x}_i.\mathbf{x}_j)\} \\ \text{sujeito a} \quad & 0 \leq \alpha_i \leq C, \quad para \quad i = 1, 2, 3, \ldots \\ & \sum_{i=1}^{P} \alpha_i.d_i = 0 \end{aligned}$$

Assim, vemos que para que nosso problema esteja de acordo com a rotina, é necessário transformar o problema de maximização num de minimização. Para tanto, multiplicamos a função objetivo por -1. Além disto, vemos que não temos os parâmetros A e b. Por outro lado, $\boldsymbol{f}$ deve ser um vetor com -1 em todas suas entradas.

É fácil notar também que $\mathbf{A}_{eq}$ é um vetor linha, onde cada posição $a_{1,j}$ é dada por d_j.Tem-se também que, $b_{eq} = lb = 0$. Como já discutimos, $ub = C$ depende do usuário ou de alguma heurística fornecida pelo mesmo. A matriz

H é uma matriz simétrica, positiva-definida. Sua construção pode ser vista no código.

Visto isto, estamos prontos para apresentar finalmente o algoritmo:

```
1
2  data=[%    xs            ys       yc
3    0.05100797  0.16086164    0
4   -0.74807425  0.08904024    0
5   -0.77293371  0.26317168    0
6    0.21837360  0.12706142    0
7    0.37268336  0.49656200    0
8   -0.62931544  0.63202159    0
9   -0.43307167  0.14479166    0
10  -0.84151970 -0.19131316    0
11   0.47525648  0.22483671    0
12   0.32082976  0.32721288    0
13   0.32061253  0.33407547    0
14  -0.89077472  0.41168783    0
15   0.17850119  0.44691359    0
16   0.31558002  0.38853383    0
17   0.55777224  0.47272748    0
18   0.03191877  0.01222964    0
19   0.25090585  0.30716705    0
20   0.23571547  0.22493837    0
21  -0.07236203  0.33376524    0
22   0.50440241  0.08054579    0
23  -0.63223351  0.44552458    0
24  -0.76784656  0.23614689    0
25  -0.70017557  0.21038848    0
26  -0.64713491  0.15921366    0
27  -0.76739248  0.09259038    0
28  -0.51788734  0.03288107    0
29   0.17516644  0.34534871    0
30  -0.68031190  0.47612156    0
31   0.01595199  0.32167526    0
32  -0.71481078  0.51421443    0
33   0.07837946  0.32284981    0
34  -0.80872251  0.47036593    0
35  -0.84211234  0.09294232    0
36  -0.98591577  0.48309267    0
37   0.29104081  0.34275967    0
38   0.24321541  0.51488295    0
39  -0.60104419  0.05060116    0
40  -1.24652451  0.45923165    0
41  -0.82769016  0.36187460    0
42  -0.62117301 -0.10912158    0
```

```
−0.70584105  0.65907662    0
 0.06718867  0.60574850    0
 0.30505147  0.47417973    0
 0.60788138  0.39361588    0
−0.78937483  0.17591675    0
−0.53123209  0.42652809    0
 0.25202071  0.17029707    0
−0.57880357  0.26553665    0
−0.83176749  0.54447377    0
−0.69859164  0.38566851    0
−0.73642607  0.11857527    0
−0.93496195  0.11370707    0
 0.43959309  0.41430638    0
−0.54690854  0.24956276    0
−0.08405550  0.36521058    0
 0.32211458  0.69087105    0
 0.10764739  0.57946932    0
−0.71864030  0.25645757    0
−0.87877752  0.45064757    0
−0.69846046  0.95053870    0
 0.39757434  0.11810207    0
−0.50451354  0.57196376    0
 0.25023622  0.39783889    0
 0.61709156  0.10185808    0
 0.31832860  0.08790562    0
−0.57453363  0.18624195    0
 0.09761865  0.55176786    0
 0.48449339  0.35372973    0
 0.52400684  0.46616851    0
−0.78138463 −0.07534713    0
−0.49704591  0.59948077    0
−0.96984525  0.46624927    0
 0.43541407  0.12192386    0
−0.67942462  0.30753942    0
−0.62529036  0.07099046    0
−0.02318116  0.40442601    0
 0.23200141  0.71066846    0
 0.09384354  0.46674396    0
 0.14234301  0.17898711    0
−0.61686357  0.25507763    0
 0.23636288  0.51543839    0
 0.38914177  0.40429568    0
−0.95178678 −0.03772239    0
 0.24087822  0.71948890    0
 0.12446266  0.45178849    0
−0.60566430  0.26906478    0
```

```
−0.71397188  0.30871780    0
 0.31008428  0.34675335    0
 0.18018786  0.46204643    0
−0.42663885  0.64723225    0
 0.06143230  0.32491150    0
 0.07736952  0.32183287    0
 0.42814970  0.13445957    0
−0.80250753  0.66878999    0
 0.40142623  0.42516398    0
 0.37084776  0.26407123    0
−0.80774748  0.41485899    0
 0.50163585  0.23934856    0
 0.58238323  0.22842741    0
−0.59136100  0.30230321    0
−0.87037236  0.26941446    0
−0.72086765  0.19676678    0
 0.27778443  0.21792253    0
 0.33240813  0.27349865    0
−0.14092068  0.39247351    0
−0.59759518  0.14790267    0
−0.85581534  0.14513961    0
−0.88912232  0.26896001    0
 0.21345680  0.43611756    0
−0.53467949  0.57901229    0
 0.31686848  0.39705856    0
−0.68121733  0.04209840    0
−0.97586127  0.45964811    0
 0.41457183  0.27141230    0
 0.32751292  0.36780137    0
−0.93209192  0.09362034    0
 0.58395341  0.47147282    0
−0.44437309  0.23010142    0
 0.29109441  0.19365556    0
−0.51080722  0.41496003    0
−0.96597511  0.17931052    0
 0.18741315  0.29747132    0
 0.17965417  0.45175449    0
−0.72689602  0.35728387    0
−0.54339877  0.41012013    0
−0.59823393  0.98701425    1
−0.20194736  0.62101680    1
 0.47146103  0.48221146    1
−0.09821987  0.58755577    1
−0.35657658  0.63709705    1
 0.63881392  0.42112135    1
 0.62980614  0.28146085    1
```

```
−0.46223286  0.61661031    1
−0.07331555  0.55821736    1
−0.55405533  0.51253129    1
−0.43761773  0.87811781    1
−0.22237814  0.88850773    1
 0.09346162  0.67310494    1
 0.53174745  0.54372650    1
 0.40207539  0.51638462    1
 0.47555171  0.65056336    1
−0.23383266  0.63642580    1
−0.31579316  0.75031340    1
−0.47351720  0.63854125    1
 0.59239464  0.89256953    1
−0.22605324  0.79789454    1
−0.43995011  0.52099256    1
−0.54645044  0.74577198    1
 0.46404306  0.51065152    1
−0.15194296  0.81218439    1
 0.48536395  0.82018093    1
 0.34725649  0.70813773    1
 0.43897015  0.62817158    1
−0.21415914  0.64363951    1
 0.57380231  0.63713466    1
 0.38717361  0.58578395    1
 0.32038322  0.53529127    1
−0.20781491  0.65132467    1
−0.18651283  0.81754816    1
 0.24752692  0.39081936    1
 0.66049881  0.89919213    1
−0.28658801  0.73375946    1
−0.32588080  0.39865509    1
−0.25204565  0.67358326    1
 0.37259022  0.49785904    1
−0.29096564  1.04372060    1
−0.30469807  0.86858292    1
−0.21389978  1.09317811    1
−0.36830015  0.75639546    1
−0.46928218  0.88775091    1
 0.39350146  0.77975197    1
−0.45639966  0.80523454    1
 0.51128242  0.76606136    1
 0.22550468  0.46451215    1
 0.01462984  0.40190926    1
−0.19172785  0.80943313    1
 0.38323479  0.75601744    1
 0.49791612  0.61334375    1
```

```
 0.35335230  0.77324337    1
-0.34722575  0.70177856    1
 0.58380468  0.76357539    1
-0.13727764  0.71246351    1
 0.38827268  0.44977123    1
-0.53172709  0.61934293    1
-0.11684624  0.87851210    1
 0.54335864  0.41174865    1
-0.45399302  0.66512988    1
-0.21913200  0.83484947    1
 0.30485742  0.98028760    1
 0.65676798  0.75766017    1
 0.61420447  0.75039019    1
-0.45809964  0.77968606    1
-0.21617465  0.88626305    1
-0.26016108  0.81008591    1
 0.31884531  0.84517725    1
-0.23727415  0.80178784    1
 0.58310323  0.77709806    1
 0.02841337  0.75792620    1
-0.41840136  0.68041440    1
 0.67412880  0.60245461    1
-0.25278281  0.70526103    1
 0.51609843  0.62092390    1
 0.20392294  0.91641482    1
-0.17207124  1.00884096    1
 0.27274507  0.29346977    1
 0.07634798  0.56222204    1
-0.36653499  0.64831007    1
 0.44290673  0.80087721    1
-0.19976385  0.54295162    1
-0.54075738  0.65293033    1
-0.07060266  1.00296912    1
 0.50715054  0.35045758    1
-0.06048611  0.62982713    1
 0.21532928  0.60260249    1
 0.46809108  0.87182416    1
-0.29888511  0.73669866    1
 0.86129620  0.47289330    1
 0.70120877  0.74572893    1
-0.11342797  0.60067099    1
 0.31234354  0.90756345    1
-0.12172541  0.84112851    1
 0.36867857  0.37052586    1
 0.57311489  0.40949740    1
-0.25841225  0.67192335    1
```

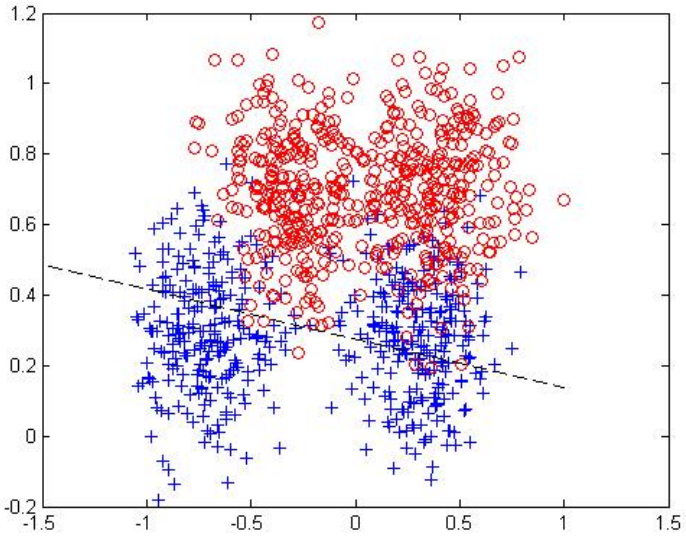
```
 0.30937186  0.50823318    1
 0.43319338  0.77016967    1
-0.30448035  0.57820106    1
 0.44276338  0.58023403    1
-0.19442057  0.89876808    1
-0.06105237  0.74184946    1
 0.07619347  0.35386246    1
 0.85826993  0.95819523    1
 0.37039200  0.72342401    1
 0.51481515  0.76203996    1
 0.43127521  0.54259166    1
 0.42286091  0.65242185    1
 0.29815001  0.93453682    1
 0.37128253  0.70089181    1
-0.51528729  0.76473490    1
 0.38525783  0.65528189    1
-0.34825368  0.50529981    1
 0.68510504  0.78067440    1
-0.36528923  0.45703265    1
-0.40903577  0.74230433    1
 0.43574387  0.44689789    1
 0.26887846  0.44559230    1
-0.49254862  1.01443372    1
 0.07615960  0.63795180    1
 0.49226224  0.46876241    1
-0.40249641  0.71301084    1 ] ;
%Lendo dimensoes
 [P,n] =size(data);
%Alterando rotulo 0 -> -1
 data(:,3)=2*data(:,3)-1;

 %Amostras
 X=data(1:P,1:(n-1)) ;
 Tdata = X;
%Rotulos
 d= data (1:P ,3 ) ;

 %Formacao da Matriz Simetrica H
 for i=1: P
     for j =1 :P
         xi = d(i) * X(i,:)  ;
         xj = d(j) * X(j,:) ;
         H (i,j) = dot(xi,xj) ;

     end
 end
```

```
%Definicao dos argumentos da Rotina quadprog
 f = -1 * ones(P,1) ;
 lb= zeros(P,1) ;
 ub= 100 *ones(P,1);
 beq = 0;
 Aeq =d';
 x= ones(P,1);
 x0 = x;
options.MaxIter = 2000;

 %Invocando a rotina quadprog
x = quadprog(H,f,[],[],Aeq,beq,lb,ub,x0,options) ;
multlagrange=x;
[P,n] =size(Tdata);
%W0 otimo
w0 = zeros(n,1) ;
for i=1: P
     amostra = Tdata(i,:)  ;
      w0 = amostra'*d(i)*x(i) + w0;
end

ind = find((d==1)&(x>0)) ;
%Encontrando os indices dos vetores com rotulo + 1
ds =   Tdata(ind,:);
%Heuristica para calculo de b0 otimo
b0 = mean(1- ds*w0) ;
[r,s] = size (w0) ;
%Adicao do bias ao vetor de pesos
w0(r+1,1) = b0 ;


%Leitura das Dimensoes dos dados
[p,q]=size(Tdata);
%Adicao da coluna relativa ao bias
Tdata(:,q+1)=1;
    vp= 0 ;
    vn = 0 ;
    fp = 0 ;
    fn = 0 ;
%Construcao da Matriz de Confusao
    for i=1: p
%Verificacao de Sinal
        sinal = 2 * (Tdata(i,:)*w0 > 0 )-1;
        l(i)=sinal ;
        if ((l(i)== (d(i))) && (d(i)<0))
            vn =  vn +1 ;
```

```
        elseif ((l(i)== (d(i))) && (d(i)>0))
            vp = vp +1 ;
        elseif ((l(i)~= (d(i))) && (d(i)<0))
            fp = fp +1 ;
        else
            fn = fn +1 ;
        end
    end
    %Matriz de Confusao
    C=[vp, fp; fn , vn];
    disp(C)
%Indices de pontos de cada classe
indA=find(d==-1);
indB=find(d==1);
x=linspace(-1.5,1);
%Hiperplano Otimo
y=-(w0(1)/w0(2))*x-(w0(3)/w0(2));
%Plotagem
plot(Tdata(indA,1),Tdata(indA,2),'+b',Tdata(indB,1),Tdata(indB,2),'
    or',x,y,'--k');
```

Usando os parâmetros ótimos advindos do código acima, podemos testar tal classificador linear para o conjunto de teste[1]:

[1] Vide código da fase de teste para obter o conjunto de dados de teste.



Fonte: Produção Própria.

Figura C.1: Vetores de teste classifcados por hiperplano ótimo de SVM

Apêndice D

Matrizes de Confusão

Uma matriz de Confusão é uma ferramenta amplamente usada quando se deseja testar a qualidade dos resultados obtidos via modelo em relação a aqueles conhecidos. Para o caso de duas classes[1], temos a seguinte configuração:

Tabela D.1: Matriz de Confusão

		Valores Previstos		
		p	**n**	**total**
Valores Reais	**p′**	Verdadeiro Positive	Falso Negativo	P′
	n′	Falso Positivo	Verdadeiro Negativo	N′
	total	P	N	

[1] Únicos casos explorados na monografia.

Apêndice E

Comparação entre Perceptron e SVM

Vamos nos fazer valer das matrizes de confusão para comparar os métodos de classificação vetores. Para tanto, vamos usar as matrizes de confusão dos dois métodos para o conjunto de vetores de teste.

Tabela E.1: Matriz de Confusão para conjunto de teste por Perceptron

		Valores Previstos	
		p	**n**
Valores Reais		494	6
		241	259

Já para o hiperplano obtido via SVM, temos que:

Tabela E.2: Matriz de Confusão para conjunto de Teste por SVMs

		Valores Previstos	
		p	**n**
Valores Reais		424	76
		37	463

Como o conjunto de testes, possui 1000 amostras, temos que a taxa de acerto é dada por:

$$TA = \frac{(VERDADEIRO\ \ POSITIVO\ \ +\ \ VERDADEIRO\ \ NEGATIVO)}{1000} \tag{E.1}$$

Então;

$$TA(SVM) = \frac{(463 + 464)}{1000} = 88,7\% \tag{E.2}$$

$$TA(Perceptron) = \frac{(429 + 494)}{1000} = 75,3\% \tag{E.3}$$

Com os resultados da matriz de confusão e a verificação da taxa de acerto, fica claro que o hiperplano encontrado através das máquinas de vetores de suporte é mais sensível à classificação correta dos dados.

Bibliografia

D.P Bertsekas. *Nonlinear Porogramming*. Athena Scientific, 2 edition, 1999.

Thomas M. Cover. Geometrical and statistical properties of linear treshold devices. 1964.

Simon Haykin. *Neural Networks: A comprehensive foundation*. Prentice Hall, 2 edition, 1999.

Tiago Prado Oliveira. Predição de tráfego usando redes neurais artificais para gerenciamento adaptativo de largura de banda de roteadores, 2014.

B.D Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 2001.

Hugo Leonardo Pereira Rufino. *Algoritmo de Aprendizado Supervisionado Baseado em Máquinas de Vetores de Suporte*. PhD thesis, 2011.

Ivan Nunes da Silva, Rogério Flauzino, and Danilo Hernane Spatti. *Redes neurais artificiais: para engenharia e ciências aplicadas*. Artliber, 1 edition, 2010.

J.S Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.