Universidade Federal do Paraná

Maxim Dmitri Lobkov

Máquinas de Vetores Suporte no contexto de Aprendizagem Semi-supervisionada

Curitiba PR
2023

Maxim Dmitri Lobkov

Máquinas de Vetores Suporte no contexto de Aprendizagem Semi-supervisionada

Trabalho apresentado à banca examinadora da Universidade Federal do Paraná, como requisito parcial à obtenção do Título de Bacharel em Matemática Industrial.

Orientador: Prof. Dr. Lucas Garcia Pedroso.

Curitiba PR
2023

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS EXATAS
COORDENAÇÃO DO CURSO DE MATEMÁTICA INDUSTRIAL

ATA DE APRESENTAÇÃO DO PROJETO DE MATEMÁTICA INDUSTRIAL

Título do Trabalho
Máquinas de vetores suporte no contexto de aprendizagem semi-supervisionada

Autor
Registro Acadêmico: GRR 20185692 Nome: Maxim Dmitri Lobkov

Apresentação
Data: 24/02/2023 Hora: 09:30 Sala: sala 300

Banca Examinadora

	Nome	Assinatura
Orientador:	Lucas Garcia Pedroso	Lucas Garcia Pedroso
Membro 1:	Ademir Alves Ribeiro	
Membro 2:	Luiz Carlos Matioli	

Avaliação

Item	Pontuação	Orientador	Membro 1	Membro 2	Média
Apresentação Oral*	de 0 a 20	20	20	20	20
Produto Escrito**	de 0 a 60	60	60	60	60
Desempenho na Arguição	de 0 a 20	20	20	20	20
Média Final		100	100	100	100

∗ Objetividade, clareza e criatividade; domínio do tema desenvolvido; evolução lógica dos argumentos; respeito ao tempo mínimo de 20 (vinte) minutos e máximo de 30 (trinta) minutos.
∗∗ Normas da UFPR para apresentação escrita de monografias; redação; revisão bibliográfica; desenvolvimento do tema (coerência, objetivos, desenvolvimento e conclusão).

Correções a serem feitas: ☐ sem correções/ correções indicadas no verso desta Ata
☒ a banca solicita as correções indicadas nas cópias da monografia

Lançamento de nota: entregar esta Ata preenchida e o arquivo eletrônico do trabalho corrigido à Coordenação do Curso, dentro do prazo estabelecido pelo coordenador no semestre corrente.

"Onde o medo não estiver mais, nada haverá. Somente eu restarei."

Frank Herbert

Agradecimentos

A toda minha família, em especial ao meu pai, à minha mãe e à minha irmã, que me apoiaram e motivaram durante toda minha jornada.

A todos os meus colegas que me acompanharam e me ajudaram durante esses anos de graduação.

Ao meu orientador, Prof. Dr. Lucas Garcia Pedroso, por todos os ensinamentos transmitidos com paciência e clareza, por aceitar em embarcar neste ciclo comigo e pelo tempo dedicado a orientar este trabalho.

A Profa. Dra. Lucelina Batista dos Santos, orientadora de minha iniciação científica, pela dedicação e paciência quanto aos estudos realizados, e pelas contribuições à minha formação.

A todos os professores e professoras do Departamento de Matemática da Universidade Federal do Paraná que fizeram parte da minha trajetória e colaboraram de alguma forma para meu desenvolvimento acadêmico.

Resumo

Neste trabalho é feita uma revisão dos conceitos principais sobre as Máquinas de Vetores Suporte que são utilizadas em problemas de classificação binária no contexto de Aprendizagem de Máquina Supervisionada. O objetivo é então estudar uma extensão do modelo, para o caso de aprendizagem semi-supervisionada, conhecido como as Máquinas de Vetores Suporte Transdutivas. O Método *Concave-Convex Procedure* é usado para resolver o problema de Otimização associado ao modelo. Foram realizados experimentos numéricos criados no *software* Octave para validar o desempenho do método.

Palavras-chave: *Máquinas de Vetores Suporte, SVM Transdutivo, Aprendizagem Semi-supervisionada.*

Abstract

This work reviews the main concepts about Support Vector Machines that are used in binary classification problems in the context of Supervised Machine Learning. The objective is then to study an extension of the model, for the semi-supervised learning case, known as the Transductive Support Vector Machines. The Concave-Convex Procedure Method is used to solve the Optimization problem associated with the model. Numerical experiments created in the Octave software were performed to validate the performance of the method.

Keywords: *Support Vector Machines, Transductive SVM, Semi-supervised Learning.*

Lista de Figuras

Lista de Tabelas

Sumário

Introdução

Aprendizagem de Máquina é uma área de pesquisa que ganhou um grande impulso devido aos avanços feitos no desenvolvimento da Inteligência Artificial. O objetivo deste trabalho é estudar um dos problemas principais desta área, a classificação binária, modelada pelas Máquinas de Vetores Suporte. Os seres humanos sempre realizaram a tarefa de classificar, como por exemplo: o que é certo ou errado, o que é comestível ou não. Por esta razão a classificação está presente em diversos modelos de Aprendizagem de Máquina, já que para certas tarefas é necessário que a máquina execute essa classificação de forma automática. Este tipo de tarefa pode ser encontrada na previsão de comportamento de clientes, por exemplo, onde o modelo de classificação vai determinar se o cliente comprará ou não um produto, com base em padrões que foram detectados anteriormente em compras passadas ou pesquisas na internet. A classificação binária também está presente em sistemas de recomendações das plataformas digitais como YouTube, onde a máquina analisa o conteúdo assistido pelo usuário, filtrando esse conteúdo a fim de recomendar ou não um vídeo.

As Máquinas de Vetores Suporte são um modelo de classificação binária que detecta padrões na distribuição de dados para realizar uma classificação adequada [1, 5]. No presente trabalho são apresentadas distintas formulações do modelo para diferentes tipos de distribuições de dados para poder aplicar esses conhecimentos a um problema mais complexo, quando algumas informações na base de dados estão ausentes. Diante disso, são estudadas as Máquinas de Vetores Suporte Transdutivas (TSVM), um modelo que é formulado para este caso de falta de informação, isto é, para um problema semi-supervisionado. O problema de Otimização referente ao modelo TSVM não será convexo [4]. A dificuldade em lidar com problemas não convexos é um fato estabelecido, por esta razão o problema é reformulado usando uma aproximação convexa da função objetivo, utilizando o Método ***Con**cave-**C**onvex **P**rocedure* (**CCCP**) [4]. Com o objetivo de validar o desempenho do método CCCP quando aplicado ao modelo TSVM, foram feitos experimentos numéricos para distintas bases de dados, sendo uma delas um conjunto de dados clássico utilizado para treinar modelos de Aprendizagem de Máquina, conhecido como MNIST [6].

No Capítulo 1 são apresentados os conceitos principais referentes a Aprendizagem de Máquina, junto aos diferentes tipos de aprendizado, como a aprendizagem supervisionada, semi supervisionada e não supervisionada. No Capítulo 2 são introduzidas as Máquinas de Vetores Suporte, iniciando com a contextualização do modelo e posteriormente é estudada a formulação do problema para as margens rígidas, margens flexíveis e o caso não linear. No Capítulo 3 são discutidas as Máquinas de Vetores Suporte Transdutivas, apresentando o modelo referente ao caso não linear, em seguida é introduzido o Método *Concave-Convex Procedure* e posteriormente é enunciado o algoritmo do método quando aplicado ao modelo TSVM. No Capítulo 4 são realizados experimentos numéricos para analisar o desempenho do algoritimo exposto no capítulo anterior. A implementação do

algoritmo e os códigos utilizados para gerar as bases de dados estão disponíveis no *GitHub* e no Apêndice. Por fim, são apresentadas as considerações finais do trabalho e propostas para estudos futuros, junto às referências utilizadas.

Capítulo 1

Aprendizagem de Máquina

Neste capítulo são introduzidos os principais conceitos sobre a Aprendizagem de Máquina que serão usados durante o trabalho. Como base foi utilizado o livro de Géron [6]. Será apresentado um panorama geral sobre a aprendizagem de máquina e os seus distintos tipos de aprendizagem, como supervisionada, semi-supervisionada e não supervisionada.

1.1 Introdução à Aprendizagem de Máquina

O conceito de Aprendizagem de Máquina (AM ou ML - *Machine Learning*) foi inicialmente definido em 1959 por Arthur Lee Samuel como "um campo de estudo que dá aos computadores a habilidade de aprender sem terem sido programados para tal". A obtenção dos modelos matemáticos nesta área estão fortemente ligados aos estudos feitos em Estatística e Otimização. Atualmente este ramo de pesquisa ganhou popularidade devido ao impulso da Inteligência Artificial (*Artificial Intelligence* - AI).

O objetivo da Aprendizagem de Máquina consiste em realizar um reconhecimento de padrões por parte dos computadores/máquinas, sem a necessidade de serem programados para realizar tarefas específicas, ou seja, sem intervenção humana. Este aprendizado é feito a partir de alguma base de dados, com o intuito dessa máquina realizar uma detecção automática de padrões, para que os modelos criados, ao serem expostos a uma nova base de dados, sejam capazes de se adaptar de forma independente. Vale ressaltar que esta ciência está cada vez mais presente no cotidiano, como por exemplo nos sistemas de recomendações nas plataformas digitais como Netflix e YouTube, no reconhecimento facial de pessoas, na detecção de fraudes, entre outros.

Na seção a seguir serão expostos tipos diferentes de aprendizado que uma máquina pode realizar, sendo que cada um desses casos envolve problemas que devem ser resolvidos usando abordagens específicas.

1.2 Tipos de Aprendizagem de Máquina

Na Aprendizagem de Máquina existem diversas formas de treinar a máquina para realizar alguma tarefa específica, mas neste trabalho serão apresentadas três maneiras: AM supervisionada, semi-supervisionada e não supervisionada.

No aprendizado supervisionado, os dados que são usados no treinamento da máquina possuem suas entradas e saídas conhecidas, isto é, estão rotulados. Desta forma,

o algoritmo do modelo que a máquina vai seguir extrai o conhecimento a partir deles e detecta padrões com o auxílio dos dados rotulados para prever os rótulos de dados adicionais. Por exemplo, modelos usados nos problemas de regressão e classificação são utilizados no contexto de aprendizagem supervisionada.

No aprendizado semi-supervisionado, os dados do conjunto de treinamento são divididos em dados rotulados e não rotulados, sendo que ambos são utilizados para treinar a máquina. A obtenção de dados não rotulados requer menos esforço, por esta razão, eles predominam o conjunto total de dados.

No aprendizado não supervisionado, a máquina será treinada a partir de um conjunto de dados que não está rotulado. Desta forma o algoritmo realiza uma busca de padrões com base nas semelhanças presentes no agrupamento de dados. Este tipo de aprendizagem está presente em problemas de clusterização, análise de sequenciamento e associação.

Será estudado um modelo de classificação de dados, as Máquinas de Vetores Suporte, que é um problema de aprendizagem supervisionada, porém o foco deste trabalho é estender as ideias desse modelo para o caso semi-supervisionado. Antes de estudar essa extensão, é preciso introduzir os conceitos fundamentais sobre as Máquinas de Vetores Suporte, desta forma, o próximo capítulo diz respeito a este problema de classificação.

Capítulo 2

Máquinas de Vetores Suporte

Neste capítulo serão apresentadas as Máquinas de Vetores Suporte, um método que realiza uma classificação binária. Serão tomados como base os estudos feitos por Vapnik [9] e Benatti [1]. Será apresentado o modelo e sua obtenção para os distintos casos relacionados à distribuição dos dados a serem classificados.

2.1 Introdução às Máquinas de Vetores Suporte

Em muitas aplicações torna-se necessário realizar uma separação automática em duas categorias a partir de um conjunto de dados. No âmbito de Aprendizagem de Máquina existem vários métodos que realizam esta tarefa de obter esse classificador, para poder fazer uma análise posterior. As Máquinas de Vetores Suporte (*Support Vector Machines* - SVM) foram inicialmente propostas em 1963 por Vapnik e Lerner [9]. A ideia dos autores foi usar um classificador linear para separar os dados em duas classes, usando como critério de otimização a maximização da distância entre os dados e o classificador.

Neste trabalho será considerado um problema em que o conjunto de dados é dividido em duas classes, denominado problema de classificação binária. Considere um vetor $x_i \in \mathbb{R}^n$ da forma $x_i = \left(x_1^{(i)}; \ldots; x_n^{(i)}\right)^T$ e um conjunto X dado por $X = \left\{(x_1; y_1), \ldots, (x_m; y_m)\right\}$, sendo $y_i \in \{-1, 1\}$ para $i = 1, \ldots, m$. Os elementos do conjunto X serão separados em duas classes definidas como

- X^+ para a classe positiva, representada por $y_i = 1$.
- X^- para a classe negativa, representada por $y_i = -1$.

Como dito anteriormente, o objetivo do método SVM consiste em obter um hiperplano que separe o conjunto em duas classes, classe positiva e negativa. Tal hiperplano é descrito por $w^T x + b = 0$. Uma vez determinado o hiperplano e definida função de decisão $m(x) = sinal\left(w^T x + b\right)$, é possível rotular uma nova amostra $\overline{x}$, usando a função de decisão, como pertencendo à classe positiva se $m(\overline{x}) = w^T \overline{x} + b > 0$ ou negativa se $m(\overline{x}) = w^T \overline{x} + b < 0$. A seguir serão vistos alguns casos particulares do modelo para distintas distribuições de dados.

2.1.1 SVM para Margens Rígidas

O primeiro caso a ser considerado corresponde a um conjunto de dados que está distribuído de forma linearmente separável.

Definição 2.1. *Um conjunto é dito linearmente separável quando é possível separar corretamente os pontos de classes diferentes por pelo menos um hiperplano* [1].

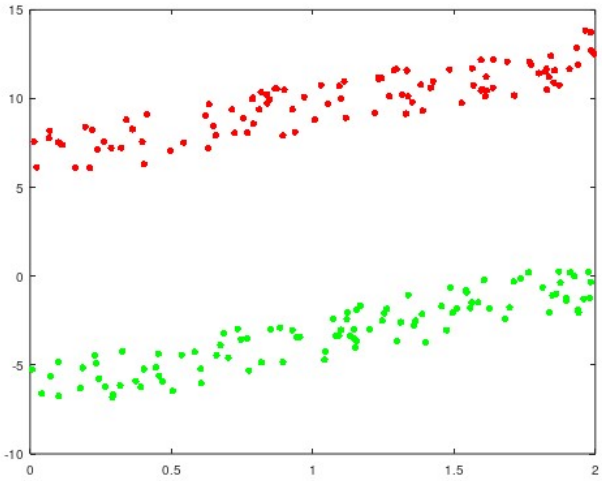

Figura 2.1: Conjunto de dados $X \subset \mathbb{R}^2$. Fonte: O Autor.

Suponha que na Figura 2.1 os dados em vermelho representam à classe positiva, isto é X^+, e os em verde a classe negativa X^-. Veja que há uma infinidade de hiperplanos separadores possíveis (Figura 2.2) para classificar os dados desse conjunto. Para obter o hiperplano ótimo, é preciso introduzir a noção de margem.

Definição 2.2. *Sejam $X \subset \mathbb{R}^{n+1}$ um conjunto de dados e β o hiperplano separador dado por $w^T x + b = 0$. A margem do classificador é a distância de β ao ponto $x \in X$ mais próximo* [1].

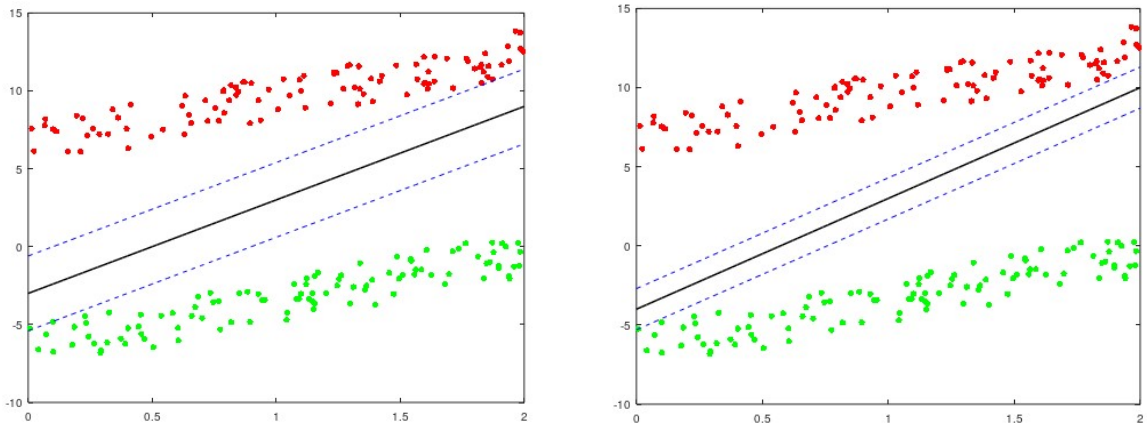

Figura 2.2: Hiperplanos com distintas margens. Fonte: O Autor.

Na Figura 2.2 são exibidos alguns exemplos de possíveis hiperplanos que separam o conjunto X de dados. O hiperplano ótimo que desejamos obter deve possuir uma margem máxima, já que implica em uma maior generalização do algoritmo SVM, classificando

melhor os elementos que não estavam previamente no conjunto de dados. Desta forma o hiperplano β exposto anteriormente satisfaz

$$\begin{cases} w^T x_i + b & \geq 1, \text{ para todo } i \text{ tal que } y_i = 1 \text{ (classe positiva);} \\ w^T x_i + b & = 1, \text{ para pelo menos um } x_i \in X; \\ w^T x_i + b & \leq -1, \text{ para todo } i \text{ tal que } y_i = -1 \text{ (classe negativa);} \\ w^T x_i + b & = -1, \text{ para pelo menos um } x_i \in X. \end{cases} \tag{2.1}$$

Na Figura 2.3 está representado o hiperplano ótimo com a maior margem que cumpre com as relações (2.1) enunciadas anteriormente.

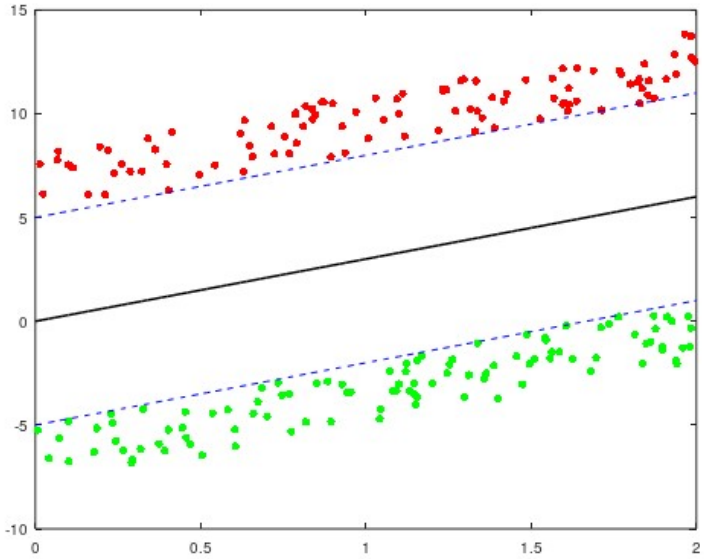

Figura 2.3: Hiperplano Ótimo para o Conjunto X linearmente separável. Fonte: O Autor.

Para obter o modelo, observe que usando a distância Euclidiana, a distância entre o hiperplano β e um ponto $x_i \in X$ é expressa por

$$dist(x_i, \beta) = \frac{|w^T x_i + b|}{\|w\|} = \frac{y_i(w^T x_i + b)}{\|w\|}.$$

Seguindo as relações estabelecidas em (2.1), observe que $y_i(w^T x_i + b) \geq 1$ para $i = 1, \ldots, m$, logo

$$dist(x_i, \beta) \geq \frac{1}{\|w\|}.$$

Os pontos $x_i \in X$ que estão na margem do classificador são denominados vetores suporte. Logo, se x_i é um vetor suporte então $dist(x_i, \beta) = \frac{1}{\|w\|}$. Obter um hiperplano com uma margem máxima, isto é com distância $dist(x_i, \beta)$ máxima, equivale a minimizar o inverso da distância, isto é determinar $\|w\|$ mínimo. Logo, o hiperplano desejado é aquele que resulta da resolução do seguinte problema de Otimização:

$$\begin{aligned} \underset{w,b}{\text{minimizar}} \quad & \|w\| \\ \text{sujeito a} \quad & y_i \left(w^T x_i + b \right) \geq 1, \quad i = 1, \ldots, m, \end{aligned}$$

substituindo a função objetivo anterior por outra equivalente e diferenciável, é obtido o problema quadrático convexo:

$$\begin{aligned} \underset{w,b}{\text{minimizar}} \quad & \frac{1}{2}\|w\|^2 \\ \text{sujeito a} \quad & y_i\left(w^T x_i + b\right) \geq 1, \quad i = 1, \ldots, m, \end{aligned} \tag{2.2}$$

em que $w \in \mathbb{R}^n, b \in \mathbb{R}$.

Nesta seção foi apresentado o modelo da Máquina de Vetores Suporte para margens rígidas, que pode ser considerado o mais simples e básico, já que os dados estão distribuídos de forma linearmente separável e consequentemente, existe um classificador linear que separa esses pontos.

2.1.2 SVM para Margens Flexíveis

Na seção anterior foi abordado o caso em que o conjunto de dados está distribuído de forma linearmente separável. Porém em aplicações reais isto raramente ocorre. Diante disso, será exposta uma modificação ao problema (2.2) a fim de obter um classificador linear para casos onde os dados não são linearmente separáveis, permitindo alguns erros de classificação. Esta situação pode ser observada na Figura 2.4.

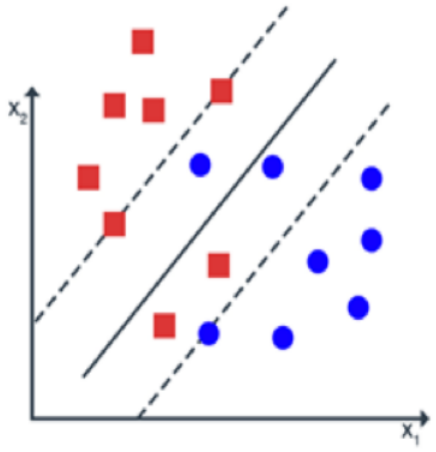

Figura 2.4: Ilustração para o caso das Margens Flexíveis. Fonte: `https://blog.quantinsti.com/support-vector-machines-introduction/`

Sobre as mesmas condições propostas anteriormente em (2.1), são acrescentadas às restrições do problema variáveis de folga $\xi_i \geq 0$, $i = 1, \ldots, m$. Com o objetivo de suavizar as restrições impostas na obtenção do hiperplano ótimo.

Estas variáveis de folga são penalizadas na função objetivo por uma constante de regularização $C > 0$ que pondera a minimização dos erros de classificação. O problema de Otimização para margens flexíveis tem expressão

$$\begin{aligned} \underset{w,b,\xi}{\text{minimizar}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{sujeito a} \quad & y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi \geq 0, \end{aligned}$$

em que $w \in \mathbb{R}^n, b \in \mathbb{R}, \xi \in \mathbb{R}^m$. Vale destacar que as margens flexíveis são utilizadas quando há poucos dados no conjunto, pois neste caso é altamente provável que o uso de um classificador não linear produza um *overfitting*, isto é, o modelo se ajustaria bem para o conjunto de dados mas se tornaria ineficiente em prever novos resultados.

Nesta seção foi apresentada uma adaptação do modelo das margens rígidas para o caso não linearmente separável, com poucos dados, obtendo um classificador linear que separa o conjunto de dados com alguns erros nesta classificação. A seguir será apresentado um novo modelo, desta vez não linear, que é usado em diversas aplicações práticas, principalmente quando a base contém um número maior de dados.

2.1.3 SVM Não Linear

Até o momento foi apresentado o modelo SVM Linear, que resolve um problema de Otimização simples a fim de obter um classificador linear $f(x) = w^T x + b$. Porém, nos problemas de classificação mais elaborados e reais, os dados não estão distribuídos de forma linearmente separável, então encontrar um hiperplano que separe esses dados se torna uma tarefa impossível com a abordagem anterior.

Para realizar esta tarefa de obter o classificador ótimo para o caso não linear, usaremos o "truque do *kernel*". A ideia principal será transformar um conjunto de dados não linearmente separável em outro linearmente separável aumentando a dimensão do problema, usando uma função específica chamada de função *kernel*. A seguir, na Figura 2.5, está a visualização geométrica deste truque.

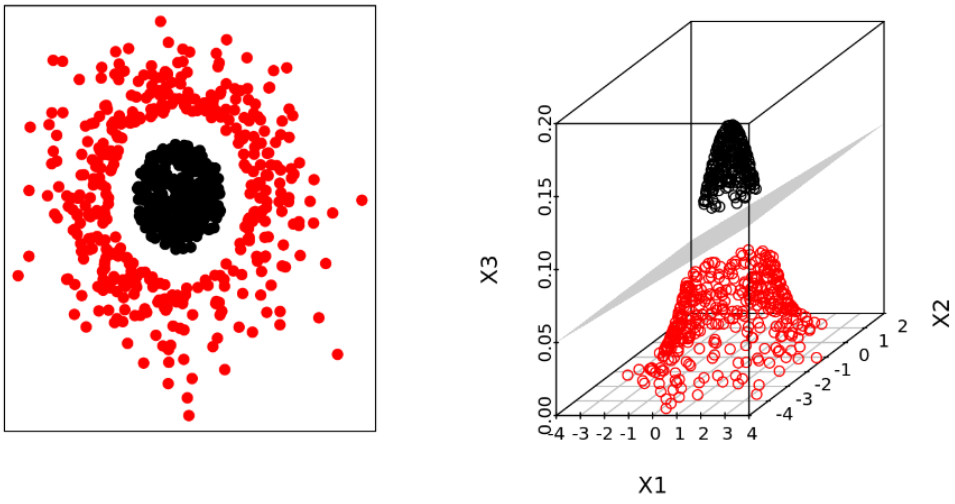

Figura 2.5: Representação gráfica do "Truque do *Kernel*". Fonte: `https://rpubs.com/Joaquin_AR/267926`

Considere um conjunto de dados $X \subset \mathbb{R}^n$ que não é linearmente separável, então o "truque do *kernel*" consiste em definir uma transformação não-linear $\phi : \mathbb{R}^n \to \mathcal{S}$, que transforma o espaço original em outro onde os pontos se tornam linearmente separáveis, onde $\mathcal{S}$ é o novo espaço com $\dim(\mathcal{S}) > n$. O hiperplano neste novo espaço tem expressão $w^T \phi(x_i) + b = 0$. Desta forma, para as margens rígidas, o problema pode ser formulado como

$$\begin{array}{ll} \underset{w,b}{\text{minimizar}} & \frac{1}{2}\|w\|^2 \\ \text{sujeito a} & y_i\left(w^T\phi(x_i)+b\right) \geq 1, \quad i=1,\ldots,m. \end{array}$$

Para as margens flexíveis o problema de Otimização resultante é dado por

$$\begin{array}{ll} \underset{w,b,\xi}{\text{minimizar}} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{sujeito a} & y_i\left(w^T\phi(x_i)+b\right) \geq 1-\xi_i, \quad i=1,\ldots,m \\ & \xi \geq 0. \end{array} \tag{2.3}$$

Como a função $\phi(x)$ nem sempre é conhecida e normalmente difícil de determinar, através do "truque do *kernel*" é obtida a chamada função *kernel* dada por:

$$k(x_i,x_j) = \phi(x_i)^T\phi(x_j).$$

Consequentemente a matriz *kernel* que é simétrica semidefinida positiva $K \in \mathbb{R}^{m\times m}$ dada por

$$K_{ij} = k(x_i,x_j).$$

Tomando $H_{ij} = y_iy_j \cdot k\left(x_i,x_j\right)$ e $e_j = 1$, para $i,j = 1,\ldots,m$, reformulando o problema (2.3) nas variáveis duais α,

$$\begin{array}{ll} \underset{\alpha}{\text{minimizar}} & \frac{1}{2}\alpha^T H\alpha - e^T\alpha \\ \text{sujeito a} & y^T\alpha = 0 \\ & 0 \leq \alpha \leq C. \end{array} \tag{2.4}$$

Para resolver o problema de Otimização (2.4) é preciso ver qual função *kernel* melhor se adapta ao problema em questão. A tabela a seguir mostra as funções mais utilizadas

Kernel	$\mathbf{k(x_i,x_j)}$	**Parâmetros**
Linear	$x_i^Tx_j$	Nenhum
Polinômio de grau p	$((x_i^Tx_j)+\eta)^p$	η = constante
Sigmoidal	$\tanh(\gamma(x_i^Tx_j)+\eta)$	$\gamma > 0, \eta$ = constante
Gaussiano (RBF - *Radial Basis Function*)	$\exp(-\gamma\|x_i-x_j\|^2)$	$\gamma > 0$

Tabela 2.1: Funções *Kernel* mais utilizadas.

Neste capítulo foram estudadas as Máquinas de Vetores Suporte e suas diversas formulações para distintas distribuições de dados. A seguir serão apresentadas às Máquinas de Vetores Suporte Transdutivas, uma extensão do método SVM já estudado, mas para o caso em que o problema de Aprendizagem de Máquina é semi-supervisionado.

Capítulo 3

Máquinas de Vetores Suporte Transdutivas

O estudo realizado neste capítulo engloba as Máquinas de Vetores Suporte Transdutivas, uma generalização do modelo SVM previamente visto, para bases de dados com informações incompletas. Tendo em vista os trabalhos feitos por Collobert et al. [4] e Joachims [8], será apresentado o modelo e um método para abordá-lo.

3.1 Introdução às Máquinas de Vetores Suporte Transdutivas

No contexto das Máquinas de Vetores Suporte Transdutivas (*Transductive SVM* - TSVM), grande parte do banco de dados não está rotulada, isto é, trata-se de um problema semi-supervisionado, impossibilitando o uso do modelo anterior. Diante disso, Collobert [4] e Joachims [8] propuseram este novo modelo (TSVM) a fim de obter, mesmo com falta de informação no conjunto de dados, um classificador binário.

Considere um vetor x_i em $\mathbb{R}^n$ da forma $x_i = \left(x_1^{(i)}; \ldots; x_n^{(i)}\right)^T$ com seu respectivo rótulo $y_i \in \{-1, 1\}$ para $i = 1, \ldots, L$. Suponha então dois conjuntos

$$\mathcal{L} = \left\{(x_1; y_1), \ldots, (x_L; y_L)\right\} \quad \text{e} \quad \mathcal{U} = \left\{x_{L+1}, \ldots, x_{L+U}\right\},$$

sendo que $\mathcal{L}$ representa o conjunto dos L **dados rotulados** e $\mathcal{U}$ o conjunto dos U **dados não rotulados**.

Deseja-se obter um hiperplano da forma $w^T x + b = 0$ que separe os dados, mas usando os dados rotulados e não rotulados para obter o classificador, como representado na Figura 3.1.

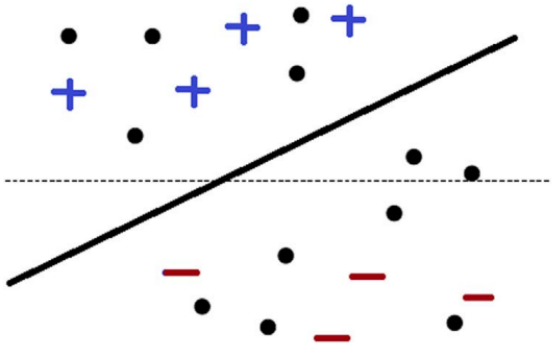

Figura 3.1: Hiperplano obtido pelo modelo TSVM. Fonte: [2]

O hiperplano ótimo para o modelo TSVM irá separar os dados rotulados tomando a margem máxima e forçando os dados não rotulados a estarem o mais longe possível desta margem para evitar erros de classificação. Observe que na Figura 3.1 o hiperplano (reta em preto) está separando os dados em classe positiva (sinal positivo em azul) e classe negativa (sinal negativo em vermelho), simultaneamente levando em conta a distribuição dos dados não rotulados (em preto) com relação à margem.

Similarmente com o modelo SVM exposto no capítulo anterior, existem os casos das **margens rígidas**, **margens flexíveis** e o **caso não linear**, mas o foco deste trabalho será o estudo do caso **TSVM não linear**, que possui a formulação

$$\begin{aligned} \underset{w,b,\xi,\xi^*}{\text{minimizar}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{j=L+1}^{L+U}\xi_j^* \\ \text{sujeito a} \quad & y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, \quad i = 1,\ldots,L \\ & \left|w^T x_i + b\right| \geq 1 - \xi_j^*, \quad j = L+1,\ldots,L+U \\ & \xi, \xi^* \geq 0 \end{aligned} \tag{3.1}$$

com $w \in \mathbb{R}^{L+U}, b \in \mathbb{R}$. Onde o parâmetro C penaliza os erros de classificação dos dados rotulados, enquanto que C^* penaliza os dados não rotulados que estão entre as margens. Usando a função $H_s(t) = \max(0, s-t)$, em particular quando $s = 1$, $H_1(t)$ chamada de Função Perda de Articulação (*Hinge Loss Function*), que mede os erros cometidos na classificação do modelo. É possível incorporar as restrições do problema de Otimização (3.1) e reescrevê-lo como outro equivalente de expressão

$$\underset{w,b}{\text{minimizar}} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} H_1\left(y_i(w^T x_i + b)\right) + C^*\sum_{i=L+1}^{L+U} H_1\left(|w^T x_i + b|\right). \tag{3.2}$$

Para uma grande base de dados, é possível que o classificador obtido após resolver o problema (3.2) classifique todos os dados não rotulados como pertencendo apenas a uma classe, o que é indesejável e consequentemente levará a erros de classificação. Diante disso, para evitar que isso ocorra, é conveniente adicionar ao modelo a seguinte restrição:

$$\frac{1}{U}\sum_{i=L+1}^{L+U}(w^T x_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i. \tag{3.3}$$

Segundo estudos feitos em [4], a restrição (3.3) garante que os dados não rotulados sejam classificados em ambas classes, respeitando a distribuição dos dados que estão

previamente rotulados. É importante ressaltar que o problema (3.2) não é convexo e a função $H_1(|\cdot|)$ não é diferenciável, logo resolvê-lo é uma tarefa desafiadora. Devido a esse fato o problema de Otimização será reformulado usando o Método *Concave-Convex Procedure* (CCCP).

3.2 Método *Concave-Convex Procedure*

O método CCCP supõe que uma função objetivo F pode ser reescrita como uma soma de uma parte convexa (F_{vex}) e outra côncava (F_{cav}). O método então substitui a parte côncava pela sua aproximação tangente e minimiza a função resultante, que agora é convexa, ou seja, resolve o problema

$$\arg\min_{x} \left(F_{\text{vex}}(x) + F'_{\text{cav}}(x) \cdot x\right).$$

Novamente vamos reescrever o problema (3.2) para que fique no formato necessário a fim de poder aplicar o método.

A função *hinge loss* $H_1(|\cdot|)$, referente aos dados não rotulados, será substituída pela função denominada de Função *Ramp Loss* dada por

$$R_s(\cdot) = \min(1-s\,,\, \max(0\,,\, 1-\cdot)) = H_1(\cdot) - H_s(\cdot).$$

Na Figura 3.2 estão representados os gráficos das funções *ramp loss* e *hinge loss* convexa, respectivamente, usando $s = -0.3$.

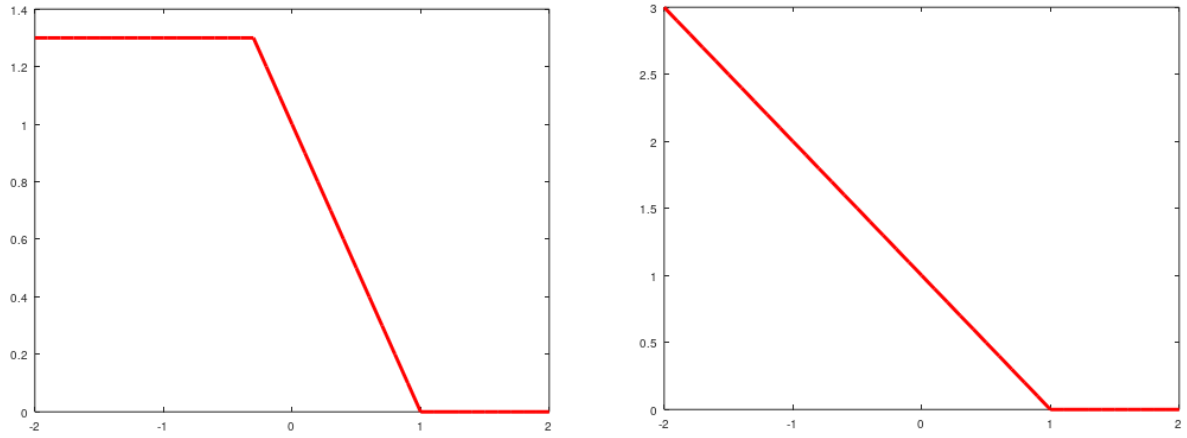

Figura 3.2: Função *Ramp Loss* e Função *Hinge Loss*. Fonte: O Autor.

Para treinar o modelo TSVM, é usada a função R_s para os dados não rotulados, que serão classificados de forma preliminar como pertencendo a ambas às classes (positiva e negativa), isto é:

$$\begin{aligned} y_i &= +1\,, \quad i \in \{L+1, \ldots, L+U\}\,, \\ y_i &= -1\,, \quad i \in \{L+U+1, \ldots, L+2U\}\,, \\ x_i &= x_{i-U}\,, \quad i \in \{L+U+1, \ldots, L+2U\}. \end{aligned}$$

Assim, reescrevendo o problema de Otimização (3.2) obtém-se:

$$\underset{w,b}{\text{minimizar}} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} H_1\left(y_i(w^T x_i + b)\right) + C^*\sum_{i=L+1}^{L+2U} R_S\left(y_i(w^T x_i + b)\right). \tag{3.4}$$

Definindo $\theta = (w; b)$, seja $F(\theta) = F_{\text{vex}}(\theta) + F_{\text{cav}}(\theta)$ a função objetivo a ser minimizada no problema (3.4). Lembrando que $R_s = H_1 - H_s$, tem-se que a função pode ser separada por sua parte convexa e côncava, respectivamente:

$$F_{\text{vex}}(\theta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} H_1\left(y_i(w^T x_i + b)\right) + C^*\sum_{i=L+1}^{L+2U} H_1\left(y_i(w^T x_i + b)\right),$$

$$F_{\text{cav}}(\theta) = -C^*\sum_{i=L+1}^{L+2U} H_s\left(y_i(w^T x_i + b)\right).$$

Diante disso, tem-se que $F(\theta) = F_{\text{vex}}(\theta) + F_{\text{cav}}(\theta)$, logo é possível aplicar o Método CCCP para resolver o problema de Otimização (3.4). O processo é iterativo e a partir de uma aproximação inicial θ^0, se determina o parâmetro $\theta = (w; b)$ na $(m+1)$-ésima iteração resolvendo

$$\theta^{m+1} = \arg\min_{\theta} \left(F_{\text{vex}}(\theta) + F'_{\text{cav}}(\theta^m)\cdot\theta\right),$$

sujeito a restrição (3.3) exposta anteriormente. Vale notar que este método é iterativo e gera sequências convergentes (v. Yuille e Rangarajan (2002)). Considere $f_\theta(x) = w^T x + b$, então a substituição das respectivas funções, F_{vex} e F_{cav} na expressão anterior resulta em

$$F'_{\text{cav}}(\theta) = \frac{\partial F_{\text{cav}}(\theta)}{\partial\theta} = -C^*\sum_{i=L+1}^{L+2U}\frac{\partial F_{\text{cav}}(\theta)}{\partial f_\theta(x_i)}\cdot\frac{\partial f_\theta(x_i)}{\partial\theta}$$

Denotando $\dfrac{\partial F_{\text{cav}}(\theta)}{\partial f_\theta(x_i)} = \beta_i y_i$, logo

$$\begin{aligned}
\beta_i &= y_i\frac{\partial F_{\text{cav}}(\theta)}{\partial f_\theta(x_i)} \\
&= \begin{cases} C^* H'_s(y_i f_\theta(x_i)), & \text{se} \quad i \geq L+1 \\ 0, & \text{caso contrário} \end{cases} \\
&= \begin{cases} C^*, & \text{se} \quad y_i f_{\theta^{m+1}}(x_i) < s \quad \text{e} \quad i \geq L+1 \\ 0, & \text{caso contrário.} \end{cases}
\end{aligned}$$

Assim, cada iteração do método CCCP é encontrada minimizando

$$\begin{aligned}
F_{\text{vex}}(\theta) + \frac{\partial F_{\text{cav}}(\theta)}{\partial\theta}\cdot\theta &= F_{\text{vex}}(\theta) + \left(\sum_{i=L+1}^{L+2U}\beta_i y_i\cdot\frac{\partial f_\theta(x_i)}{\partial\theta}\right)\cdot\theta \\
&= F_{\text{vex}}(\theta) + \sum_{i=L+1}^{L+2U}\beta_i y_i\cdot f_\theta(x_i)
\end{aligned}$$

sujeito a restrição (3.3). Portanto, combinando ambas equações, é obtido o problema de Otimização,

$$\begin{aligned} \underset{\theta,\xi}{\text{minimizar}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i + \sum_{i=L+1}^{L+2U}\beta_i y_i f_\theta(x_i) \\ \text{sujeito a} \quad & \frac{1}{U}\sum_{i=L+1}^{L+U} f_\theta(x_i) = \frac{1}{L}\sum_{i=1}^{L} y_i \\ & y_i f_\theta(x_i) \geq 1-\xi_i, \quad 1 \leq i \leq L+2U \\ & \xi_i \geq 0, \quad 1 \leq i \leq L+2U. \end{aligned} \tag{3.5}$$

Obter o classificador equivale a resolver iterativamente o problema (3.5) reescrito nas variáveis duais α (v. [4], Apêndice A), sendo que $\alpha_i = y_i\hat{\alpha}_i + \beta_i$, para $i = 1, \ldots, L+U$ e o vetor ζ dado por, $\zeta_0 = \frac{1}{L}\sum_{i=1}^{L} y_i$; $\zeta_i = y_i$, para $i = 1, \ldots, L+2U$. Logo será resolvido, utilizando o comando ***quadprog*** do *software* Octave, o problema de Otimização

$$\min_{\hat{\alpha}}\left(\frac{1}{2}\hat{\alpha}^T K\hat{\alpha} - \zeta^T\hat{\alpha}\right) \quad \text{s.a} \quad \begin{cases} \sum_{i=0}^{L+2U}\hat{\alpha}_i = 0 \\ 0 \leq y_i\hat{\alpha}_i \leq C, \quad 1 \leq i \leq L \\ -\beta_i \leq y_i\hat{\alpha} \leq C^* - \beta_i, \quad i \geq L+1. \end{cases}$$

Com os elementos discutidos, é obtido o seguinte algoritmo.

Algoritmo 1: Método *Concave-Convex Procedure* para TSVM

- **Passo 1:**

  Inicializar os parâmetros, a base de dados e obter o classificador inicial usando SVM.

  Definir: $\beta_i^0 = \begin{cases} C^*, & \text{se} \quad y_i f_{\theta^0}(x_i) < s \quad \text{e} \quad i \geq L+1 \\ 0, & \text{caso contrário} \end{cases}$

  Definir: $\zeta_0 = \frac{1}{L}\sum_{i=1}^{L} y_i \quad ; \quad \zeta_i = y_i \quad , \quad 1 \leq i \leq L+2U$

- **Passo 2**:

  Resolver para $K_{ij} = \exp(-\gamma\|x_i - x_j\|^2)$:

  $$\min_{\hat{\alpha}}\left(\frac{1}{2}\hat{\alpha}^T K\hat{\alpha} - \zeta^T\hat{\alpha}\right) \text{ s.a } \begin{cases} \langle\hat{\alpha}, 1\rangle = 0 \\ 0 \leq y_i\hat{\alpha}_i \leq C, \quad 1 \leq i \leq L \\ -\beta_i^m \leq y_i\hat{\alpha} \leq C^* - \beta_i^m, \quad i \geq L+1 \end{cases}$$

  Definir: $\hat{\alpha}_i = y_i(\alpha_i - \beta_i^m) \quad ; \quad b^{m+1} = y_i - K\alpha_i \quad ; \quad f_{\theta^{m+1}}(x_i) = \alpha_i K + b^{m+1}.$

  Definir: $\beta_i^{m+1} = \begin{cases} C^*, & \text{se} \quad y_i f_{\theta^{m+1}}(x_i) < s \quad \text{e} \quad i \geq L+1 \\ 0, & \text{caso contrário} \end{cases}$

  Repetir o **Passo 2**, até que $\beta^{m+1} = \beta^m$.

Neste capítulo foram apresentadas as Máquinas de Vetores Suporte Transdutivas, junto ao Método CCCP usado para abordar o problema não-convexo. No seguinte capítulo serão feitos experimentos numéricos referentes a um problema semi-supervisionado, com o objetivo de analisar o desempenho do algoritmo implementado.

Capítulo 4

Experimentos Numéricos

Com o objetivo de medir o desempenho do algoritmo CCCP para TSVM exposto no capítulo anterior, neste capítulo são feitos experimentos numéricos computacionais. Na primeira seção é realizado um experimento preliminar usando uma base de dados criada pelo autor. Na segunda seção, o algoritmo é treinado com uma base de dados clássica utilizada na Ciência de Dados chamada de MNIST.

Na resolução do problema de Otimização referente ao modelo TSVM, foi utilizado o *kernel* Gaussiano (RBF), já que na literatura tem fornecido os melhores resultados (v. [7]). O Algoritmo 1 descrito no capítulo anterior foi inicialmente implementado em C++ por Collobert et al. [4], e adaptado pelo autor deste trabalho no *software* Octave, na implementação foi utilizada a biblioteca livre das Máquinas de Vetores Suporte conhecida como LIBSVM [3, 7]. O código está descrito no Apêndice A deste trabalho e disponível em `https://github.com/Maxim-Lobkov/TSVM-CCCP`.

4.1 Experimento Preliminar

Para começar a testar o desempenho do algoritmo, foi gerado um conjunto com alguns dados distribuídos aleatoriamente no interior de uma circunferência de expressão $x^2 + y^2 = 2$, para $x, y \in \mathbb{R}$, e outros fora dela. Especificamente a base de dados é composta por 200 dados rotulados e 150 não rotulados, estes dados estão ilustrados na Figura 4.1.

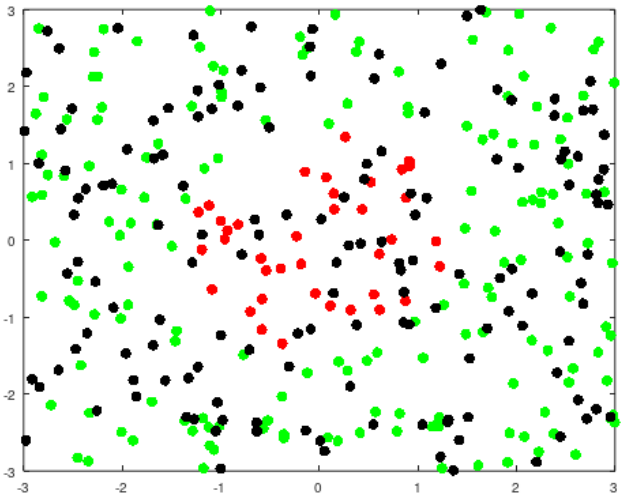

Figura 4.1: Base de Dados com pontos separados por uma circunferência. Fonte: O Autor.

Os pontos em vermelho representam os dados pertencentes à classe positiva, os pontos em verde à classe negativa e os pontos em preto correspondem aos dados não rotulados. Como foi visto no capítulo anterior, o objetivo do algoritmo é obter um classificador separando os conjuntos e usar esse classificador para rotular os dados não rotulados. Desta forma, na Figura 4.2 observa-se o resultado obtido após executar o algoritmo sobre a base de dados definida.

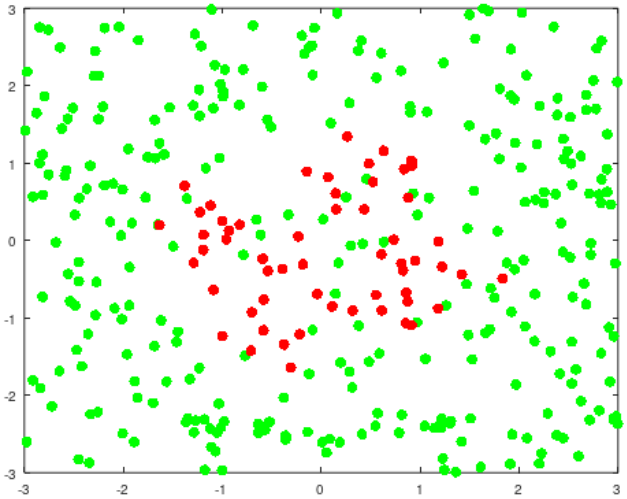

Figura 4.2: Rotulação dos dados propostos usando o algoritmo CCCP para TSVM. Fonte: O Autor.

Para esta base de dados foram utilizados os parâmetros: $s = -0.4$, $C = 2.597$, $C^* = 0.3$, $\gamma = 3$ e na rotulação foram obtidos 124 acertos, o que equivale a uma taxa de acerto de 82.6%. Neste experimento não foi utilizado o método de busca por parâmetros, o método *Grid Search*. Para obter estes parâmetros foi realizada uma busca heurística, por esta razão alguns dados foram classificados de forma incorreta. Vale destacar que o número de acertos foi determinado com base nos rótulos verdadeiros dos dados não rotulados, que foram ocultados do algoritmo durante o treinamento.

4.2 Base de Dados MNIST

Na Aprendizagem de Máquina existem diversas bases de dados clássicas que são utilizadas para treinar modelos. Neste trabalho será usada uma base de dados dos dígitos manuscritos, conhecida como MNIST. O conjunto de dados é formado por 70000 imagens de dígitos manuscritos (de 0 a 9). Para a máquina, estes dados são representados por uma matriz de tamanho 70000×784, onde as primeiras 60000 linhas representam o conjunto de treinamento e as últimas 10000 linhas representam o conjunto de teste. Em cada linha desta matriz está armazenada vetorialmente uma imagem de tamanho 28×28 que corresponde à representação de algum número. Na Figura 4.3 estão representadas algumas destas imagens do conjunto.

Vale notar que inicialmente o conjunto de dados já está rotulado, porém, neste experimento os rótulos do conjunto de teste foram "escondidos" da máquina, a fim de obter um problema semi-supervisionado. A alteração na base de dados foi feita para poder realizar uma comparação entre a rotulação real e a rotulação obtida pelo método.

Figura 4.3: Representação de algumas amostras do conjunto de dados MNIST. Fonte: [6].

A proposta do experimento é classificar os dados do conjunto MNIST como sendo **"5"** ou **"não 5"**, representados pelas classes positiva e negativa, respectivamente. Na base de dados cada dígito possui várias formas diferentes de ser representado, na Figura 4.4 há um exemplo de uma amostra do conjunto que deve ser classificada como sendo o dígito 5.

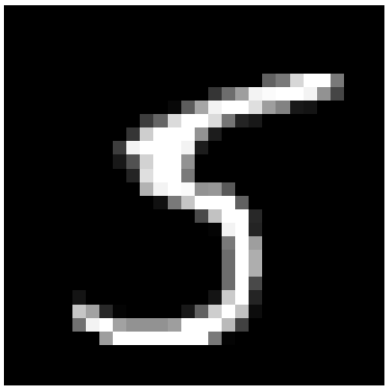

Figura 4.4: Exemplo de uma amostra que é rotulada como 5. Fonte: O Autor

É importante ressaltar que o experimento proposto nesta seção foi realizado usando o *software* Octave que não possui a base de dados MNIST pré instalada, por esta razão os dados foram baixados pelo *GitHub*, `https://github.com/daniel-e/mnist_octave`.

No experimento proposto foram utilizados os primeiros 300 dados da matriz, sendo que 100 são dados rotulados e 200 são não rotulados, na Tabela 4.1 estão os resultados obtidos ao executar o algoritmo nesta base de dados, para diferentes valores de parâmetros.

Parâmetros	**Número de Acertos**	**Porcentagem de Acertos**
$s = -0.5$, $C = 10$, $C^* = 0.3$, $\gamma = 0.03$	91	45.5%
$s = -0.1$, $C = 10$, $C^* = 0.1$, $\gamma = 0.0128$	97	48.5%
$s = -0.3$, $C = 10$, $C^* = 0.1$, $\gamma = 0.03$	127	63.5%
$s = -0.4$, $C = 2.597$, $C^* = 0.3$, $\gamma = 0.03$	145	72.5%
$s = -0.4$, $C = 2.597$, $C^* = 0.2$, $\gamma = 0.01$	180	90%

Tabela 4.1: Resultados da base de dados MNIST aplicando o Método CCCP.

Vale notar que o algoritmo implementado possui um alto custo computacional e que durante estes testes não foi usado o método *Grid Search*, que faz uma busca eficaz de parâmetros. Por estas razões foram escolhidos poucos dados. Diante disso, o próximo teste foi realizado fixando os parâmetros que forneceram os melhores acertos e variando a quantidade de dados rotulados e não rotulados, a fim de analisar o desempenho do método para distintas quantidades de dados. Estes resultados estão indicados na Tabela 4.2.

Rotulados	**Não Rotulados**	**Número de Acertos**	**Porcentagem de Acertos**
180	120	53	44.1%
120	180	82	45.5%
50	250	211	84.4%
20	280	245	87.5%

Tabela 4.2: Análise do número de dados aplicando o Método CCCP.

Observe que a medida em que a quantidade de dados não rotulados aumenta, o método CCCP fornece uma maior porcentagem de acertos. Isto se deve ao fato de que para a máquina, rotular um conjunto de teste com base em padrões encontrados em ambos os conjuntos, de treinamento e de teste, se torna uma tarefa mais fácil do que obter um modelo a partir do conjunto de treinamento, para poder aplicá-lo ao conjunto de teste (Vapnik, 1982). Portanto, para TSVM o algoritmo CCCP possui melhor desempenho quando o conjunto de dados não rotulados supera a quantidade de dados rotulados.

Conclusão

O objetivo deste trabalho foi estender a formulação do modelo das Máquinas de Vetores Suporte, que é um problema supervisionado, para poder aplicá-lo a um problema de Aprendizagem de Máquina semi-supervisionada. Para isso foi feito o estudo das Máquinas de Vetores Suporte Transdutivas, que permite obter um classificador binário utilizando uma base de dados com dados rotulados e não rotulados. Esse classificador foi obtido após resolver um problema de Otimização que inicialmente não era convexo utilizando o Método *Concave-Convex Procedure*, que substitui a parte côncava da função objetivo por sua aproximação tangente. O método foi implementado no *software* Octave e para medir seu desempenho foram feitos alguns experimentos numéricos, concluindo que o algoritmo apresenta melhores resultados quando o conjunto de dados não rotulados supera a quantidade de dados rotulados. O próximo passo para continuar o estudo seria de buscar outros métodos que resolvem o modelo TSVM, a fim de comparar o desempenho entre os métodos e também implementar uma busca eficiente dos parâmetros presentes na formulação do modelo, usando por exemplo o Método *Grid Search*.

Referências Bibliográficas

[1] BENATTI, N. M. ***Métodos de Busca Direta para Seleção de Parâmetros em Máquinas de Vetores Suporte***. Curitiba, UFPR: Dissertação, 2017.

[2] CEVIKALP, H. e FRANC, V. ***Large-scale robust transductive support vector machines***. Neurocomputing 235, 2017. pp. 199-209.

[3] CHANG, C. e LIN, C. ***LIBSVM:*** *a library for support vector machines*. ACM Transactions on Intelligent Systems and Technology, 2011. `https://www.csie.ntu.edu.tw/~cjlin/libsvm/` .

[4] COLLOBERT, R. , SINZ, F. , WESTON, J. e BOTTOU, L. ***Large Scale Transductive SVMs***. Journal of Machine Learning Research 7, 2006. pp. 1687-1712.

[5] CRISTIANINI N, SHAWE-TAYLOR J. ***An introduction to support vector machines and other kernel-based learning methods***. Cambridge University Press; 2000.

[6] GÉRON, A. ***Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow:*** *Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly, 2ª Edição, 2019.

[7] HSU, C. , CHANG, C. e LIN, C. ***A Practical Guide to Support Vector Classification***. Universidade Nacional de Taiwan, 2016.

[8] JOACHIMS, T. ***Transductive Inference for Text Classification using Support Vector Machines*** . International Conference on Machine Learning, ICML, 1999.

[9] VAPNIK, V. N. e LERNER, A.Y. ***Recognition of Patterns with help of Generalized Portraits***. Avtomat. i Telemekh. V 24, Issue 6, 1963. pp. 774–780.

Apêndice A

Códigos

1. Base de Dados 1 implementado em Octave:

```
%%% Base de dados aleatória em formato de circunferência %%%
n = 200;
for i = 1:n
  xr(i,:) = rand(1,2)*6 - 3;
  if (xr(i,1))^2 + (xr(i,2))^2 <= 2
    y(i) = 1;
  else
    y(i) = -1;
  endif
endfor

% Definindo rótulos dos dados de teste para a comparação: %
m = 150;
for i = 1:m
  xnr(i,:) = rand(1,2)*6 - 3;
  if (xnr(i,1))^2 + (xnr(i,2))^2 <= 2
    yytest(i) = 1;
  else
    yytest(i) = -1;
  endif
endfor

ytest = yytest';
X = [xr y'];
indn = y < 0;
indp = y > 0;

% Plotagem dos pontos %
plot(X(indn,1),X(indn,2),'g.','markersize',12,X(indp,1),X(indp,2),'r.','markersize',12)
hold on
plot(xnr(:,1),xnr(:,2),'k.','markersize',12)
```

1. Base de Dados 2 implementado em Octave:

```
%% Dados MNIST com classificação "5" ou "não 5" %%
d = load('mnist.mat');
%% Dados: %%
xr = d.trainX;
xnr = d.testX;

yy = d.trainY;
y5 = yy';
y = ones(length(y5),1);

ytest = d.testY;
y5test = ytest';
yytest = ones(length(y5test),1);

for i = 1:(length(y5)) %% Definindo rótulos %%
  if (y5(i) == 5)
    y(i) = 1;
  else
    y(i) = -1;
  endif
endfor

%% Definindo rótulos dos dados de teste para a comparação: %%
for i = 1:(length(y5test))
  if (y5test(i) == 5)
    yytest(i) = 1;
  else
    yytest(i) = -1;
  endif
endfor

%% Saída de dados %%
xr = im2double(xr);
y = im2double(y);
xnr = im2double(xnr);
yytest = im2double(yytest);
```

1. Algoritmo CCCP para TSVM implementado em Octave:

```
%%% Algoritmo de Concave-Convex Procedure para TSVM %%%
%%% xr: dados de treinamento(rotulados); y: rótulos dos dados de treino;
xnr: dados não rotulados %%%
%%% Parâmetros: s,C,CC = C*,gamma %%%
function [fn,Xrot,yrot] = cccp(xr,y,xnr,s,C,CC,gamma)
  L = rows(xr); % Número de dados rotulados %
  U = rows(xnr); % Número de dados não rotulados %

  % Matrizes com dados rotulados e não rotulados: %
  ynr = [ones(U,1);-ones(U,1)];
  xnrt = [xnr;xnr];
  X = [xr y;xnrt ynr];
  [m,n] = size(X);
  yt = X(:,n);
  xt = X(:,1:(n-1));

  % SVM nos dados de treinamento: %
  modelo = svmtrain(y,xr,cstrcat("-g ", num2str(gamma)," -c ", num2str(C)));
  [~,~,f0] = svmpredict(-ones(rows(xt),1),xt,modelo); % Obtendo o classificador inicial %

  for i = 1:(L+2*U)
    if (i >= L+1 && yt(i)*f0(i) < s) % Inicializando beta %
      beta0(i) = CC;
    else
      beta0(i) = 0;
    endif
  endfor

  zeta0 = 0;
  for i = 1:L % Obtendo Zeta %
    zeta0 = zeta0 + (1/L)*y(i);
  endfor

  for i = 1:(L+2*U)
    zeta(i) = yt(i);
  endfor
  zt = [zeta0 zeta]'; % Vetor Zeta %

  % Matriz Kernel: %
  for i= 1:(L + 2*U)
    K(i,i) = 1;
    for j = (i+1):(L+2*U)
      K(i,j) = exp(-gamma*((xt(i,:) - xt(j,:))*(xt(i,:) - xt(j,:))'));
      K(j,i) = K(i,j);
    endfor
  endfor

  % Matriz Kernel com o elemento adicionado x0: %
  vec = 0;
  Kt(1,1) = 1;
  for i = 2:(L+2*U+1)
    for j = (L+1):(L+U)
      vec = vec + (1/U)*exp(-gamma*((xt(j,:) - xt(i-1,:))*(xt(j,:) - xt(i-1,:))'));
      Kt(i,1) = vec;
    endfor
    Kt(1,i) = Kt(i,1);
    for j = 2:(L+2*U+1)
      Kt(i,j) = K(i-1,j-1);
    endfor
  endfor

  sair = 0;
  % yt e beta0 com o elemento adicionado x0: %
  yyt = [1;yt];
  bbt0 = [0;beta0'];

  iter = 0;
  % Resolvendo o problema convexo TSVM cm CCCP: %
  while sair == 0
     % Entradas para minimizar o problema convexo: %
     A = []; B = [];
```

```
   AEQ = ones(1,L+2*U+1); BEQ = 0;
   LB = min([C*ones(L+1,1);CC*ones(2*U,1)].*yyt-bbt0,zeros(L+2*U+1,1));
   UB = max([C*ones(L+1,1);CC*ones(2*U,1)].*yyt-bbt0,zeros(L+2*U+1,1));

   % Minimizando o Problema Convexo retornando at: %
   [at] = quadprog(Kt,-zt,A,B,AEQ,BEQ,LB,UB);

   % Solução do problema convexo original: %
   alpha = at.*yyt + bbt0;
   % Componentes (índices) de alpha: %
   ind = yyt.*alpha > -bbt0 & yyt.*alpha < [C*ones(L+1,1);CC*ones(2*U,1)] - bbt0;
   bn = mean(yyt(ind) - Kt(ind,:)*alpha); % Novo b %
   fn = Kt*alpha + bn; % Novo f %
   yyt = [yyt(1:(L+1));sign(fn(L+2:end))]; % Rótulo %

   % Atualizando Beta: %
   for i = 1:(L+2*U+1)
     if (i >= L+2 && yyt(i)*fn(i) < s)
       bbt(i) = CC;
     else
       bbt(i) = 0;
     endif
   endfor
   % Critério de Parada: %
   if sum(bbt0 == bbt) == (L + 2*U + 1)
     sair = 1;
   else
     bbt0 = bbt';
   endif
   iter = iter + 1
endwhile

%% Rotulando os dados %%
yrot = yyt(2:L+U+1);
xx = [xr;xnr];
Xrot = [xx yrot];
```