Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Planejamento de Trajetórias com Utilização de Máquinas de Vetor Suporte

Rafael de Oliveira Werneck

JUIZ DE FORA
DEZEMBRO, 2011

Planejamento de Trajetórias com Utilização de Máquinas de Vetor Suporte

Rafael de Oliveira Werneck

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Raul Fonseca Neto

JUIZ DE FORA

DEZEMBRO, 2011

Planejamento de Trajetórias com Utilização de Máquinas de Vetor Suporte

Rafael de Oliveira Werneck

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Raul Fonseca Neto
D.Sc. em Engenharia de Sistemas e Computação - COPPE/UFRJ

Saul de Castro Leite
D.Sc. em Modelagem Computacional - LNCC

Maicon Ribeiro Corrêa
D.Sc. Modelagem Computacional - LNCC

JUIZ DE FORA

09 DE DEZEMBRO, 2011

Aos pais, pelo carinho e dedicação.
Aos meus amigos e irmãos, pela força que eles me deram.

Resumo

O problema de planejamento de trajetórias em um conjunto de dados de obstáculos se resume a encontrar uma função de classificação tal que essa função separe os obstáculos em dois conjuntos tais que cada conjunto tenha um valor de rótulo diferente do outro. É interessante que essa função de classificação calculada (que será a trajetória que se deseja obter) seja a mais curta possível, pois essa solução representa a obtenção de um caminho mais rápido, e que também guarde uma margem de distância entre alguns obstáculos, garantindo que a trajetória obtida não colida com os obstáculos. Os algoritmos implementados combinam dois algoritmos, o primeiro, o Perceptron, traça a trajetória entre os dois conjuntos de dados de obstáculos, e o segundo, o Algoritmo de Margem Incremental, procura, à cada iteração, um valor de margem maior do que o anterior alcançado pelo Perceptron e pelo Algoritmo de Margem Incremental.

Palavras-chave: Planejamento de Trajetórias, Máquina de Vetor Suporte, Algoritmo de Margem Incremental.

Abstract

The problem of path planning in a data set of obstacles is to find a classification function that separates the obstacles in two sets such that each set has a label value different from the other. It is interesting that this calculated classification function (that is the path you want to obtain) is as short as possible, because this solution represents the achievement of a faster track, and also keep a margin of distances between some obstacles, ensuring that the path obtained does not collide with the obstacles. The implemented algorithms combine two algorithms, the first, the Perceptron, traces the path between the two data sets of obstacles, and the second, the Incremental Margin Algorithm demand, in each iteration, a margin value larger than the margin reached by the previous Perceptron and the Incremental Margin Algorithm.

Keywords: Path Planning, Support Vector Machine, Incremental Margin Algorithm.

Agradecimentos

Agradeço primeiramente a Deus, que me deu paciência e força para poder completar mais essa jornada em minha vida.

Aos meus pais, que sempre acreditaram em mim e me auxiliaram em tudo que eu precisei nesses anos.

Aos meus amigos, que sempre me apoiaram, e sem os quais esses anos de universidade não seriam tão bons assim.

Ao professor Raul pela orientação e incentivo à continuar com a pesquisa acadêmica.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos que ajudaram a expandir o conhecimento e me deram vontade de sempre continuar a aprender.

"O princípio da sabedoria é o desejo autêntico de instrução".

Sabedoria 6, 17

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
FMP	*Fixed Margin Perceptron*
IMA	*Incremental Margin Algorithm*
SVM	*Support Vector Machine*
TSVM	*Transductive Support Vector Machine*
UFJF	Universidade Federal de Juiz de Fora

1 Introdução

O problema de planejamento de trajetória é um problema importante, com principal aplicação para a área de robótica. O planejamento de trajetória tem por objetivo encontrar um caminho num espaço contendo obstáculos, tal que nesse caminho não ocorram colisões com os obstáculos. Essa trajetória dividirá o espaço em duas regiões, e essa divisão é interpretada como um problema de classificação (Miura, 2006). O *Support Vector Machine* (SVM) é utilizado para achar uma função que separe alguns obstáculos à direita e outros à esquerda da trajetória, de acordo com os rótulos dos pontos, que determinam de que lado o obstáculo estará da trajetória.

1.1 Objetivo deste trabalho

O presente trabalho tem por objetivo a elaboração de um algoritmo que auxilie no planejamento de trajetórias utilizando uma máquina de vetor suporte. A trajetória ideal deve possuir menor comprimento, para ser o caminho mais rápido para qualquer aplicação, e também guardar uma margem em relação a determinados obstáculos, garantindo que a trajetória esteja livre de colisões. Existem aplicações para esse problema tanto em $\mathbb{R}^2$ quanto para $\mathbb{R}^3$. Nesse trabalho, será focado somente o espaço com duas dimensões.

1.2 Estrutura da monografia

No capítulo 2 é mostrado como os conjuntos de dados dos obstáculos podem ser apresentados, e os tipos de aprendizado utilizado em cada um deles.

No capítulo 3 são explicadas as possíveis hipóteses de separabilidade dos pontos de obstáculos e os algoritmos para resolvê-los.

No capítulo 4 é apresentada a fundamentação teórica para o SVM, o histórico e sua formulação.

No capítulo 5 o *Transductive Support Vector Machine* (TSVM) é definido formal-

mente, é mostrado seu algoritmo e enumera-se porque ele funciona.

No capítulo 6 são descritas as etapas do algoritmo implementado.

No capítulo 7 é explicado o modo como foram criados os gráficos dos resultados.

No capítulo 8 são expostos os resultados obtidos pelo algoritmo construído.

No capítulo 9 apresenta-se as conclusões desse trabalho e sugestões para trabalhos futuros.

2 Conjunto de Dados dos Obstáculos

2.1 Dados Rotulados

O planejamento de trajetória num conjunto de dados de obstáculos no qual são conhecidos os rótulos, ou seja, de que lado da trajetória o obstáculo deverá ficar, pode ser resumido a um problema de encontrar uma função de classificação tal que separe os dados dos obstáculos de acordo com os seus rótulos. Para encontrar essa função de classificação, será utilizado o método de aprendizado supervisionado, que realiza uma inferência indutiva nos dados.

2.1.1 Inferência Indutiva

A filosofia clássica considera dois tipos de inferência. O primeiro, denominado indução, descreve o movimento de um caso particular (dados rotulados) para um caso geral (função de classificação), enquanto o segundo, denominado dedução, descreve o movimento de um caso geral (função de classificação) para um caso mais particular (rotular os dados).

O aprendizado por inferência indutiva é realizado em dois estágios: o primeiro, onde é estimada a função de acordo com os dados para os quais já se possui o rótulo; e o segundo estágio, em que, com os dados sem rótulos, classifica-os de acordo com a função estimada no primeiro estágio. Esse cenário utilizando dois estágios tenta resolver um problema simples, que é a classificação de acordo com a função, resolvendo primeiramente um muito mais difícil, que é estimar a função para os dados rotulados.

2.1.2 Aprendizado Supervisionado

O aprendizado supervisionado refere-se ao aprendizado utilizando exemplos de treinamento, no caso, um conjunto de dados onde é conhecido o rótulo correto. Se o algoritmo possui uma quantidade suficiente de dados rotulados, o classificador pode ser treinado na correta predição dos rótulos desses dados e, assim, generalizar a classificação para os

dados não rotulados. Entretanto, esse método pode sofrer de *overfitting*, diminuindo a sua capacidade de generalização (Handl et al., 2006).

2.2 Dados Não Rotulados

Para o planejamento de trajetórias, pode-se ter um conjunto de dados contendo algumas amostras não rotuladas. Com isso, é possível ter como objetivo restringir a margem sobre os obstáculos tal que essa restrição seja a menor distância entre a trajetória e os obstáculos. Um exemplo de aplicação para esse tipo de conjunto de dados seria um rebocador de navios que conhece os obstáculos e a disposição destes no caminho, e assim poderia ser traçada uma rota que passasse a uma distância considerável dos obstáculos. Para essa resolução, será utilizado o método de aprendizado semi-supervisionado.

2.2.1 Inferência Transdutiva

A inferência indutiva, apresentada na seção 2.1.1, utiliza os dados sobre os quais se tem informações para encontrar a função de classificação. Entretanto, nem sempre se possui uma quantidade de informações suficiente para poder estimar bem essa função. Com isso, Vapnik propôs, nos anos 90, um outro tipo de inferência, a transdutiva, que seria o processo de estimar os rótulos dos pontos de interesses à partir dos exemplos rotulados, ou seja, saindo de um caso particular (exemplos) para outro caso particular (rótulos dos pontos de interesse), evitando, assim, a resolução de um problema difícil, que era estimar a função para os dados conhecidos. “Se você está limitado a uma quantidade restrita de informações, não resolva o seu problema específico resolvendo um problema mais geral” (Vapnik).

Na figura 2.1, é exemplificado os dois tipos de inferência;

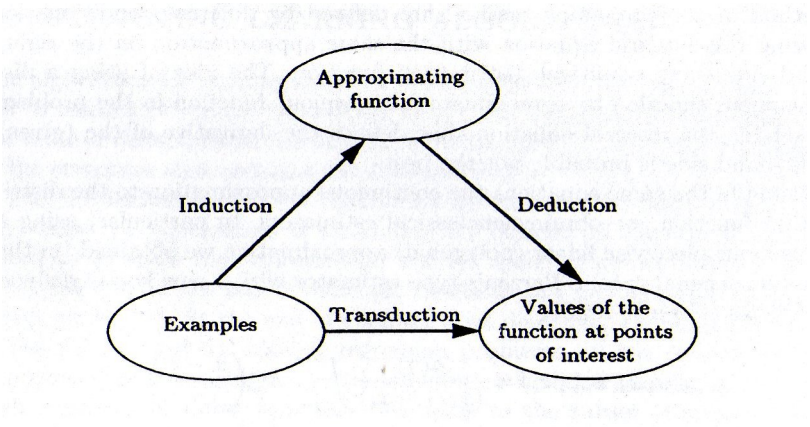


Figura 2.1: Diferentes tipos de inferência, retirado de (Vapnik, 2000).

2.2.2 Aprendizado Semi-Supervisionado

Para se explicar o método do aprendizado semi-supervisionado, será apresentado, primeiramente, o aprendizado não supervisionado.

Não Supervisionado

Contrastando com o aprendizado supervisionado, o não supervisionado pode ser aplicado na ausência de qualquer conhecimento acerca dos rótulos, ou de exemplos de treinamento. Ele se apoia no fato de que a estrutura das classes dos dados se refletem na atual distribuição desses dados. Obviamente, esse método falha se não existir uma estrutura com os dados, sendo assim, menos poderoso que o método supervisionado. Porém, o método pode ser usado para explorar cenários onde existe pouca informação sobre os dados, além disso, ele também não é afetado por *overfitting* (Handl et al., 2006).

Semi-Supervisionado

O aprendizado semi-supervisionado combina as vantagens do aprendizado supervisionado e do não supervisionado, em que explora tanto o conhecimento dos dados rotulados como da distribuição espacial dos dados. A combinação de dados rotulados e não rotulados torna possível orientar o algoritmo de classificação, enquanto ele leva em consideração a estrutura dos dados utilizados. Isso é útil quando se lida com uma grande quantidade de dados não rotulados e poucos dados rotulados (Handl et al., 2006).

3 Hipóteses de Separabilidade dos Obstáculos

O modo como os dados dos obstáculos estão dispostos no espaço de representação influencia na formulação e no *kernel* utilizados para o cálculo da trajetória pelo algoritmo, a saber:

3.1 Hipóteses Lineares

Para as hipóteses lineares, ou seja, o conjunto de dados de obstáculos que podem ser separados por uma reta, utiliza-se o *kernel* produto interno com a formulação primal ou dual do SVM para conjuntos de dados com todos os exemplos rotulados e da formulação dual do TSVM para conjunto de dados parcialmente rotulados.

Nos capítulos 4 e 5 são apresentadas as duas formulações desses métodos.

3.2 Hipóteses Não Lineares

Porém, caso os dados dos obstáculos não possam ser separados por uma reta, faz-se necessário o uso de outro *kernel* tal que ele resulte numa função de classificação que separe os conjuntos de dados corretamente. Nessa caso, utiliza-se um *kernel* polinomial de grau d, $d \geq 2$, juntamente com a formulação dual do algoritmo a ser utilizado (SVM ou TSVM).

4 Máquina de Vetor Suporte

4.1 Histórico

A capacidade de encontrar padrões no mundo que o cerca é da natureza do ser humano, que a executa sem maiores dificuldades (Semolini, 2002), diferentemente da matemática utilizada pelos computadores, que ainda é bem recente. De acordo com (Principe et al, 1999) foi nos anos 30 que R. A. Fisher estabeleceu os princípios matemáticos do reconhecimento estátistico de padrões.

Já a pesquisa computacional começou na década de 60, sendo criados vários métodos desde então, como Redes Neurais Artificiais, Algoritmos Genéticos, e outros. Em 1992, um grupo da AT&T Bell Laboratories desenvolveu outro método de classificação utilizando margens ótimas, que foi aperfeiçoada por Cortes & Vapnik (1995), quando o método passou a ser conhecido como Máquinas de Vetor Suporte (SVM) (Carvalho, 2005).

4.2 SVM Linear

De acordo com (Ormonde, 2009), o SVM Linear pode ser formalizado da seguinte maneira:

Seja um conjunto com p elementos utilizado para treinar o algoritmo, tal que esses elementos estejam mapeados como vetores $\overrightarrow{x_i}$ e rótulos y_i, com $1 \leq i \leq p$.

$$(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \cdots, (\overrightarrow{x_p}, y_p) \tag{4.1}$$

e que os rótulos assumem os seguintes valores:

$$\begin{cases} y_k = 1, & \text{se } x_k \in \text{Classe 1} \\ y_k = -1, & \text{se } x_k \in \text{Classe -1} \end{cases}$$

Os dados do conjunto de treinamento serão linearmente separáveis se existir um vetor $\overrightarrow{w}$ e um escalar b tal que as seguintes inequações são válidas para todos os elementos desse conjunto.

$$\begin{cases} \overrightarrow{w}.\overrightarrow{x_i} + b \geq 1, & \text{se } \ y_i = 1 \\ \overrightarrow{w}.\overrightarrow{x_i} + b \leq -1, & \text{se } \ y_i = -1 \end{cases} \tag{4.2}$$

As duas inequações de 4.2 podem ser combinadas na seguinte equação:

$$y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1, \qquad i = 1, \cdots, p \tag{4.3}$$

Existe somente um hiperplano tal que os dados do conjunto de treinamento (4.1) são separados dele por uma margem máxima. Este hiperplano determina a direção de $\overrightarrow{w}$ em que é máxima a distância entre as projeções dos vetores dos dados das duas diferentes classes. Expressa-se a distância pela seguinte fórmula:

$$\delta = \min_{x_i:y_i=1} \frac{\overrightarrow{x_i} \cdot \overrightarrow{w}}{||\overrightarrow{w}||} - \max_{x_i:y_i=-1} \frac{\overrightarrow{x_i} \cdot \overrightarrow{w}}{||\overrightarrow{w}||} \tag{4.4}$$

O hiperplano em que a distância (4.4) é máxima é denominado hiperplano ótimo. Utilizando (4.3) e (4.4), tem-se:

$$\delta = \frac{2}{||\overrightarrow{w}||} = \frac{2}{\sqrt{\overrightarrow{w} \cdot \overrightarrow{w}}} \tag{4.5}$$

Isso diz que o hiperplano que maximiza δ é único, e pode ser maximizada minimizando $||\overrightarrow{w}||^2$, obedecendo as restrições de (4.3).

$$\begin{aligned} \text{Min} \quad & \frac{1}{2}||w||^2 \\ \text{Sujeito a:} \quad & y_i(\langle w, x_i \rangle + b) \geq 1 \\ & \text{para } i = 1, \cdots, m \end{aligned}$$

Também pode-se escrever o problema de minimizar $||\overrightarrow{w}||^2$ utilizando os multiplicadores de Lagrange, pois as restrições (4.3) substituídas pelas restrições dos multiplicadores são mais simples de se trabalhar, além de que, com essa reformulação do problema, os dados de treinamento somente apareceram sobre a forma de produto interno entre os vetores, que é de fundamental importância para a generalização para problemas não

lineares.

Essa reformulação consiste em introduzir os multiplicadores de Lagrange $\alpha_i, \quad i = 1, \cdots, p$, um para cada restrição das inequações de 4.3. Resultando no seguinte problema primal:

$$\text{Minimizar: } L_P = \frac{||\overrightarrow{w}||^2}{2} - \sum_{i=1}^{p} \alpha_i y_i (\overrightarrow{x_i} \cdot \overrightarrow{w} + b) + \sum_{i=1}^{p} \alpha_i \tag{4.6}$$

tal que:

$$\alpha_i \geq 0$$

Como esse é um problema de programação quadrática convexo, então o problema dual equivalente obterá a mesma solução. Transformando o problema primal em dual, tem-se o seguinte problema de otimização:

$$\text{Maximizar: } L_D = \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j (\overrightarrow{x_i} \cdot \overrightarrow{x_j}) \tag{4.7}$$

de acordo com as seguintes restrições:

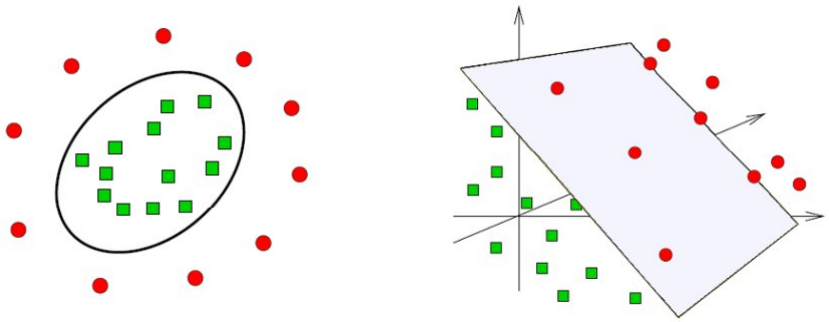
$$\begin{aligned} \sum_{i=1}^{p} \alpha_i y_i &= 0 \\ \alpha_i &\geq 0 \end{aligned}$$

4.3 SVM Não Linear

Até o presente momento, levou-se em consideração somente o caso onde os dados pudessem ser linearmente separáveis. Porém, na prática, muitos problemas reais não são linearmente separáveis, limitando o poder de computação desse algoritmo (Ormonde, 2009).

Porém, este problema pode ser resolvido e o SVM generalizado para o caso não linear. Primeiramente, tem-se o fato de que os dados de treinamento aparecem na equação (4.7) como produto interno dos vetores. Assim, se for possível calcular o produto interno de dois vetores em um espaço euclidiano hiperdimensional qualquer, a função de decisão

poderá ser calculada, e a separação linear por um hiperplano será possível.



(a) Problema original (b) Problema mapeado

Figura 4.1: Mapeamento dos dados para uma dimensão maior

Seja um mapeamento Φ que mapeia os dados de treinamento de um espaço de entrada de d dimensões para um espaço de características da seguinte maneira:

$$\Phi : \mathbb{R}^d \longmapsto \Omega \tag{4.8}$$

Com isso, todo o treinamento e execução do algoritmo como classificador vai depender dos produtos internos dos vetores no espaço Ω. Assim, a equação (4.7) pode ser reescrita como:

$$\text{Maximizar: } L_D = \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j (\Phi(\overrightarrow{x_i}) \cdot \Phi(\overrightarrow{x_j})) \tag{4.9}$$

Agora, tem-se o problema de se calcular $\Phi(\overrightarrow{x_i}) \cdot \Phi(\overrightarrow{x_j})$, porém isso também pode ser feito de maneira bastante simples e eficiente. Ao invés de calcular $\Phi(\overrightarrow{x_k})$ para cada vetor k, computa-se o valor de $\Phi(\overrightarrow{x_i}) \cdot \Phi(\overrightarrow{x_j})$ através de uma função que usa diretamente os dados de entrada. Uma função que possui essa capacidade é denominada função *Kernel*.

Uma função *Kernel* é uma função K tal que, para todo $\overrightarrow{x_i}, \overrightarrow{x_j} \in \mathbb{R}^d$, satisfaz:

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = \Phi(\overrightarrow{x_i}) \cdot \Phi(\overrightarrow{x_j}) \tag{4.10}$$

Assim sendo, a equação 4.9 pode ser reescrita para obter a equação de um SVM Não Linear:

$$\text{Maximizar: } L_D = \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j K(\overrightarrow{x_i}, \overrightarrow{x_j}) \tag{4.11}$$

É interessante notar que, com o uso das funções *kernel*, pode-se calcular um SVM para espaços de grandes dimensões sem ter de trabalhar diretamente com o mapeamento dos dados.

4.4 Características

(Lorena et al., 2003) cita algumas das características que tornam o SVM atrativo:

- Boa capacidade de generalização: A capacidade de generalização de um classificador é medido de acordo com sua eficiência ao classificar dados que estão fora de seu conjunto de treinamento. As SVMs, em geral, alcançam bons resultados de generalização.
- Teoria bem definida: A base teórica do SVM é bem estabelecida.
- Robustez de grandes dimensões: O SVM é bastante utilizado em imagens, devido a essa sua robustez.
- Convexidade da função objetivo: diferentemente das Redes Neurais Artificiais, em que há a presença de mínimos locais na função objetivo, a aplicação do SVM implica na otimização quadrática, com somente um mínimo.

5 Máquina de Vetor Suporte Transdutivo

O SVM transdutivo é um algoritmo proposto por Vapnik para se aproveitar tanto do conjunto de treinamento rotulado quanto do conjunto de teste não rotulado durante o tempo de predição. Foi assim nomeado porque Vapnik provou que os limites na performace generalizada fornecida pelo conjunto de testes foi superior ao baseado na indução utilizando somente o conjunto de treinamento rotulado (Karlen et al., 2006).

5.1 Definição Formal

5.1.1 TSVM Primal

(WikiBooks, 2011) define, formalmente, o TSVM pelo seguinte problema de otimização primal:

$$
\begin{aligned}
&Min(w, b, c^*) \\
&\quad \frac{1}{2}||w||^2 \\
&\text{Sujeito a, para qualquer } i = 1, \cdots, n \text{ e } j = 1, \cdots, k \\
&\quad c_i(\langle w, x_i \rangle - b) \geq 1, \\
&\quad c_j^*(\langle w, x_j^* \rangle - b) \geq 1, \\
&\quad c_j^* \in \{-1, 1\}
\end{aligned}
\tag{5.1}
$$

sendo c_j o rótulo do dado x_j, b o bias, que é a movimentação do hiperplano paralelamente a ele mesmo, para ajustar o classificados, e w o vetor que define a função de classificação.

5.1.2 TSVM Dual

Baseando-se na definição do SVM Dual para margem *soft* com bias zero, a formulação dual do problema de otimização do TSVM é escrita como a minimização das funções de custo do SVM Dual, que é o inverso da margem mais os erros de treinamento, através da matriz de rótulos Γ:

$$\begin{aligned} \min_{\Gamma} \max_{\alpha} \quad & 2\alpha' e - \alpha'(\mathbf{K} \odot \Gamma)\alpha \\ \text{Sujeito a} \quad & C \geq \alpha_i \geq 0 \\ & \Gamma = \begin{pmatrix} y^t \\ y^w \end{pmatrix} . \begin{pmatrix} y^t \\ y^w \end{pmatrix}' \\ & y_i^w \in \{1, -1\} \end{aligned}$$

A matriz simétrica Γ é parametrizada pelo vetor dos dados não rotulados $y^w \in \{-1, 1\}^{n_w}$, sendo n_w o tamanho do conjunto de testes. O vetor $y^t \in \{-1.1\}^{n_t}$, sendo n_t o número de pontos de treinamento, é dado pelos rótulos conhecidos dos pontos de treinamento. A matriz simétrica $\mathbf{K} \in \mathbb{R}^{(n_w+n_t)\times(n_w+n_t)}$ é a matriz de *kernel* sobre o conjunto de treinamento e o conjunto de teste. O vetor dual é denominado por $\alpha \in \mathbb{R}^{n_w+n_t}$, e $\mathbf{e}$ é um vetor de tamanho adequando contendo todos os elementos. O símbolo $\odot$ representa o produto das matrizes elemento a elemento. Esse é um problema de combinatória, em que a complexidade cresce exponencialmente com o tamanho do conjunto de testes (De Bie et al., 2003).

5.2 Algoritmo Heurístico

Algoritmo 1: Algoritmo *Branch and Bound*

Entrada: Y, ub
Saída: Y^*, v
início
  se $\sum \max(0, Y_i) > ur \ \ OU \sum \max(0, -Y_i) < n - ur$ **então**
    retorna
  fim se
  $v \leftarrow \text{SVM}(Y)$
  se $v > ub$ **então**
    retorna
  fim se
  se Y *é totalmente rotulado* **então**
    $Y^* \leftarrow Y$
    retorna
  fim se
  Encontra o próximo ponto a ser rotulado
  $Y_i \leftarrow -y$
  $(Y^*, v) \leftarrow \text{TSVM}(Y, ub)$
  $Y_i \leftarrow -Y_i$
  $(Y_2^*, v_2) \leftarrow \text{TSVM}(Y, \min(ub, v))$
  se $v_2 < v$ **então**
    $Y^* \leftarrow Y_2^*$
    $v \leftarrow v_2$
  fim se
fim

sendo Y um vetor com os dados, ub um limite máximo para o valor objetivo, Y^* o vetor ótimo com todos os rótulos e v a função objetiva (Chapelle et al., 2007).

(Gammerman et al., 1998) descreve o algoritmo transdutivo da seguinte maneira:

Considerando dois cenários no espaço $\mathbb{R}^n$, ambas os cenários contendo $(p + 1)$ dados, sendo p pontos do conjunto de treinamento e 1 ponto a ser classificado. Com os pontos de treinamento já rotulados, a única diferença entre os cenários é a classificação do $(p + 1)$-ésimo ponto, que está rotulado como -1 no primeiro cenário e +1 no segundo cenário. Pode ser provado que o ponto $p + 1$ será suporte em pelo menos um desses cenários. Seja SV(1) (e, respectivamente, SV(-1)) o conjunto de índices dos suportes no primeiro cenário (respectivamente, no segundo cenário) e $\#A$ a cardinalidade do conjunto A. O algoritmo dá as seguintes predições e incertezas:

- 1 se

$$((p+1) \in \mathrm{SV}(-1)) \ \ \& \ \ ((p+1) \notin \mathrm{SV}(1))$$

ou

$$((p+1) \in \mathrm{SV}(-1) \cap \mathrm{SV}(1) \ \ \& \ \ (\#\mathrm{SV}(-1) < \#\mathrm{SV}(1))$$

a incerteza nessa predição é

$$\frac{\#\mathrm{SV}(-1)}{p+1}$$

- -1 se

$$((p+1) \in \mathrm{SV}(1)) \ \ \& \ \ ((p+1) \notin \mathrm{SV}(-1))$$

ou

$$((p+1) \in \mathrm{SV}(-1) \cap \mathrm{SV}(1) \ \ \& \ \ (\#\mathrm{SV}(1) < \#\mathrm{SV}(-1))$$

com a incerteza

$$\frac{\#\mathrm{SV}(1)}{p+1}$$

- Qualquer predição se

$$((p+1) \in \mathrm{SV}(-1) \cap \mathrm{SV}(1) \ \ \& \ \ (\#\mathrm{SV}(-1) = \#\mathrm{SV}(1))$$

com a incerteza

$$\frac{\#\mathrm{SV}(-1)}{p+1} = \frac{\#\mathrm{SV}(1)}{p+1}$$

5.3 Por que o TSVM Funciona

(Wu et al., 2003) enumera alguns itens que explica porquê que o TSVM funciona:

- Fornece dados não rotulados para teste e validação;
- Possui precisão como prioridade;
- Fornece rótulos provisórios para os dados não rotulados;

6 Algoritmo de Margem Incremental Transdutivo

Para a elaboração do Algoritmo de Margem Incremental Transdutivo, foi utilizado como base os algoritmos apresentado em (Leite et al., 2008).

Esse algoritmo é composto por duas principais funções. A primeira, um Perceptron de Margem Fixa, que tem por objetivo encontrar a solução de um problema a partir da margem fixa. A segunda, um Algoritmo de Margem Incremental, incrementa gradualmente o valor da margem fixa calculado pelo FMP. O algoritmo tem por vantagem sempre possuir uma solução em mãos (Leite et al., 2008).

O pseudo-código para o algoritmo implementado é o seguinte:

Algoritmo 2: Pseudo-código do Algoritmo Implementado

1. Treinar o SVM somente com os dados rotulados;
2. Rotular os dados não rotulados de acordo com a função sinal utilizada;
3. Treinar todos os dados;
4. Repetir os passos 2 e 3 até que o número máximo de iterações seja alcançado.

6.1 Perceptron

O perceptron proposto por Rosenblatt é a forma mais simples de uma rede neural usada para a classificação de padrões linearmente separáveis. Ele consiste em um único neurônio com os pesos e bias ajustáveis. Rosenblatt provou que, se os padrões usados no treinamento do perceptron são provenientes de duas classes linearmente separáveis, então o algoritmo converge e coloca a superfície de decisão com a forma de um hiperplano entre as duas classes.

Esse perceptron é utilizado para se determinar o vetor w em um limitado número de iterações. Esse número de iterações está intimamente relacionado à quantidade de vezes que o vetor de pesos é atualizado e, por conseguinte, à quantidade de erros cometido pelo

algoritmo (Fonseca, 2006; Haykin, 2009).

(Grudic, 2005) explica o algoritmo do perceptron da seguinte maneira:

Algoritmo 3: Algoritmo do Perceptron

1. Iniciar com um hiperplano aleatório;
2. Movimentar gradativamente o hiperplano até que os pontos classificados erroneamente por ele se aproximem da lado correto da fronteira;
3. Parar quando todos os exemplos de treinamento estão corretamente classificados.

Para um conjunto de dados de treinamento, ocorre um erro se:

$$y_i(\langle w, x_i \rangle + b) < 0 \tag{6.1}$$

Assim, pode-se adotar como função de perda a quantidade de exemplos incorretamente classificados. Essa função, definida como função de perda 0-1 e linear por partes, é definida como:

$$J(w) = \sum_i \text{Max } \{0, -y_i(\langle w, x_i \rangle + b)\}, \qquad (x_i, y_i) \in Z \tag{6.2}$$

sendo Z o conjunto de treinamento.

6.2 Perceptron de Margem Fixa

6.2.1 Perceptron Primal

Duda, Hart e Stork (2001) propõem uma versão para o algoritmo do perceptron que inclui a utilização de uma regra de incremento variável para a função de perda quadrática e um valor de margem fixo γ para adaptar a sua solução ao método de relaxação. Considerando a introdução do novo parâmetro γ, a nova solução do problema consiste em determinar a solução viável do sistema de inequações lineares na forma:

$$y_i(\langle w, x_i \rangle + b) \geq \gamma \tag{6.3}$$

Porém, caso se utilize uma regra de incremento fixo e não seja possível limitar o valor da norma quadrática do vetor w com a adição da restrição adicional de normalização

$||w||_2 = 1$, o sistema de inequações, caso ele seja linearmente separável, apresentará uma solução viável considerando o crescimento da norma e do valor do produto interno, para qualquer margem γ. Para conseguir resolver esse problema necessita-se estabelecer alguma forma de regularização para limitar o valor da norma do vetor w.

Por isso, procurou-se uma nova formulação para o perceptron que garantisse que o conjunto de dados guardasse uma distância mínima em relação ao hiperplano sem limitar o valor da norma de w.

Para tanto, considera-se a restrição de que cada amostra de dados deve possuir um valor de margem geométrica igual ou superior ao valor da margem fixa, sendo que o valor da margem geométrica é definido como o valor da margem funcional da respectiva amostra dividido pela norma euclidiana do vetor w.

Nesse sentido, resolve-se o seguinte sistema de inequações não lineares para determinado valor de margem fixa, representado por γ_f:

$$\begin{aligned} \frac{y_i(\langle w, x_i\rangle + b)}{||w||_2} &\geq \gamma_f \qquad \text{ou} \\ y_i(\langle w, x_i\rangle + b) &\geq \gamma_f ||w||_2 \end{aligned} \tag{6.4}$$

Por causa dessa modificação 6.4, é necessário reescrever a função de perda 6.2 de forma que ela possibilite a obtenção de uma nova regra de correção.

$$J(w) = \sum_i \text{Max}\left\{0, \gamma_f - \frac{y_i(\langle w, x_i\rangle + b)}{||w||_2}\right\}, \qquad (x_i, y_i) \in Z \tag{6.5}$$

Portanto esse definição, ao contrário do algoritmo básico do perceptron, considera como erro os exemplos que não estão a uma distância mínima do hiperplano, apesar de eles estarem classificados corretamente.

A solução do sistema de inequações é aquela que minimiza a função de erro 6.5. Assim, tomando-se o gradiente da função em relação ao vetor w, caso ocorra um erro

$$y_i(\langle w, x_i\rangle + b) < \gamma_f ||w||_2 \tag{6.6}$$

em uma amostra $(x_i, y_i) \in M$, sendo M o conjunto de amostras classificadas

incorretamente, tem-se a seguinte regra de correção:

$$w_{t+1} = w_t - \eta\left(\gamma_f \frac{w}{||w||_2} - x_i y_i\right) \tag{6.7}$$

em que η é referente à taxa de aprendizagem.

Algoritmo 4: Perceptron de Margem Fixa Primal

Entrada: $z_m, w_{init}, \gamma_f, \eta, T_{MAX}$
Saída: w^t
início
  $w^0 \leftarrow w_{init}$;
  $t \leftarrow 0$;
  repita
    para *(*$i = 1, \cdots, p$*)* **faça**
      se *(*$y_i(\langle x_i, w^t\rangle + b) < \gamma_f||w^t||$*)* **então**
        $w^{t+1} \leftarrow w^t\lambda_t + \eta y_i x_i$;
        $t \leftarrow t + 1$;
      fim se
    fim para
  até *(não houver erro) ou (*$t > T_{MAX}$*)*;
fim

6.2.2 Perceptron Dual

Para que o algoritmo do Preceptron de Margem Fixa (FMP) tenha o poder de classificação de uma máquina *kernel*, torna-se necessário que ele seja desenvolvido no plano de variáveis duais. Para isso, o vetor w é representado em variáveis duais como se segue (Leite et al., 2008):

$$w = \sum_{i=1}^{p} \alpha_i y_i x_i, \quad \text{para } \alpha_i > 0 \tag{6.8}$$

Para erros de classificação, a regra de atualização das variáveis duais é a seguinte:

$$\alpha_i = \alpha_i + \eta \tag{6.9}$$

Pode-se construir a nova atualização do FMP da regra de atualização das variáveis primais. Observa-se que o vetor w^t é multiplicado por um fator λ_t antes da correção ser efetivada. Essa multiplicação em variáveis duais é dada por:

$$w^t \lambda_t = \sum_{i=1}^{p} \lambda_t \alpha_i y_i x_i \tag{6.10}$$

Como somente o α_i não é uma constante, essa multiplicação é equivalente a multiplicar os valores de α_i por λ_t. O fator de multiplicação λ_t tem um efeito interessante. Por escalonar para baixo cada multiplicador antes das atualizações, na verdade, está sendo feita a escolha dos vetores suporte. Ou seja, os multiplicadores que foram modificados no início do algoritmo e não estão associados a nenhum vetor suporte tem seu valor levados a zero.

Define-se a norma $||w||$ por 6.8 da seguinte maneira:

$$\begin{aligned} ||w||^2 &= \left\langle \sum_{i=1}^{p} \alpha_i y_i x_i, \sum_{j=1}^{p} \alpha_j y_j x_j \right\rangle \\ &= \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ &= \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \end{aligned} \tag{6.11}$$

Para uma maior eficiência computacional, é possível calcular o valor de 6.11 após cada atualização. Por exemplo, após cada atualização por erros de classificação, $||w^{t+1}||$ pode ser descrito pela seguinte regra:

$$\begin{aligned} ||w^{t+1}||^2 &= \alpha_t^2 \langle w^t, w^t \rangle + 2\eta \alpha_t y_i \langle w^t, x_i \rangle + \eta^2 \langle x_i, x_i \rangle \\ &= \alpha_t^2 ||w^t||^2 + 2\eta \alpha_t y_i \langle w^t, x_i \rangle + \eta^2 K(x_i, x_i) \end{aligned} \tag{6.12}$$

em que $y_i \langle w^t, x_i \rangle = y_i \sum_{j=1}^{p} \alpha_j^t y_j K(x_j, x_i)$ já foi computado. (Leite et al., 2008) define o seguinte algoritmo final para o FMP Dual:

Algoritmo 5: Perceptron de Margem Fixa Dual

Entrada: $z_m, \alpha_{init}, \gamma_f, \eta, T_{MAX}$
Saída: α^t
início
  $\alpha^0 \leftarrow \alpha_{init}$;
  $t \leftarrow 0$;
  repita
    para *(*$i = 1, \cdots, p$*)* **faça**
      se $\left(y_i \sum_{j=1}^{p} \alpha_j y_j K(x_j, x_i) \ < \ \gamma_f ||w^t|| \right)$ **então**
        $\alpha^{t+1} \leftarrow \lambda^t \alpha_t$;
        $\alpha_i^{t+1} \leftarrow \alpha_i^t + \eta$;
        $t \leftarrow t + 1$;
      fim se
    fim para
  até *(não houver erro) ou (*$t > T_{MAX}$*)*;
fim

6.3 Algoritmo de Margem Incremental

Com a finalidade de se obter uma aproximação da máxima margem, foi elaborado um algoritmo que obtem sucessivas soluções do FMP para valores de margem cada vez maiores. Essa margem fixa, denominada γ_f, inicia-se com valor 0 e é incrementada a cada iteração até que se consiga atingir um valor de margem próximo do ótimo (Leite et al., 2008).

Ou seja, para um conjunto de valores $\gamma_f \in [0, \cdots, \gamma^*)$, sendo:

$$\gamma_f^{t+1} > \gamma_f^t \tag{6.13}$$

para $t = 1, \cdots, T-1$, $\gamma_f^1 = 0$, com $\gamma_f^T \approx \gamma^*$, soluciona-se, pelo método de relaxação, o seguinte problema de inequações não lineares:

$$y_k f(x_k) \geq \gamma_f ||w||_2, \qquad k = 1, \cdots, p \tag{6.14}$$

em que cada solução é equivalente à solução do FMP.

Para a atualização do valor da margem fixa a cada iteração, adota-se as seguintes duas regras que garantem um número finito de correções e a convergência para a solução máxima.

Primeira regra: caso a solução encontrada pelo FMP forneça as margens positiva e negativa diferente uma da outra, diz-se que essa solução não caracteriza uma solução de máxima margem. Então, é necessário corrigir o valor da margem fixa da seguinte forma:

$$\gamma_f^{t+1} = \frac{\gamma^+ + \gamma^-}{2} \tag{6.15}$$

onde γ^+ e γ^- são os valores das menores distâncias dos pontos dos conjuntos definidos pelas classes $+1$ e -1 ao hiperplano na t-ésima iteração, pois assim.

Certamente, tem-se as relações:

$$\begin{gathered} \gamma^+ > \gamma_f^t \quad \text{e} \quad \gamma^- \geq \gamma_f^t \\ \text{ou} \\ \gamma^+ \geq \gamma_f^t \quad \text{e} \quad \gamma^- > \gamma_f^t \end{gathered} \tag{6.16}$$

garantindo que $\gamma_f^{t+1} > \gamma_f^t$.

Neste caso, sempre haverá a garantia de solução do novo problema, já que a nova margem fixa é inferior à margem ótima, ou seja:

$$\gamma_f^{t+1} = \frac{\gamma^+ + \gamma^-}{2} < \gamma^* \tag{6.17}$$

Pode-se confirmar isso devido ao fato de que as margens positiva e negativa não são iguais e, portanto, a margem total não é máxima, o que implica que:

$$(\gamma^+ + \gamma^-) < 2\gamma^* \tag{6.18}$$

Infelizmente a condição de desigualdade das margens não é suficiente para caracterizar a margem ótima, não devendo ser escolhida como critério de parada para o algoritmo. Caso ele seja utilizado, a solução encontrada não será o ótimo global, e sim um ótimo local.

Segunda regra: Caso as margens positiva e negativa obtidas pela solução do

FMP sejam iguais, pode ser que a solução seja um ótimo local. Por isso, é importante garantir um acréscimo na margem fixa, da forma:

$$\gamma_f^{t+1} = \gamma_f^t + \max\left\{\delta, \frac{(\gamma^+ + \gamma^-)}{2} - \gamma_f^t\right\} \tag{6.19}$$

sendo δ uma constante compreendida entre 0 e 1.

Porém, para essa atualização, não se tem garantia da solução do novo problema, pois o novo valor da margem fixa poderá ser maior ou igual ao valor da margem ótima.

Para a solução desse empecilho é suficiente a limitação de um número máximo de iterações no número de épocas do algoritmo de treinamento. Caso não encontre uma nova solução do FMP, a solução será o valor da margem fixa obtida anteriormente.

(Leite et al., 2008) descreve o algoritmo do Algoritmo de Margem Incremental (IMA) da seguinte maneira:

Algoritmo 6: Algoritmo de Margem Incremental

Entrada: $z_m, \eta, \delta, T_{MAX}$
Saída: o último w viável
início
  $w \leftarrow 0$;
  $\gamma_f \leftarrow 0$;
  repita
    $w \leftarrow PerceptronPrimal(z_m, w, \gamma_f, \eta, T_{MAX})$;
    $\gamma_f \leftarrow max\left(\frac{\gamma^+(w) + \gamma^-(w)}{2}, (1+\delta)\gamma_f\right)$;
  até *que a convergência do FMP no máximo de* T_{MAX} *iterações não seja alcançada*;
fim

6.4 Função *Kernel*

A representação dual do perceptron, que também é chamada de representação dependente dos dados, pode ser interpretada como uma função *kernel*. A medida de similaridade entre os dados é computada pelo produto interno dos vetores do espaço de entrada.

Assim, pode-se garantir a separabilidade linear dos dados quando a medida de similaridade computada pelos respectivos produtos internos é suficiente para permitir a discriminação dos dados em duas classes.

Caso o problema não seja linearmente separável no espaço de entrada, torna-se necessário mapear os dados. Entretanto, na maior parte dos casos não é necessário conhecer o tipo de mapeamento. Para tanto, utiliza-se uma função *kernel* tal que os valores desta função correspondem aos valores do produto interno dos vetores mapeados no espaço de mais alta dimensão.

As funções *kernel* mais utilizadas nos projetos de classificadores são as seguintes (Defilippo, 2004):

- **Produto Interno**: $K(x, x_i) = \langle x, x_i \rangle$
- **Produto Interno Normalizado**: $K(x, x_i) = \dfrac{\langle x, x_i \rangle}{\sqrt{\langle x, x \rangle \langle x_i, x_i \rangle}}$
- **Gaussiano**: $K(x, x_i) = e^{\frac{||x - x_i||^2}{\sigma^2}}$
- **Polinomio de Grau**: $K(x, x_i) = (\langle x, x_i \rangle + \Theta)^d$
- **Sigmoidal**: $K(x, x_i) = tanh(kx^T x_i b)$
- **Multiquadrático**: $K(x, x_i) = \dfrac{1}{\sqrt{||x - x_i||^2 - c^2}}$
- **Multidimensional Spline**: $K(x, x_i) = \prod_k K_k(x^k, x_i^k)$
- **Multiplicativo Anova**: $K(x, x_i) = \left(\sum_k e^{-\gamma (x^k - x_i^k)^2} \right)^d$

6.5 Função Sinal

Foi elaborada uma função sinal para ser aplicada à partir da segunda iteração do FMP, em que essa função rotulasse todos os dados que não eram originalmente rotulados em relação à função que está sendo estimada.

Algoritmo 7: Classificar um dado em relação à função sinal

Entrada: alpha, b, posiçao atual, pontos
Saída: rótulo
início
  Calcular o valor da função estimada de acordo com *alpha* e *b* no ponto *pontos*[*posição atual*] e guarda em *valor*;
  se *valor* < 0 **então**
    Retornar *rótulo* $= -1$;
  senão
    Retornar *rótulo* $= 1$;
  fim se
fim

7 Impressão e Cálculo do Comprimento da Curva

7.1 Impressão dos Dados de Obstáculos

Para a impressão da trajetória e dos dados de obstáculos, utilizou-se a ferramenta *gnuplot*. Os dados foram divididos em 3 grupos, os dados não rotulados, os dados com $y_i = 1$ e os dados com $y_i = -1$. Assim, cada legenda dos pontos terá uma cor diferente, facilitando a diferenciação entre cada tipo de dados.

7.2 Impressão da Trajetória

Para cada algoritmo implementado, foi elaborado uma maneira de se imprimir a trajetória:

7.2.1 Algoritmo Primal Linear

No Algoritmo Primal Linear, temos que o vetor $\overrightarrow{w}$ é determinado pelo hiperplano. Assim, juntamente com o bias, temos que a função de classificação que determina a trajetória é descrita como:

$$f(x) = -\frac{w[0]}{w[1]}x - \frac{b}{w[1]} \tag{7.1}$$

7.2.2 Algoritmo Dual Linear

Para o Algoritmo Dual Linear, tem-se que o algoritmo utilizado retorna o vetor de α_i. Para calcular o vetor $\overrightarrow{w}$, temos que:

$$\overrightarrow{w} = \sum_{i=1}^{p} \alpha_i \overrightarrow{x_i} y_i \tag{7.2}$$

Então, pode-se escrever a função $f(x)$ de acordo com o vetor w calculado em 7.2.

$$f(x) = -\frac{w[0]}{w[1]}x - \frac{b}{w[1]} \tag{7.3}$$

7.2.3 Algoritmo Dual Não Linear

A função trajetória $f(x, y)$ é definida para a função *kernel* polinomial da seguinte maneira:

$$f(x, y) = \sum_{i=1}^{p} \alpha_i y_i (\langle x, x_i \rangle + \theta)^d + b \tag{7.4}$$

Nos algoritmos implementados, foram utilizados os *kernels* polinomiais de grau 2 e 3, no qual será exposta a resolução para o *kernel* polinomial de grau 3.

Na equação 7.4, substituindo o valor de $d = 3$, tem-se que:

$$\begin{aligned}
f(x,y) &= \sum_{i=1}^{p} \alpha_i y_i (\langle x, x_i \rangle + \theta)^3 + b \\
&= \sum_{i=1}^{p} \alpha_i y_i ((x_{ix}x) + (x_{iy}y) + \theta)^3 + b \\
&= \sum_{i=1}^{p} \alpha_i y_i (x^2 x_{ix}^2 + 2x x_{ix} y x_{iy} + 2x x_{ix}\theta + y^2 x_{iy}^2 + 2y x_{iy}\theta + \theta^2) \\
&\quad ((x_{ix}x) + (x_{iy}y) + \theta) + b \\
&= \sum_{i=1}^{p} \alpha_i y_i (x^3 x_{ix}^3 + 3x^2 x_{ix}^2 y x_{iy} + 3x^2 x_{ix}^2 \theta + 3x x_{ix} y^2 x_{iy}^2 + 6x x_{ix} y x_{iy}\theta \\
&\quad + 3x x_{ix}\theta^2 + y^3 x_{iy}^3 + 3y^2 x_{iy}^2 \theta + 3y x_{iy}\theta^2 + \theta^3) + b
\end{aligned} \tag{7.5}$$

Com 7.6, pode-se definir o vetor w que compõe $f(x, y)$ para facilitar os cálculos.

$$
\begin{cases}
w[0] = \sum_{i=1}^{p} \alpha_i y_i x^3 x_{ix}^3 \\
w[1] = \sum_{i=1}^{p} 3\alpha_i y_i x^2 x_{ix}^2 y x_{iy} \\
w[2] = \sum_{i=1}^{p} 3\alpha_i y_i x^2 x_{ix}^2 \theta \\
w[3] = \sum_{i=1}^{p} 3\alpha_i y_i x x_{ix} y^2 x_{iy}^2 \\
w[4] = \sum_{i=1}^{p} 6\alpha_i y_i x x_{ix} y x_{iy} \theta \\
w[5] = \sum_{i=1}^{p} 3\alpha_i y_i x x_{ix} \theta^2 w[6] = \sum_{i=1}^{p} \alpha_i y_i y^3 x_{iy}^3 w[7] = \sum_{i=1}^{p} 3\alpha_i y_i y^2 x_{iy}^2 \theta \\
w[8] = \sum_{i=1}^{p} 3\alpha_i y_i y x_{iy} \theta^2 \\
w[9] = \sum_{i=1}^{p} \alpha_i y_i \theta^3
\end{cases}
$$

Devido à função gerada pelo algoritmo ser uma função implícita, foi necessário utilizar algumas funções do *gnuplot* para que essa curva encontrada estivesse no mesmo plano que os pontos impressos na subseção 7.1.

7.3 Comprimento da curva

Inicialmente, pretendia-se calcular o comprimento da curva (trajetória) através da seguinte fórmula:

$$c = \int_{\alpha}^{\beta} \sqrt{1 + (f'(x))^2} dx \tag{7.6}$$

Para isso, era necessário conseguir a função $f(x)$ que determinasse a curva à partir da função implícita $f(x, y)$. Com a ferrementa *gnuplot*, utilizou-se algumas funções que aproximavam alguns pontos definidos pela curva $f(x, y) = 0$ pelo método de Mínimos Quadrados. Com a função $f(x)$, utiliza-se o método de Quadratura Gaussiana para encontrar o resultado da integral, que é o comprimento da curva que se deseja.

Porém, nem sempre a quantidade de pontos definida pela curva $f(x, y) = 0$ é suficiente para que o método de Mínimos Quadrados pudesse calcular a função corretamente. Então optou-se por, em vez de obter a equação $f(x)$, plotar e ligar os pontos obtidos pelo *gnuplot* ao plotar o gráfico de $f(x, y) = 0$ e calcular o comprimento obtido através desses pontos pela fórmula da distância entre dois pontos.

$$dist = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \tag{7.7}$$

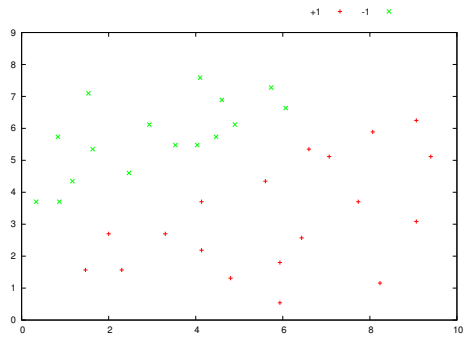
8 Resultados

Nesse capítulo serão apresentadas as instâncias elaboradas para o trabalho e suas respectivas soluções encontradas pelos algoritmos desenvolvidos. Foram elaboradas 7 conjuntos de instâncias, sendo três instâncias linearmente separáveis e quatro não linearmente separáveis. Dessas quatro, duas instâncias separáveis por um polinômio quadrático e as outras duas por um polinômio cúbico.

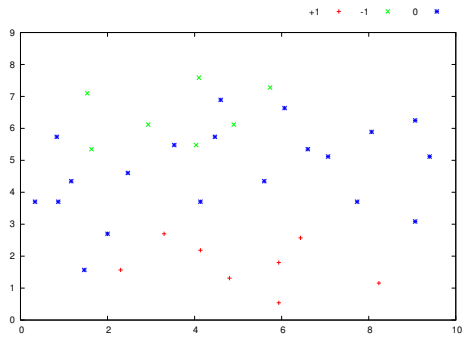
Os algoritmos foram executados considerando como critério de parada uma quantidade de iterações igual a 1.000.000 de iterações, com exceção da instância número 7, em que, para o algoritmo polinomial de grau 3, utilizou-se um número de iterações igual a 1.100.000, devido à maior complexidade da instância. Nesse trabalho não utilizou-se a restrição de margem, devido ao algoritmo elaborado ter como objetivo buscar uma margem máxima para as instâncias.

8.1 Instância Número 1

A instância número 1 foi retirada da figura 2 de (Bennet et al., 1999), disposta de tal forma que os dados estejam separados linearmente. Essa instância é dividida em três grupos de dados: um com 8 dados com rótulo +1, um conjunto de 7 dados com rótulo -1 e 20 dados sem rótulos. A seguir, tem-se duas imagens, a primeira (8.1(a)) onde estão plotados os pontos com os rótulos ideais, e a segunda (8.1(b)), onde são mostrados todos os três grupos que compoêm a instância.

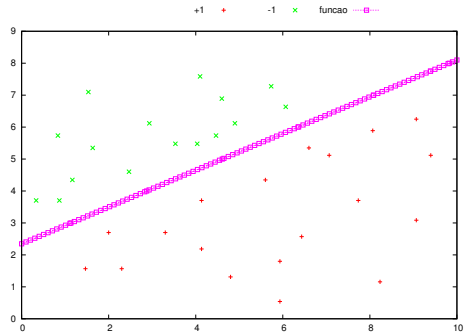


(a) Instância com todos os dados rotulados

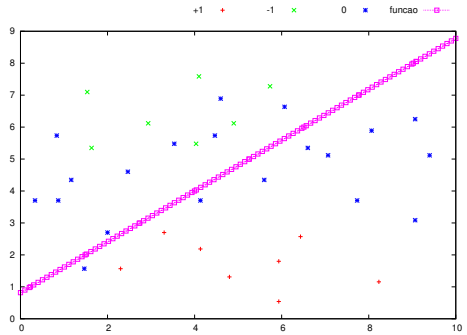


(b) Instância com alguns dados rotulados

Figura 8.1: Instância Número 1

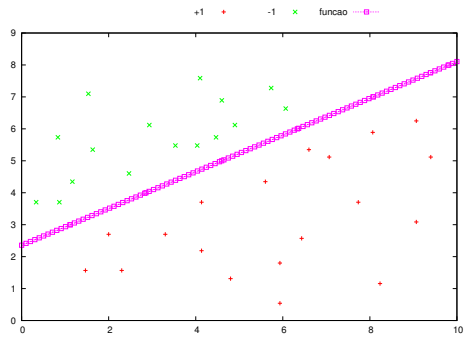


(a) Solução com todos os dados rotulados

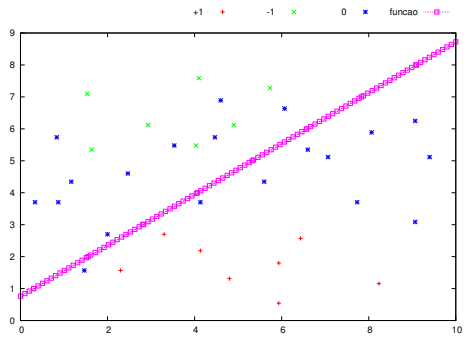


(b) Solução com alguns dados rotulados

Figura 8.2: Soluções do Algoritmo Primal com *Kernel* Produto Interno

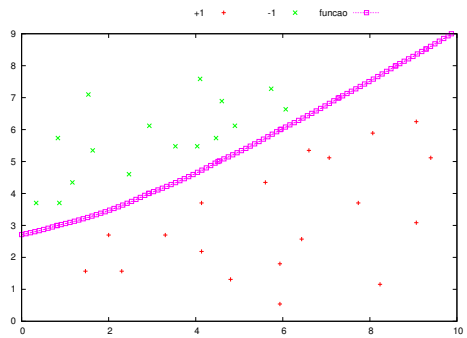


(a) Solução com todos os dados rotulados

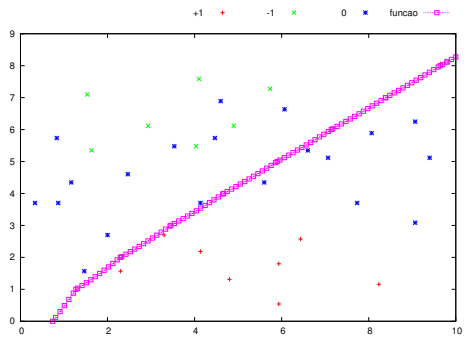


(b) Solução com alguns dados rotulados

Figura 8.3: Soluções do Algoritmo Dual com *Kernel* Produto Interno

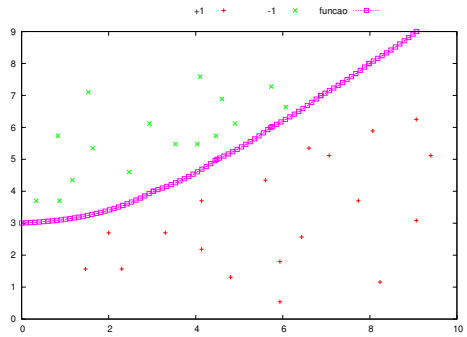


(a) Solução com todos os dados rotulados

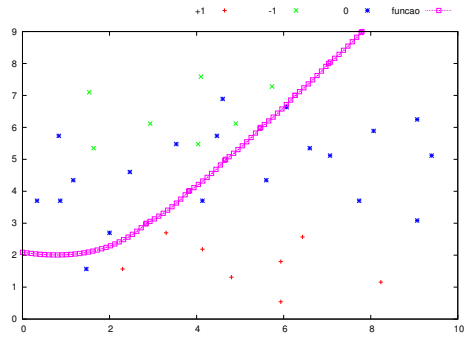


(b) Solução com alguns dados rotulados

Figura 8.4: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 2



(a) Solução com todos os dados rotulados



(b) Solução com alguns dados rotulados

Figura 8.5: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 3

Com esses gráficos gerados, tem-se a seguinte tabela com a margem alcançada com cada algoritmo para a instância 8.1.

	Todos os dados rotulados	Alguns dados rotulados
Primal Produto Interno	6.896553e-01	2.744798e-01
Dual Produto Interno	6.898966e-01	2.747903e-01
Dual Polinomio de grau 2	4.006460e+00	7.765757e-01
Dual Polinomio de grau 3	**1.880070e+01**	4.945839e+00

Tabela 8.1: Comparativo das margens das curvas para a instância Número 1

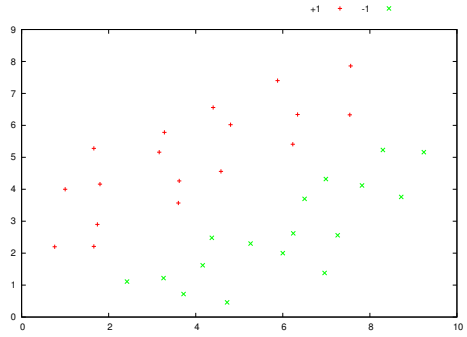
E, a seguir, tem-se uma tabela comparando os comprimentos das curvas alcançadas por cada algoritmo para a instância 8.1.

	Todos os dados rotulados	Alguns dados rotulados
Primal Produto Interno	1.153698e+01	1.278042e+01
Dual Produto Interno	1.153522e+01	1.278275e+01
Dual Polinomio de grau 2	1.178019e+01	1.250507e+01
Dual Polinomio de grau 3	1.111072e+01	**1.096917e+01**

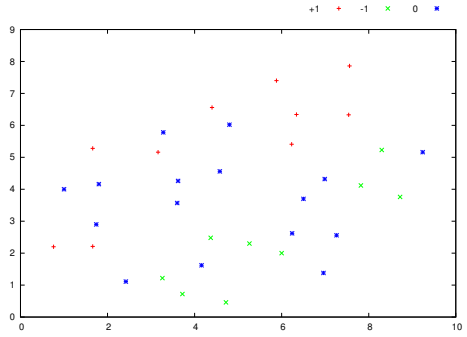
Tabela 8.2: Comparativo dos comprimentos das curvas para a instância Número 1

8.2 Instância Número 2

A instância número 2 foi criada com o auxílio do programa *Geogebra*. Nessa instância, tem-se: 9 pontos com rótulo -1, 10 dados rotulados com valor +1 e um conjunto de 16 pontos sem rótulos. Os dados estão espaçados de maneira que a função de classificação gerada pelo algoritmo se aproxime de uma reta.

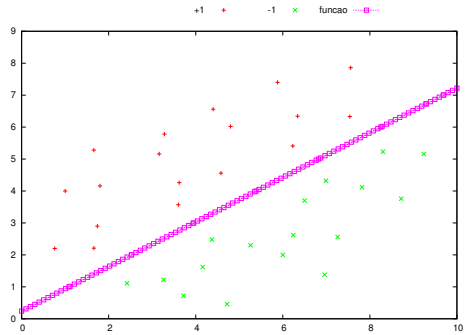


(a) Instância com todos os dados rotulados

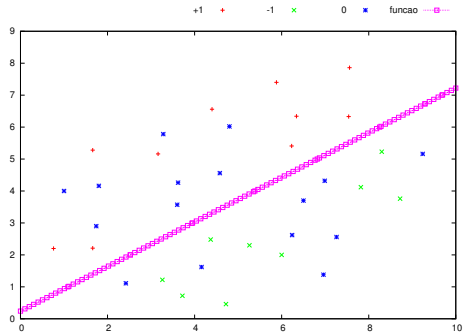


(b) Instância com alguns dados rotulados

Figura 8.6: Instância Número 2

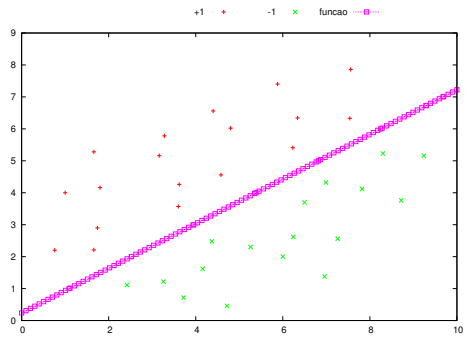


(a) Solução com todos os dados rotulados

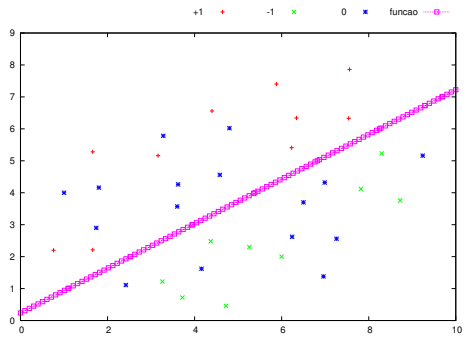


(b) Solução com alguns dados rotulados

Figura 8.7: Soluções do Algoritmo Primal com *Kernel* Produto Interno

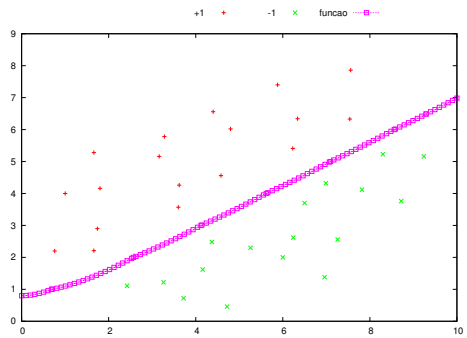


(a) Solução com todos os dados rotulados

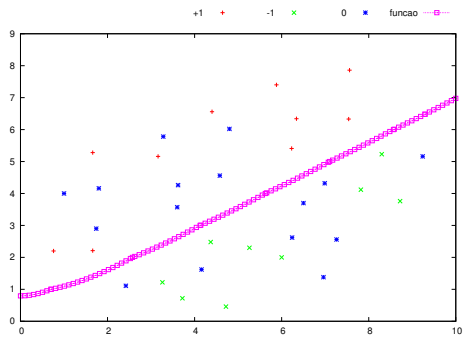


(b) Solução com alguns dados rotulados

Figura 8.8: Soluções do Algoritmo Dual com *Kernel* Produto Interno

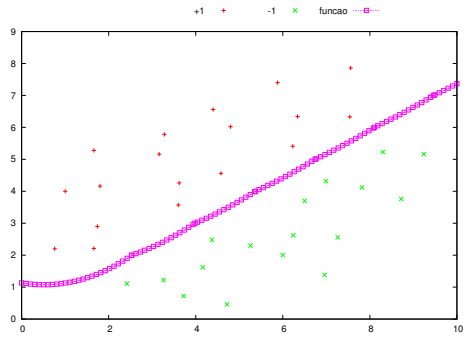


(a) Solução com todos os dados rotulados

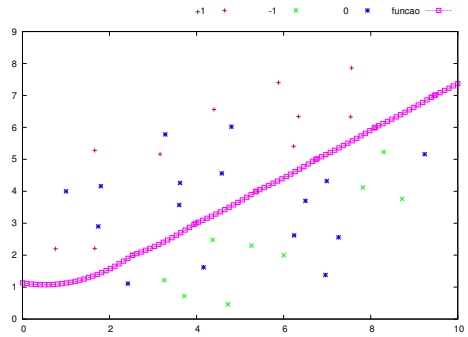


(b) Solução com alguns dados rotulados

Figura 8.9: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 2



(a) Solução com todos os dados rotulados



(b) Solução com alguns dados rotulados

Figura 8.10: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 3

Com esses gráficos gerados, tem-se a seguinte tabela com a margem alcançada com cada algoritmo para a instância 8.6.

	Todos os dados rotulados	Alguns dados rotulados
Primal Produto Interno	6.667492e-01	6.666106e-01
Dual Produto Interno	6.665115e-01	6.658185e-01
Dual Polinomio de grau 2	2.667027e+00	2.667153e+00
Dual Polinomio de grau 3	9.209390e+00	**9.209589e+00**

Tabela 8.3: Comparativo das margens das curvas para a instância Número 2

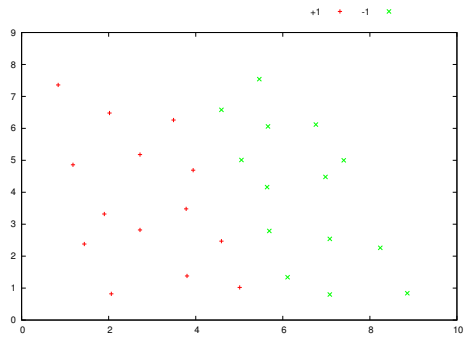
E, a seguir, tem-se uma tabela comparando os comprimentos das curvas alcançadas por cada algoritmo para a instância 8.6.

	Todos os dados rotulados	Alguns dados rotulados
Primal Produto Interno	1.219514e+01	1.219621e+01
Dual Produto Interno	1.219819e+01	1.219749e+01
Dual Polinomio de grau 2	1.181839e+01	**1.181695e+01**
Dual Polinomio de grau 3	1.200539e+01	1.200585e+01

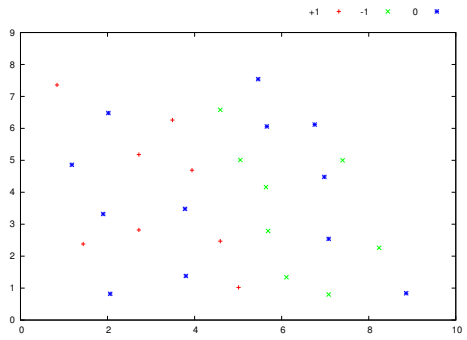
Tabela 8.4: Comparativo dos comprimentos das curvas para a instância Número 2

8.3 Instância Número 3

A instância número 3, também gerada com o auxílio do programa *Geogebra* possui 12 pontos não rotulados, 8 com rótulo -1 e 8 dados com rótulo de valor +1. Nessa instância, os dados foram espaçados tal que fossem linearmente separáveis.

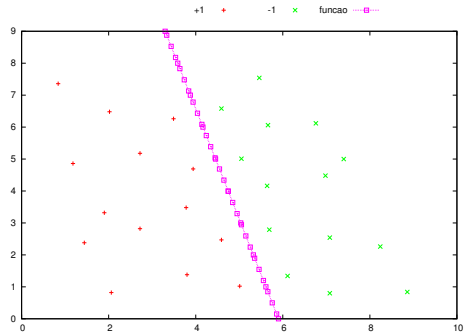


(a) Instância com todos os dados rotulados

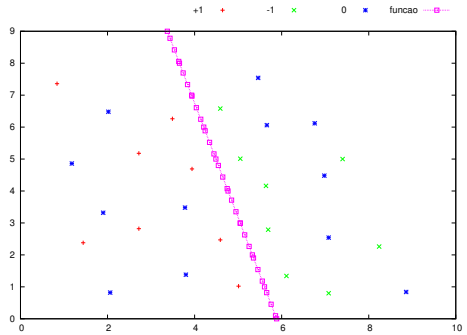


(b) Instância com alguns dados rotulados

Figura 8.11: Instância Número 3

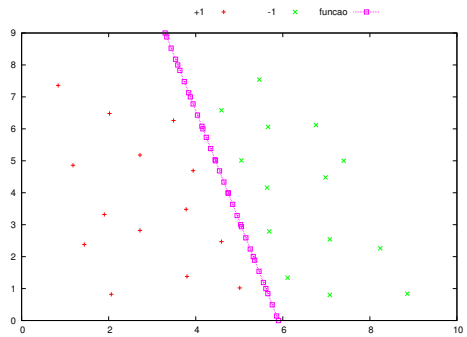


(a) Solução com todos os dados rotulados

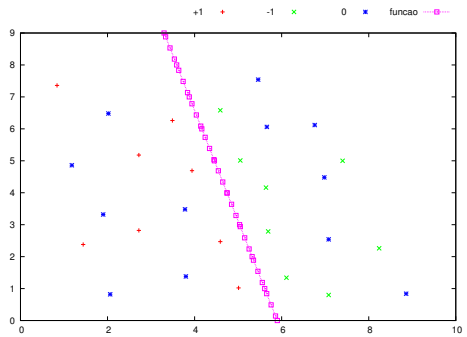


(b) Solução com alguns dados rotulados

Figura 8.12: Soluções do Algoritmo Primal com *Kernel* Produto Interno

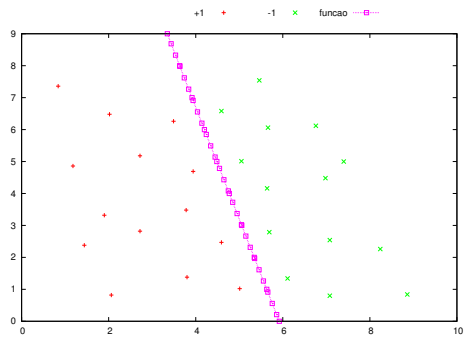


(a) Solução com todos os dados rotulados

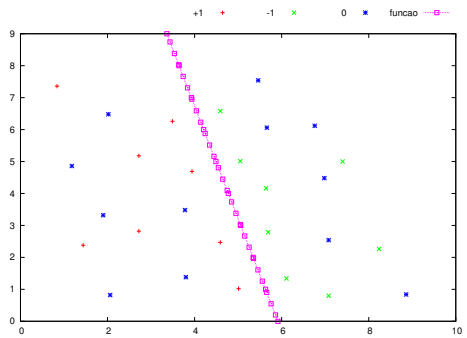


(b) Solução com alguns dados rotulados

Figura 8.13: Soluções do Algoritmo Dual com *Kernel* Produto Interno

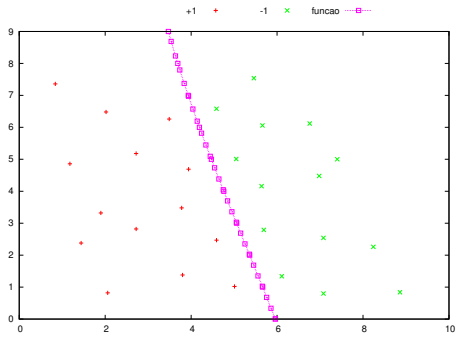


(a) Solução com todos os dados rotulados

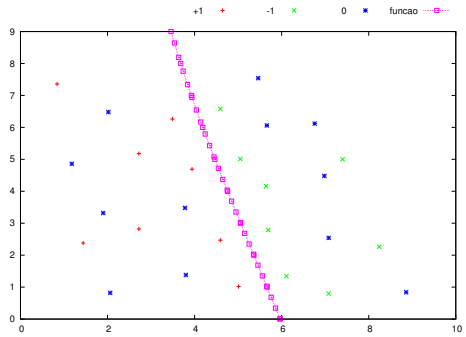


(b) Solução com alguns dados rotulados

Figura 8.14: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 2



(a) Solução com todos os dados rotulados



(b) Solução com alguns dados rotulados

Figura 8.15: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 3

Com esses gráficos gerados, tem-se a seguinte tabela com a margem alcançada com cada algoritmo para a instância 8.11.

	Todos os dados rotulados	Alguns dados rotulados
Primal Produto Interno	5.725083e-01	5.725041e-01
Dual Produto Interno	5.721985e-01	5.717715e-01
Dual Polinomio de grau 2	6.498055e+00	6.521777e+00
Dual Polinomio de grau 3	5.570767e+01	**5.625272e+01**

Tabela 8.5: Comparativo das margens das curvas para a instância Número 3

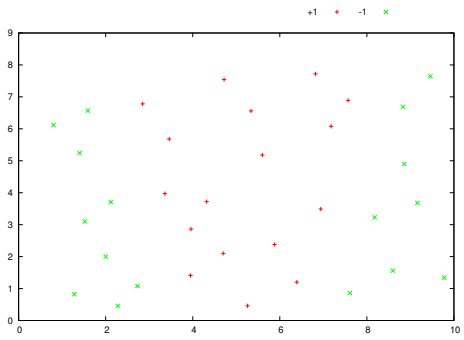
E, a seguir, tem-se uma tabela comparando os comprimentos das curvas alcançadas por cada algoritmo para a instância 8.11.

	Todos os dados rotulados	Alguns dados rotulados
Primal Produto Interno	9.369242e+00	9.343957e+00
Dual Produto Interno	9.368700e+00	9.368117e+00
Dual Polinomio de grau 2	9.360571e+00	9.355075e+00
Dual Polinomio de grau 3	**9.342180e+00**	9.345474e+00

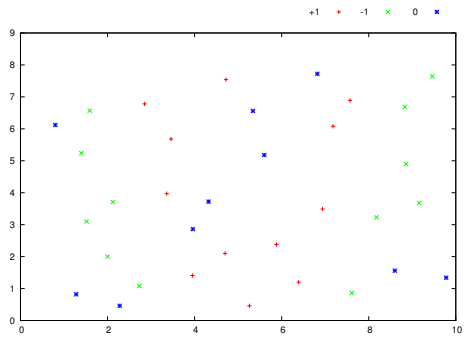
Tabela 8.6: Comparativo dos comprimentos das curvas para a instância Número 3

8.4 Instância Número 4

A instância 4, diferentemente das instâncias anteriores, é separável por um polinômio quadrático, por essa razão, executou-se para essa instância somente os algoritmos elaborados com as funções de *kernel* polinomial de grau 2 e grau 3. Essa instância possui 12 pontos rotulados com valor +1, 12 pontos com valor -1 e 10 sem rótulos.

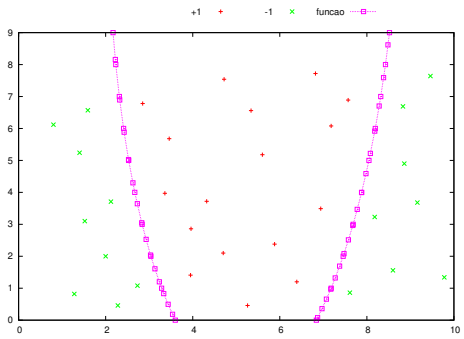


(a) Instância com todos os dados rotulados

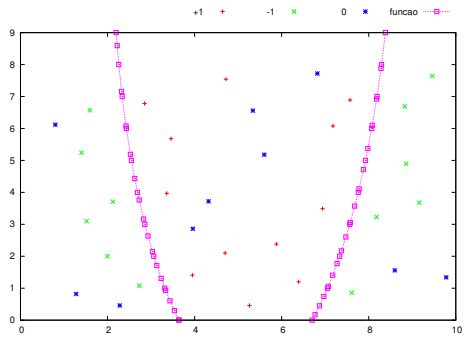


(b) Instância com alguns dados rotulados

Figura 8.16: Instância Número 4

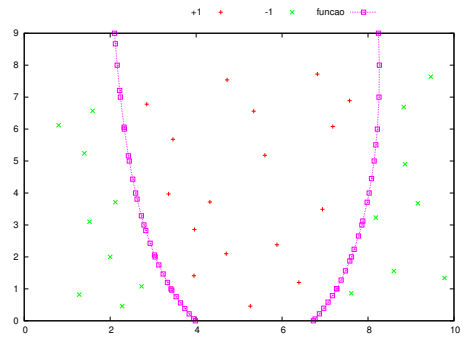


(a) Solução com todos os dados rotulados

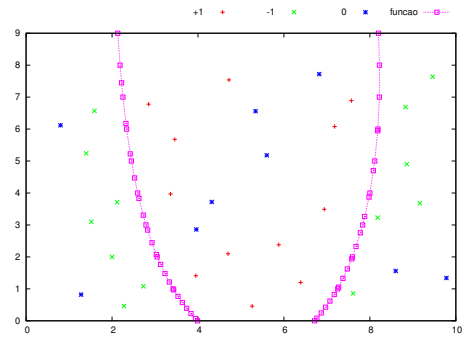


(b) Solução com alguns dados rotulados

Figura 8.17: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 2



(a) Solução com todos os dados rotulados



(b) Solução com alguns dados rotulados

Figura 8.18: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 3

Com esses gráficos gerados, tem-se a seguinte tabela com a margem alcançada com cada algoritmo para a instância 8.16.

	Todos os dados rotulados	Alguns dados rotulados
Dual Polinomio de grau 2	3.231758e-01	3.229148e-01
Dual Polinomio de grau 3	**3.246651e+00**	3.246656e+00

Tabela 8.7: Comparativo das margens das curvas para a instância Número 4

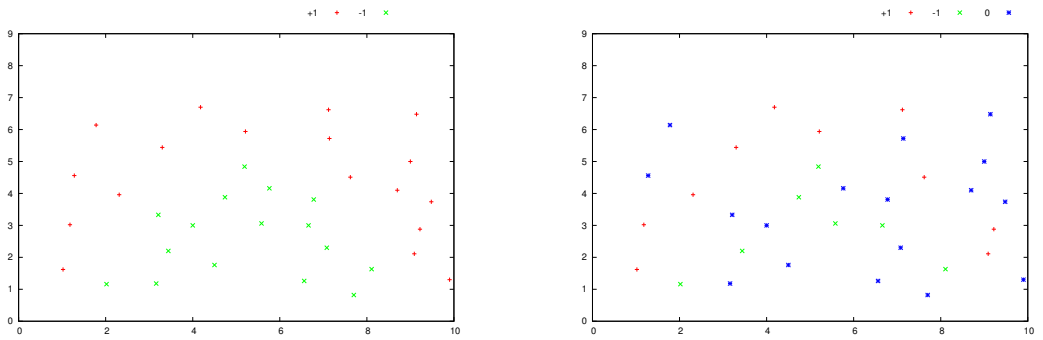
E, a seguir, tem-se uma tabela comparando os comprimentos das curvas alcançadas por cada algoritmo para a instância 8.16.

	Todos os dados rotulados	Alguns dados rotulados
Dual Polinomio de grau 2	**1.832000e+01**	1.832848e+01
Dual Polinomio de grau 3	1.854611e+01	1.854056e+01

Tabela 8.8: Comparativo dos comprimentos das curvas para a instância Número 4

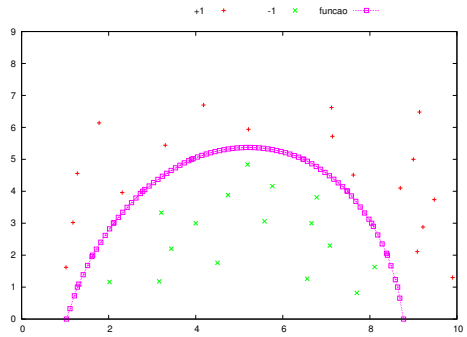
8.5 Instância Número 5

Para a instância número 5, gerada com o auxílio da ferramenta *Geogebra*, tem-se a seguinte classificação dos dados: 10 com rótulo +1, 7 com rótulo -1 e 17 dados sem rotulação, dispostos de tal maneira que são separados por uma função quadrática.

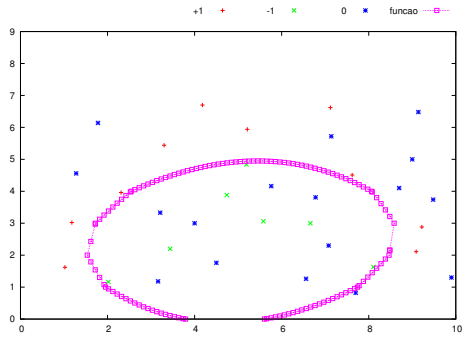


(a) Instância com todos os dados rotulados (b) Instância com alguns dados rotulados

Figura 8.19: Instância Número 5

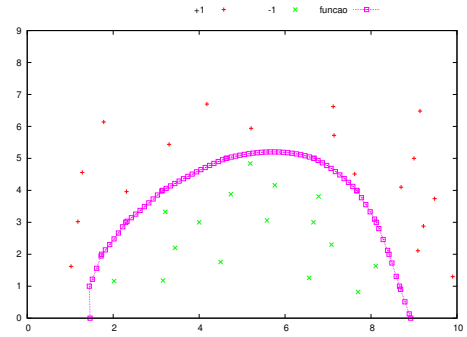


(a) Solução com todos os dados rotulados

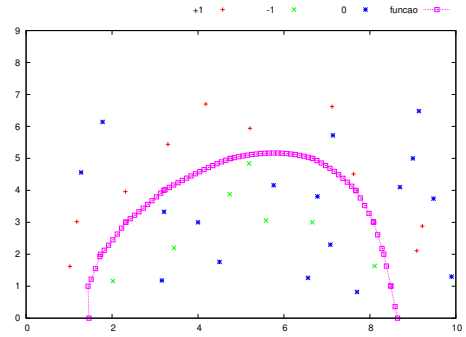


(b) Solução com alguns dados rotulados

Figura 8.20: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 2



(a) Solução com todos os dados rotulados



(b) Solução com alguns dados rotulados

Figura 8.21: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 3

Com esses gráficos gerados, tem-se a seguinte tabela com a margem alcançada com cada algoritmo para a instância 8.19.

	Todos os dados rotulados	Alguns dados rotulados
Dual Polinomio de grau 2	3.269074e-01	1.152131e-01
Dual Polinomio de grau 3	2.441628e+00	**2.449847e+00**

Tabela 8.9: Comparativo das margens das curvas para a instância Número 5

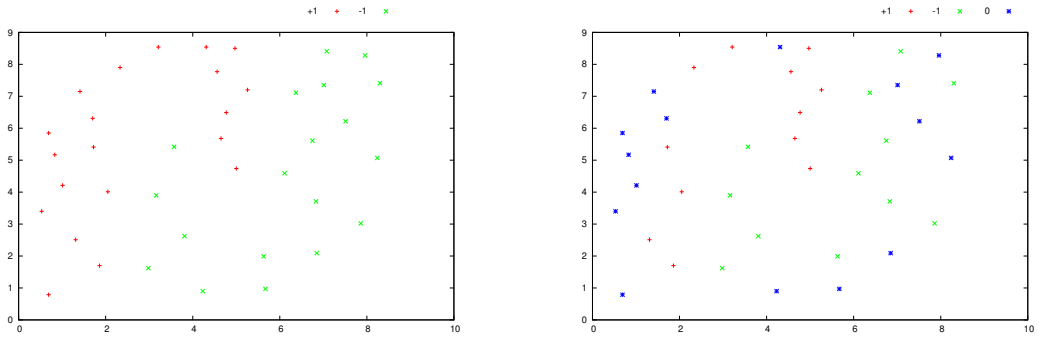
E, a seguir, tem-se uma tabela comparando os comprimentos das curvas alcançadas por cada algoritmo para a instância 8.19.

	Todos os dados rotulados	Alguns dados rotulados
Dual Polinomio de grau 2	1.429418e+01	1.717878e+01
Dual Polinomio de grau 3	1.401420e+01	**1.389633e+01**

Tabela 8.10: Comparativo dos comprimentos das curvas para a instância Número 5

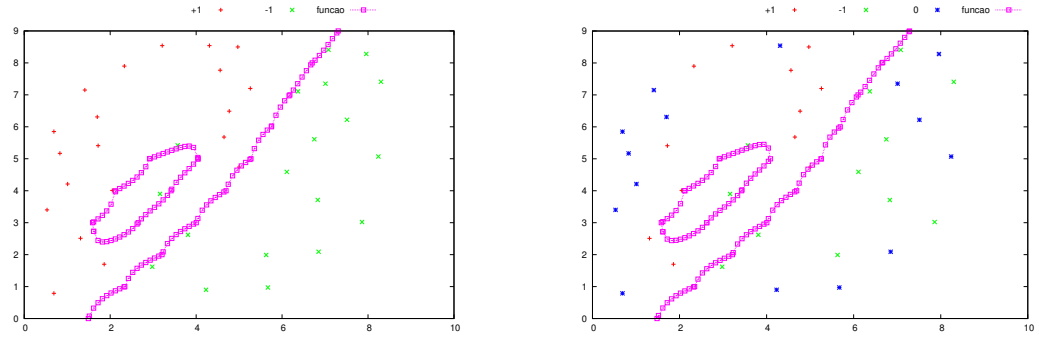
8.6 Instância Número 6

A instância número 6, separada por uma função cúbica, é executada somente pelo algoritmo de *kernel* polinomial de grau 3, e possui a seguinte separação dos dados: 12 dados com rótulo +1, 12 com rótulo com valor -1 e 15 dados não rotulados.



(a) Instância com todos os dados rotulados (b) Instância com alguns dados rotulados

Figura 8.22: Instância Número 6



(a) Solução com todos os dados rotulados (b) Solução com alguns dados rotulados

Figura 8.23: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 3

Com esses gráficos gerados, tem-se a seguinte tabela com a margem alcançada com cada algoritmo para a instância 8.22.

	Todos os dados rotulados	Alguns dados rotulados
Dual Polinomio de grau 3	**9.848254e-02**	9.528678e-02

Tabela 8.11: Comparativo das margens das curvas para a instância Número 6

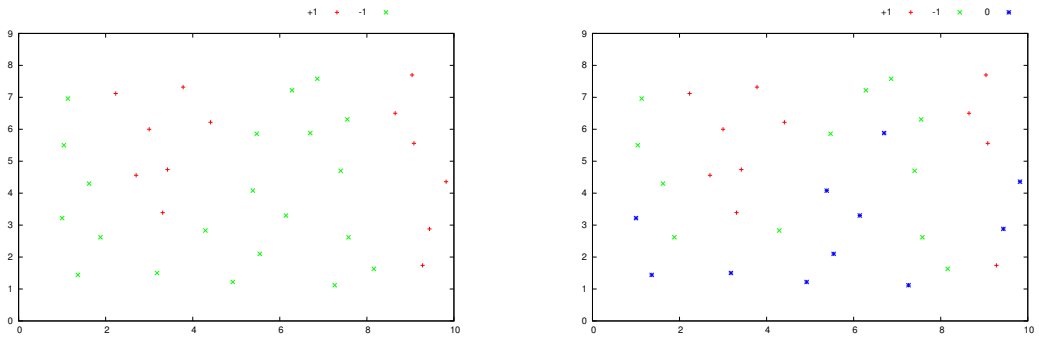
E, a seguir, tem-se uma tabela comparando os comprimentos das curvas alcançadas por cada algoritmo para a instância 8.22.

	Todos os dados rotulados	Alguns dados rotulados
Dual Polinomio de grau 3	**2.187168e+01**	2.195466e+01

Tabela 8.12: Comparativo dos comprimentos das curvas para a instância Número 6

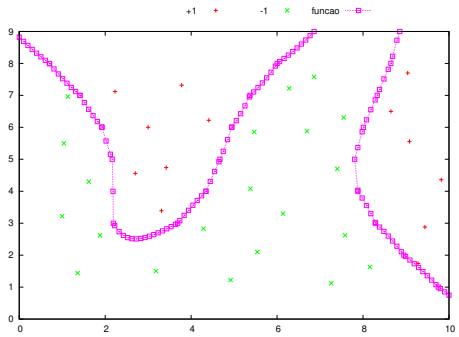
8.7 Instância Número 7

Para a sétima instância elaborada, tem-se 11 pontos não rotulados, 12 pontos rotulados com valor -1 e 11 pontos com valor +1. Essa instância foi executada somente com o algoritmo que implementa o *kernel* polinomial de grau 3 devido à sua elaboração ter feita de acordo com uma função cúbica.

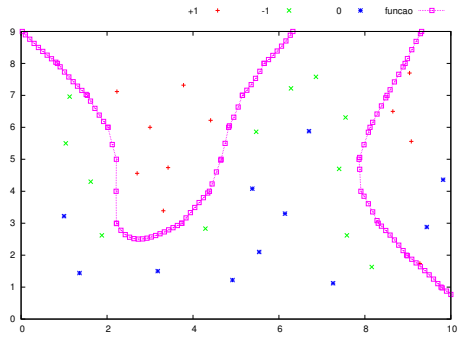


(a) Instância com todos os dados rotulados (b) Instância com alguns dados rotulados

Figura 8.24: Instância Número 7



(a) Solução com todos os dados rotulados



(b) Solução com alguns dados rotulados

Figura 8.25: Soluções do Algoritmo Dual com *Kernel* Polinomial de Grau 3

Com esses gráficos gerados, tem-se a seguinte tabela com a margem alcançada com cada algoritmo para a instância 8.24.

	Todos os dados rotulados	Alguns dados rotulados
Dual Polinomio de grau 3	1.682365e-01	**1.792460e-01**

Tabela 8.13: Comparativo das margens das curvas para a instância Número 7

E, a seguir, tem-se uma tabela comparando os comprimentos das curvas alcançadas por cada algoritmo para a instância 8.24.

	Todos os dados rotulados	Alguns dados rotulados
Dual Polinomio de grau 3	**2.427953e+01**	2.431455e+01

Tabela 8.14: Comparativo dos comprimentos das curvas para a instância Número 7

9 Conclusões e trabalhos futuros

9.1 Conclusões

O problema de planejamento de trajetórias em um espaço contendo obstáculos se resume a um problema de classificação desses obstáculos em dois conjuntos de dados separados por uma função de classificação, de um lado da função estarão os dados (já rotulados ou não) com rótulos +1, e do outro lado, os dados (já rotulados ou não) com valor de rótulo igual a -1.

Foram desenvolvidos quatro algoritmos nesse trabalho: o primeiro implementando o algoritmo primal do perceptron linear e o segundo implementando a versão dual do primeiro algoritmo, ambos foram utilizados para as instâncias linearmente separáveis; o terceiro algoritmo implementou a versão dual do FMP, porém, diferentemente da segunda implementação, utilizou-se como função *kernel* o *kernel* polinomial de grau 2; e o quarto algoritmo diferencia-se do terceiro pela utilização de *kernel* polinomial de grau 3 como função *kernel*.

Esses algoritmos retornaram resultados consideráveis, sendo capaz de rotular corretamente a maioria dos dados de exemplos de acordo com o resultado esperado. Os poucos dados os quais o algoritmo retornou um valor de rótulo diferente do esperado, foi porque o algoritmo, no primeiro laço por ele executado, rotulou o dado com o rótulo errado, e esse rótulo adquirido influenciou consideravelmente nos valores de γ calculados no decorrer do algoritmo, sem, no entanto, conseguir modificar seu valor.

Um dos problemas encontrados no trabalho foi que os algoritmos são muito suscetíveis aos dados não rotulados, principalmente se esses dados influenciam bastante na curva encontrada como solução do algoritmo, ou seja, quanto maior o valor de α do dado, maior a influência dele na solução do algoritmo.

9.2 Trabalhos Futuros

Considerando os resultados encontrados, alguns tópicos surgiram para serem seguidos em futuros trabalhos:

- Aumentar o poder de processamento do algoritmo para um maior número de dimensões.
- Utilizar outras funções *kernel*, como, por exemplo, o *kernel* gaussiano.
- Melhorar o poder de classificação dos dados não rotulados, realizando a troca o rótulo de dados que influenciam bastante na margem. Caso o dado já estivessem corretamente rotulado, a margem atual será pior que a anterior. Caso ele estivesse mal classificado, a margem resultante será melhor que a anterior.
- Estudar o comportamento do algoritmo no que diz respeito às transformações do espaço de entrada para o espaço de características, e consequentemente, do espaço de características para o espaço de entrada.

Referências Bibliográficas

Bennett, K.; Demiriz, A. Semi-supervised support vector machines. **Advances in Neural Information processing systems**, p. 368–374, 1999.

de Carvalho, B. O estado da arte em métodos para reconhecimento de padrões: Support vector machine. 2005.

Chapelle, O.; Sindhwani, V. ; Keerthi, S. Branch and bound for semi-supervised support vector machines. 2007.

De Bie, T.; Cristianini, N. Convex methods for transduction. **Advances in neural information processing systems**, v.16, 2003.

Defilippo, S. B. Máquinas de vetores suportes. **Monografia para obtenção do grau de Bacharel ao Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora**, 2004.

Neto, R. F. **Novos algoritmos para problemas de classificação e regressão baseados em estratégias de balanceamento e na solução de sistemas de inequações**, 2006.

Gammerman, A.; Vovk, V. ; Vapnik, V. **Learning by transduction**. In: Uncertainty in Artificial Intelligence, p. 148–155, 1998.

Grudic, G. **Introduction to classification**, 2005. Lecture's notes.

Handl, J.; Knowles, J. **On semi-supervised clustering via multiobjective optimization**. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, p. 1465–1472. ACM, 2006.

Haykin, S. **Neural networks and learning machines**, volume 3. Prentice Hall, 2009.

Karlen, M.; Weston, J.; Erkan, A. ; Collobert, R. Large scale manifold transduction. 2006.

Leite, S.; Neto, R. Incremental margin algorithm for large margin classifiers. **Neurocomputing**, v.71, n.7-9, p. 1550–1560, 2008.

Lorena, A. C.; de Carvalho, A. C. P. L. F. **Introdução às máquinas de vetores suporte (support vector machines)**. Technical report, Universidade de São Paulo, 2003.

Miura, J. **Support vector path planning**. In: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, p. 2894–2899. IEEE, 2006.

Ormonde, R. **Classificação automática de páginas web multi-label via mdl e support vector machines**. 2009. Dissertação de Mestrado - Universidade de Brasília.

Principe, J. C.; Euliano, N. R. ; Lefebvre, W. C. **Neural and Adaptive Systems: Fundamentals through Simulations**. Wiley, 1999.

Semolini, R. **Support vector machines, inferência transdutiva e o problema de classificação**. 2002. Dissertação de Mestrado - Universidade Estadual de Campinas.

Vapnik, V. **The nature of statistical learning theory**. Springer Verlag, 2000.

WikiBooks. **Support vector machines**. `http://en.wikibooks.org/wiki/Support_Vector_Machines#Transductive_support_vector_machines`, Mar. 2011.

Wu, Z.; hung Li, C. **Featuring selection using/for transductive support vector machine**, 2003. NIPS03 Workshop of Feature Extraction and Feature Selection Challenge.