

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Francisco Acélio Cunha de Sousa

Um Novo Algoritmo de Treinamento para Máquinas de Vetores Suporte

Fortaleza – CE

13 de setembro de 2024

Francisco Acélio Cunha de Sousa

Um Novo Algoritmo de Treinamento para Máquinas de Vetores Suporte

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Inteligência Artificial

Orientador: Prof Dr.Ajalmar Rêgo da Rocha Neto

Coorientador: Prof Dr.Thiago Alves Rocha

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

13 de setembro de 2024

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

C972n Cunha de Sousa, Francisco Acélio.

Um Novo Algoritmo de Treinamento para Máquinas de Vetores Suporte : Novos algoritmos de treinamento para SVM / Francisco Acélio Cunha de Sousa. - 2024.
91 f. : il. color.

Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus Fortaleza, 2024.

Orientação: Prof. Dr. Dr.Ajalmar Rêgo da Rocha Neto.

Coorientação: Prof. Dr. Dr.Thiago Alves Rocha.

1. Aprendizado de máquina; máquinas de vetores suporte; classificação; otimização sequencial mínima; algoritmos de treinamento.. I. Título.

CDD 004



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
Av. 13 de maio, 2081 - Bairro Benfica - CEP 60040-531 - Fortaleza - CE - www.ifce.edu.br

FOLHA DE APROVAÇÃO

Processo: 23256.008346/2024-35

Interessado: Francisco Acélio Cunha de Sousa

FRANCISCO ACÉLIO CUNHA DE SOUSA

Um Novo Algoritmo de Treinamento para Máquinas de Vetores Suporte

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) *Campus* de Fortaleza, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação.

Aprovada em: **13/09/2024**.

BANCA EXAMINADORA

Prof. Dr. Ajalmar Rêgo da Rocha Neto (Orientador)

Instituto Federal de Educação, Ciência e Tecnologia do Ceará - PPGCC/IFCE

Prof. Dr. Josias Guimarães Batista

Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE

Prof. Dr. Guilherme de Alencar Barreto

Universidade Federal do Ceará – UFC



Documento assinado eletronicamente por **Ajalmar Rego da Rocha Neto, Professor do Ensino Básico, Técnico e Tecnológico**, em 16/09/2024, às 15:50, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Josias Guimaraes Batista, Professor do Ensino Básico, Técnico e Tecnológico**, em 16/09/2024, às 16:05, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Guilherme de Alencar Barreto, Usuário Externo**, em 16/09/2024, às 18:29, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade do documento pode ser conferida no site https://sei.ifce.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 informando o código verificador **6497111** e o código CRC **02C5EBD4**.

Dedicado à minha família, em especial minha mãe.

Agradecimentos

Gostaria de expressar minha profunda gratidão a todos que estiveram ao meu lado durante esta jornada acadêmica.

Ao meu orientador, Ajalmar, agradeço imensamente pelo apoio e orientação. Suas valiosas contribuições foram essenciais para a realização deste trabalho, e sou profundamente grato por todo o esforço dedicado.

Ao meu coorientador, Thiago, meu sincero agradecimento por sua colaboração e pelas importantes sugestões que enriqueceram significativamente este projeto. Sua expertise e disponibilidade foram fundamentais para o desenvolvimento do trabalho.

Ao professor Amaury, que sempre esteve disposto a ouvir minhas ideias e a me ajudar a organizá-las, expresso minha mais profunda gratidão. Sua disposição em compartilhar conhecimentos e orientar-me com paciência e compreensão fez toda a diferença no meu percurso acadêmico.

Ao meu amigo e colega de curso, Reginaldo, agradeço pela parceria, pelo apoio constante e pelas discussões construtivas que contribuíram para o aprimoramento deste trabalho. Sua camaradagem tornou esta jornada muito mais agradável e produtiva.

Ao meu amigo Mateus, expresso meu agradecimento pelo apoio e pela amizade que foram importantes para me manter motivado durante o percurso acadêmico.

Ao meu primo Marcelo, minha gratidão por ter me incentivado ao amor pela leitura através das suas recomendações dos grandes clássicos. Suas indicações literárias ampliaram meus horizontes e me proporcionaram uma base cultural sólida e enriquecedora.

Aos professores e amigos Carlos Henrique e Eurípedes, meus sinceros agradecimentos pelo incentivo ao estudo da matemática. Suas orientações e palavras de motivação foram cruciais para o desenvolvimento do meu interesse e entendimento na área.

E, em especial, à minha mãe, que sempre esteve ao meu lado. Seu amor, apoio incondicional e incentivo constante foram minha maior fonte de força e motivação. Sem você, nada disso seria possível. Agradeço de coração por acreditar em mim e estar presente em todos os momentos. Ao meu irmão Francisco, meu amor e minha gratidão.

Em especial ao meu irmão Alberto, que infelizmente não pôde presenciar a realização deste sonho, minha gratidão eterna. Sua presença em meus pensamentos e lembranças me inspirou ao longo de toda essa jornada.

Não poderia deixar de agradecer aos colegas de turma do PPGCC, por toda a sua experiência e conhecimento compartilhados. Também gostaria de agradecer aos

professores. Para finalizar, gostaria de agradecer o apoio financeiro da Coordenadoria de Aperfeiçoamento Pessoal (CAPES), que disponibilizou uma bolsa de estudos, juntamente com o PPGCC

"Se um problema não pode ser formulado, não se pode resolver."
(David Hilbert)

Resumo

Resumo: Neste trabalho, são propostos dois algoritmos de treinamento para Máquinas de Vetores-Suporte (SVM). O primeiro algoritmo baseia-se na função de decisão das SVMs, que é uma medida de distância, e inclui um hiperparâmetro referente à porcentagem de vetores de suporte. O segundo algoritmo é uma versão atualizada, na qual esse hiperparâmetro não é mais necessário. Ambos os algoritmos são comparados com a Sequential Minimal Optimization (SMO), o Kernel Adatron e os solucionadores clássicos de otimização quadrática para SVM (QP) em termos de acurácia, precisão, recall e tempo de treinamento para vários conjuntos de dados. Em geral, os resultados de acurácia são equivalentes; no entanto, nossa proposta apresenta uma vantagem significativa em termos de tempo computacional, sendo mais rápida que o SMO, o Kernel Adatron e o SVM (QP) no treinamento do modelo em todos os conjuntos de dados disponíveis.

Palavras-chaves: Aprendizado de máquina; máquinas de vetores suporte; classificação; otimização sequencial mínima; algoritmos de treinamento.

Abstract

In this work, we propose two training algorithms for Support Vector Machines (SVMs). The first algorithm is based on the SVM decision function, which is a kind of distance measure, and includes a hyperparameter referring to the percentage of support vectors. The second algorithm is an updated version, in which this hyperparameter is no longer necessary. Both algorithms are compared with Sequential Minimal Optimization (SMO), the Adatron Kernel, and classical quadratic optimization solvers for SVM (QP) in terms of accuracy, precision, recall, and training time for several datasets. In general, the accuracy results are equivalent; however, our proposal presents a significant advantage in terms of computational time, being faster than SMO, the Adatron Kernel, and SVM (QP) in training the model on all available datasets.

Keywords: Machine learning; support machines; classification; minimal sequential optimization; training algorithms.

Lista de ilustrações

Figura 1 – SVM com Kernel RBF	50
Figura 2 – Gráfico $dmSVM$	52
Figura 3 – Passos 1 e 2 do ajuste do d -SVM para dados lineares.	62
Figura 4 – Passos 3 e 4 do ajuste do d -SVM para dados lineares.	63
Figura 5 – Passos 5 e 6 do ajuste do d -SVM para dados lineares.	63
Figura 6 – Passos 7 e 8 do ajuste do d -SVM para dados lineares.	63
Figura 7 – Primeiro e segundo passos do ajuste do d -SVM.	64
Figura 8 – Terceiro e quarto passos do ajuste do d -SVM.	64
Figura 9 – Quinto e sexto passos do ajuste do d -SVM.	64
Figura 10 – Sétimo e oitavo passos do ajuste do d -SVM.	64
Figura 11 – Nono e décimo passos do ajuste do d -SVM.	65
Figura 12 – Visualização do modelo d -SVM após ajuste dos parâmetros	65
Figura 13 – Visualização do modelo dm -SVM após ajuste dos parâmetros	66

Lista de tabelas

Tabela 1	– Conjuntos de dados utilizados nas simulações computacionais.	69
Tabela 2	– Resultados para <i>kernel</i> Linear.	70
Tabela 3	– Resultados para <i>kernel</i> Polinomial.	71
Tabela 4	– Resultados para <i>kernel</i> RBF.	72
Tabela 5	– Comparação das métricas entre d-SVM, SMO, Adatron e SVMQP . . .	73
Tabela 6	– Comparação das métricas entre d-SVM, SMO, Adatron e SVMQP . . .	74
Tabela 7	– Comparação das métricas entre d-SVM, SMO, Adatron e SVMQP . . .	74
Tabela 8	– Comparação das métricas entre dm-SVM, SMO, Adatron e SVMQP .	75
Tabela 9	– Comparação das métricas entre dm-SVM, SMO, Adatron e SVMQP .	76
Tabela 10	– Comparação das métricas entre dm-SVM, SMO, Adatron e SVMQP .	76

List of Algorithms

1	Kernel Adatron para SVM	47
2	SVM	49
3	Sequential Minimal Optimization (SMO) com Kernel	53
4	d -SVM	58
5	dm -SVM	62

Lista de abreviaturas e siglas

SVM	Máquinas de Vetores Suporte
SMO	Sequencial Minimal Optimization
KKT	Karush-Kunh-Tucker
RBF	Função de Base Radial
BCW	Breast Cancer Wisconsin
HEA	Heart Disease
IFD	Iris Flower Dataset
SON	Sonar
VCP	Vertebra Column Pat
WIN	Wine
ART	Artificial

Lista de símbolos

x	Vetor de atributos
y	Rótulo
(x, y)	Padrão, instância ou amostra
n	Tamanho do conjunto de dados
$\mathbb{R}$	Conjunto dos números Reais
p	Dimensão da amostra
$g(\cdot)$	Função discriminante
$f(\cdot)$	Função
w	Vetor de pesos
b_0	Viés
$\mathbb{N}$	Conjunto dos números Naturais
h	Hiperplano ótimo
δ	Margem geométrica
$g_{\min}$	Valor mínimo da função discriminante
$g_{\max}$	Valor máximo da função discriminante
r_i	Distância associada ao ponto i
r_j	Distância associada ao ponto j
α_i	Multiplicadores de Lagrange
$\ \mathbf{w}\ ^2$	Norma ao quadrado do vetor de pesos
$\sum_{i=1}^n$	Soma sobre todos os exemplos i de 1 até n
y_i	Rótulo da i -ésima amostra
$\mathbf{x}_i$	Vetor de atributos da i -ésima amostra
$\mathbf{w} \cdot \mathbf{x}_i$	Produto escalar entre $\mathbf{w}$ e $\mathbf{x}_i$
$L(\mathbf{w}, b, \alpha)$	Função de Lagrange

Sumário

1	INTRODUÇÃO	29
1.1	Descrição do problema	30
1.2	Objetivos Geral e Específicos	32
1.3	Produção científica	32
1.4	Estrutura da Dissertação	32
2	FUNDAMENTAÇÃO TEÓRICA	35
2.1	Introdução ao Aprendizado Estatística e SVM	35
2.1.1	Funcional Risco e Funções de Perda	35
2.1.2	SVM com Margem Rígida	37
2.2	Condições de Karush-Kuhn-Tucker (KKT)	37
2.2.1	Condições Primal, Dual e de Estacionariedade	38
2.2.2	Classificador SVM com Margem Rígida	38
2.3	SVM com Margem Flexível	40
2.3.1	Formulação do Problema de Otimização	41
2.3.2	Função Lagrangeana	42
2.4	Métodos de Treinamento	44
2.4.1	Programação Quadrática	44
2.4.2	Aplicando ao SVM	45
2.4.3	Kernel Adatron	45
2.4.4	Kernel Adatron em SVMs	45
2.4.5	Processo do modelo Kernel Adatron	45
2.4.6	Desafios e Considerações	46
2.4.7	Kernels	47
2.5	Sequencial minimal optimization (SMO)	50
2.5.1	Primeira e segunda Heurística	51
2.5.2	O Hiperplano no SMO e os Vetores de Suporte	52
2.6	Conclusão	53
3	NOVOS ALGORITMOS DE TREINAMENTO PARA SVM	55
3.1	Algoritmo de treinamento com base na distância	55
3.1.1	Ajuste dos multiplicadores de Lagrange	55
3.1.2	Regra de Ajustes dos Multiplicadores de Lagrange	56
3.2	Segunda Proposta: Algoritmo de treinamento com base na distância média	58
3.2.1	Regra para Ajuste dos Multiplicadores de Lagrange	59

3.2.2	Parte Dois do Ajuste	60
3.3	Análise gráfica das etapas do d-SVM	62
3.3.1	Visualização dos modelos d -SVM e dm -SVM após ajuste dos parâmetros utilizando kernels para dados artificiais.	65
3.4	Conclusão	67
4	RESULTADOS E DISCUSSÕES	69
4.1	Hiperparâmetros e seus respectivos valores testados	69
4.2	Análise e Desempenho	73
4.2.1	Kernel Linear	73
4.2.2	Kernel Polinomial	73
4.2.3	Kernel RBF	74
4.3	Análise da segunda Proposta	75
4.3.1	Kernel Linear	75
4.3.2	Kernel Polinomial	75
4.3.3	Kernel RBF	76
4.4	Conclusão	77
5	CONCLUSÕES E TRABALHOS FUTUROS	79
	REFERÊNCIAS	81
	APÊNDICE A – ESPAÇOS DE HILBERT	83
A.1	Propriedade de Continuidade Uniforme	84
A.2	Aplicação da Continuidade Uniforme	84
A.3	Espaços Lineares e Espaços com Produto Interno	85
A.4	Produto Interno em Espaços Lineares	85
A.5	Kernels em Espaços de Hilbert	86
A.6	Kernel Linear	86
A.7	Kernel Polinomial	86
A.8	Kernel RBF (Radial Basis Function)	86
	APÊNDICE B – APÊNDICE B	87
B.1	Acurácia, Precisão e Recall	87
B.2	Demonstração do Teste de Friedman	89
B.3	Definições e Notações	89
B.4	Passos do Teste de Friedman	89
B.5	Calculando os Postos	89
B.6	Soma dos Postos por Tratamento	89
B.7	Estatística de Teste de Friedman	90

B.8	Distribuição da Estatística de Teste	90
B.9	Hipóteses	90
B.10	Considerações	90

1 Introdução

O aprendizado de máquina (machine learning) é um subcampo vital da inteligência artificial que se concentra na criação de modelos que permitem às máquinas aprenderem a partir de dados (SHALEV-SHWARTZ; BEN-DAVID, 2014). Diferente de simplesmente seguir instruções pré-programadas, os sistemas de aprendizado de máquina identificam padrões e fazem previsões com base em dados históricos. Existem três tipos principais de aprendizado de máquina: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço.

No aprendizado supervisionado, o modelo é treinado utilizando um conjunto de dados rotulados, onde cada exemplo de treinamento inclui uma entrada e a saída correspondente. O objetivo é aprender a mapear as entradas para as saídas corretas (RUSSELL; NORVIG, 2009). Já no aprendizado não supervisionado, o modelo trabalha com dados não rotulados, identificando padrões e estruturas subjacentes sem orientação prévia (LECUN; BENGIO; HINTON, 2015). No aprendizado por reforço, o modelo aprende a tomar decisões interagindo com um ambiente, recebendo feedback em forma de recompensas ou punições para otimizar suas ações ao longo do tempo (RUSSELL; NORVIG, 2009).

Classificação é um problema de aprendizado de máquina cujo objetivo é atribuir uma classe ou categoria a um conjunto de dados (RUSSELL; NORVIG, 2009). Em contraste, a regressão envolve prever um valor contínuo com base em dados de entrada, ajustando funções apropriadas. Existem diversos métodos de classificação, incluindo abordagens baseadas em redes neurais, como o perceptron multicamadas e redes RBF, que modelam relações complexas entre os dados. A classificação é uma subcategoria essencial do aprendizado supervisionado, focada na previsão de rótulos de classe categóricos para novas instâncias ou pontos de dados, com base em observações passadas. Esses rótulos de classe são valores discretos e não sequenciais, identificando a classe à qual os dados pertencem (BISHOP; NASRABADI, 2006). Um exemplo comum dessa aplicação é a identificação de spam em e-mails, onde o modelo de aprendizado de máquina desenvolve um conjunto de regras para distinguir entre duas classes possíveis: spam ou não spam (SHALEV-SHWARTZ; BEN-DAVID, 2014). Criação de um modelo capaz de identificar padrões em dados rotulados e aplicar esses padrões a novos dados, possibilitando uma classificação precisa e eficiente. Com a compreensão e aplicação dessas técnicas, é viável desenvolver sistemas que automatizam a categorização de dados em diversos contextos.

As máquinas de vetores de suporte (SVMs) ferramentas altamente eficazes no reconhecimento de padrões, encontrando sucesso em diversas aplicações graças a sua sólida base teórica (Rocha Neto, 2011), combinando princípios estatísticos com a minimização de

riscos estruturais, permitindo enfrentar até mesmo problemas complexos de reconhecimento de padrões não lineares.

Elas se destacam por sua capacidade de generalização de alto nível, o que significa que conseguem aplicar o conhecimento adquirido durante o treinamento para fazer previsões precisas em novos dados. Além disso, elas têm poucos parâmetros que precisam ser ajustados, o que simplifica o processo de configuração (Rocha Neto, 2011). Uma das vantagens mais notáveis dos classificadores SVM é sua habilidade em lidar com problemas que envolvem padrões não lineares (Dias e Neto, 2017). Isso significa que eles podem capturar relações entre os dados que não podem ser descritas por uma simples linha reta ou hiperplano. A capacidade de lidar com dados mais complexos é o que os torna tão versáteis e eficientes em várias situações.

Outro benefício importante é que as máquinas de vetores de suporte (SVMs) podem funcionar bem mesmo com uma quantidade relativamente menor de exemplos de treinamento. Isso é particularmente útil quando o acesso aos dados é limitado ou quando a coleta de dados é dispendiosa. Em resumo, a combinação de sua capacidade de generalização, configuração simplificada e exigências de amostragem reduzidas torna os SVMs uma escolha popular em várias áreas de aprendizado de máquina (Bishop e Nasrabadi, 2006).

Devido a essas vantagens notáveis, os SVMs encontraram aplicação em uma ampla gama de campos no aprendizado de máquina (Dias e Neto, 2017). Sua eficácia e adaptabilidade os tornaram uma escolha confiável para problemas complexos que envolvem padrões e relacionamentos intrincados nos dados.

No entanto, a eficácia das máquinas de vetores de suporte (SVMs) depende de vários métodos de otimização, que é uma etapa crucial para uma classificação precisa. A grande questão é que muitos dos métodos de otimização requerem um grande custo computacional (Ji et al., 2021). A necessidade de desenvolver mais rápidos e capazes de lidar com grandes conjuntos de dados é cada vez mais crucial à medida que a quantidade de informações disponíveis cresce exponencialmente em diversas áreas. À medida que os dados se tornam cada vez mais volumosos, os modelos de aprendizado de máquina, como máquinas de vetores de suporte (SVMs), enfrentam desafios significativos de escalabilidade e eficiência. mais rápidos e eficientes são essenciais para permitir que máquinas de vetores de suporte (SVMs) seja aplicado em tempo real ou em tarefas que requerem processamento em larga escala.

1.1 Descrição do problema

A máquina de vetores de suporte (SVMs) é um método de aprendizado de máquina supervisionado que tem sido aplicado tanto em tarefas de classificação quanto de regressão.

Em tarefas de classificação, o processo de treinamento da SVM encontra um hiperplano ótimo que separa padrões de duas classes diferentes. Tal hiperplano ótimo é alcançado pela aplicação de um modelo de treinamento, no qual os vetores de suporte e o viés são encontrados. Os vetores de suporte são aqueles padrões que, após o processo de treinamento, possuem multiplicadores de Lagrange não nulos (VAPNIK, 1995).

O treinamento de uma Máquina de Vetores de Suporte envolve a resolução de um problema de otimização quadrática, tipicamente definido por uma função objetivo quadrática com restrições. Modelos de treinamento alternativos incluem o método *chunking* (CRISTIANINI; SHAW-ET AL., 2000), que utiliza técnicas de gradiente conjugado projetado, o método de (OSUNA; FREUND; GIROSI, 1997) e o bem conhecido método sequential minimal optimization (SMO) (PLATT, 1998a). O problema com métodos clássicos de otimização para realizar o treinamento de máquinas de vetores de suporte (SVMs) em grandes conjuntos de dados reside no fato de que envolve lidar com uma grande matriz quadrada¹ na função objetivo do problema de otimização quadrática a ser resolvido (LI; WANG; HU, 2009). Enquanto os métodos *chunking* e de Osuna tentam mitigar essa desvantagem utilizando matrizes menores, estas ainda podem ser suficientemente grandes para tornar a resolução impraticável via métodos clássicos de otimização quadrática. Métodos alternativos de treinamento utilizam modelos genéticos (JI et al., 2021) ou *simulated annealing* (DIAS; NETO, 2017); no entanto, em alguns métodos, as restrições do problema de otimização são violadas.

O modelo SMO, por outro lado, reduz o problema de programação quadrática em uma série de problemas menores que são resolvidos analiticamente (PLATT, 1998b). Entretanto, mesmo com essa vantagem, ainda é computacionalmente lento utilizar o SMO em conjuntos de dados muito grandes, pois pode levar um tempo considerável para que o modelo convirja ao seu ponto de parada.

Uma forma mais geral de lidar com esse tipo de problema envolve, por exemplo, o uso de técnicas de seleção de padrões ou quantização vetorial para reduzir o tamanho do conjunto de treinamento (CHEN; CHENG; XU, 2008; NALEPA; KAWULOK, 2019) (DIAS; NETO, 2017). Após esse processo, pode-se utilizar métodos como o SMO ou outros clássicos mencionados para realizar o treinamento (PLATT, 1998b). No entanto, ao aplicar uma técnica de quantização vetorial, por exemplo, o conjunto de dados resultante é considerado uma representação do conjunto de dados original, e geralmente essa representação ocorre com perdas e, como tal, sua qualidade na representação depende da capacidade do método aplicado. Nesse contexto, pode-se perceber que ainda é interessante encontrar métodos que sejam eficientes para lidar com grandes conjuntos de dados.

¹ Essa matriz é quadrada de tamanho N , onde N é o número de padrões de treinamento.

1.2 Objetivos Geral e Específicos

O objetivo geral deste trabalho é propor algoritmos de treinamento para máquinas de vetores de suporte (SVMs) e investigar métodos alternativos que sejam rápidos para a obtenção dos multiplicadores de Lagrange em tais .

Mais especificamente, tem-se os seguintes objetivos particulares:

- **Proposição de algoritmos:** Propor e desenvolver algoritmos , utilizando abordagens que sejam capazes de melhorar o desempenho em termos de tempo e eficiência.
- **Avaliação de Desempenho:** Realizar uma avaliação detalhada do desempenho do método proposto, considerando métricas como acurácia, precisão, recall, F1-score e eficiência computacional. Comparar os resultados obtidos com aqueles de abordagens de referência.
- **Análise de Contribuições:** Analisar e discutir as contribuições dos métodos propostos em relação à melhoria do desempenho dos de aprendizado de máquina, sua adaptabilidade a diferentes tipos de problemas e sua escalabilidade para conjuntos de dados.

1.3 Produção científica

Durante este projeto de mestrado, os seguintes trabalhos científicos foram aceitos e publicados, a saber:

- A New Training Algorithm for Support Vector Machines, Acelio Sousa, Thiago Alves Rocha, Ajalmar Rêgo da Rocha Neto. Aceito para o *International Conference on Hybrid Artificial Intelligence Systems (HAIS 2024)*.
- Um novo método para treinamento de vetores de suporte, Acelio Sousa, Thiago Alves Rocha, Ajalmar Rêgo da Rocha Neto. Aceito para o *XXV Congresso Brasileiro de Automática (CBA 2024)*.

1.4 Estrutura da Dissertação

Esta dissertação possui uma estrutura de cinco capítulos. O capítulo inicial apresenta a introdução e contextualização do tema. Os capítulos subsequentes abordam os seguintes tópicos: e contextualização do tema. Os capítulos subsequentes abordam os seguintes tópicos:

Capítulo 2: Fundamentação Teórica: Máquinas de Vetores de Suporte (SVMs).Detalha o estado da arte das SVMs, incluindo:

- Conceitos básicos e funcionamento das SVMs
- Métodos de treinamento
- A função dos kernels na construção de espaços de características de alta dimensão.

Capítulo 3: Modelos Proposto. Apresenta os modelos desenvolvidos na pesquisa.

- Explica a fundamentação matemática de cada modelo.
- Descreve cada modelo, incluindo os princípios teóricos e as etapas de implementação.

Capítulo 4: Metodologia e Resultados. Detalha a metodologia utilizada em simulações computacionais.

- Apresenta os resultados das simulações, incluindo tabelas com métricas de desempenho e gráficos comparativos.
- Discute os resultados para validar as contribuições dos modelos propostos.

Capítulo 5: Considerações Finais. Oferece uma conclusão geral da pesquisa.

- Destaca as principais contribuições do trabalho.
- Sugere planos para trabalhos futuros, caminhos potenciais para aprofundar e expandir a pesquisa.

2 Fundamentação Téorica

Em 1992, Vapnik e colaboradores apresentaram os primeiros trabalhos relacionados as máquinas de vetores de suporte (SVMs) para problemas de classificação (BISHOP; NASRABADI, 2006). Nessa época, eles introduziram um novo modelo de treinamento que tinha como objetivo maximizar a margem de separação entre os vetores de treinamento e um hiperplano das máquinas de vetores de suporte (SVMs). Essa abordagem permitiu a obtenção de classificadores de margem ótima para perceptrons, funções de base radial (RBFs) e polinomiais. Além disso, os SVMs permitiam determinar quais vetores de treinamento seriam os mais relevantes para definir o hiperplano de separação entre as classes, sendo esses conhecidos como vetores de suporte. Essa descoberta abriu caminho para o desenvolvimento de um dos modelos mais poderosos e versáteis em aprendizado de máquina, que tem sido amplamente aplicado em problemas de classificação, regressão e outros domínios de análise de dados (ROCHA NETO, 2011).

2.1 Introdução ao Aprendizado Estatística e SVM

O aprendizado estatístico é uma abordagem fundamental para desenvolver modelos de aprendizado de máquina capazes de generalizar a partir de dados observados. As máquinas de vetores de suporte (SVMs) são uma classe proeminente de modelos de aprendizado que se baseiam em princípios estatísticos sólidos para construir preditivos robustos (VAPNIK, 1995). Nesta introdução, exploraremos conceitos-chave do aprendizado estatístico em relação às SVMs, incluindo o risco esperado, a minimização do risco empírico, a dimensão VC e o princípio da minimização do risco estrutural.

2.1.1 Funcional Risco e Funções de Perda

Dado um conjunto em $\mathbb{R}^m$ composto de todas as funções $f(x, w)$ com parâmetros $w \in \mathbb{R}^n$. De forma geral, considere a saída de $f(x, w)$ como escalar. Considere ainda um conjunto de treinamento com p amostras, denotado por $\{(x_i, y_i)\}_{i=1}^p$. Assume-se que as amostras são independentes, igualmente distribuídas e que sua função de densidade de probabilidade $p(x)$ é desconhecida. Suponha agora que uma medida de perda possa ser dada pela função genérica $L(y, f(x, \mathbf{w}))$. Funções de perda comumente empregadas são a função de perda quadrática:

$$L(y, f(x, w)) = \frac{1}{2}(y - f(x, w))^2 \quad (2.1)$$

e a função de perda absoluta:

$$L(y, f(x, w)) = |y - f(x, w)| \quad (2.2)$$

Embora saídas contínuas sejam frequentemente associadas a problemas de regressão, elas também podem ser adaptadas para tarefas de classificação. Nos problemas de classificação onde temos duas classes, a função de perda é dada normalmente por:

$$L(y, f(x, w)) = \begin{cases} 0 & \text{se } y = f(x, w) \\ 1 & \text{se } y \neq f(x, w) \end{cases} \quad (2.3)$$

Estas funções de perda podem ser utilizadas para gerar um *funcional risco*, dado por:

$$R(f) = \mathbb{E}_{p(x,y)}[L(y, f(x, \cdot))] \quad (2.4)$$

No entanto, como a função de densidade de probabilidade $p(x, y)$ é desconhecida, o risco funcional não pode ser calculado diretamente. Para lidar com isso, uma aproximação chamada de *risco empírico* é utilizada. O risco empírico é a média das perdas calculadas sobre o conjunto de treinamento e é dado por:

$$R_{\text{emp}}(f) = \frac{1}{p} \sum_{i=1}^p L(y_i, f(x_i, w)) \quad (2.5)$$

O objetivo da aprendizagem é então minimizar este risco empírico sobre o conjunto de funções $f(x, w)$, na esperança de que a minimização do risco empírico leve à minimização do risco funcional verdadeiro.

Entretanto, a minimização do risco empírico não garante que o modelo generalizará bem para dados não observados. Para analisar a capacidade de generalização de um modelo, introduzimos o conceito de *dimensão VC* (Vapnik-Chervonenkis). A dimensão VC é uma medida da capacidade de um modelo de aprendizado, definida como o maior número de pontos que pode ser *shattered* (completamente separado) pelo modelo.

Shattering ocorre quando, para um conjunto de pontos, o modelo é capaz de classificar todas as possíveis combinações de rótulos para esses pontos. Um modelo com uma dimensão VC alta tem maior capacidade de modelagem, o que pode levar à *overfitting*, enquanto um modelo com uma dimensão VC baixa pode ser subcapaz de capturar a complexidade dos dados.

A dimensão VC é relevante porque permite estabelecer limites sobre a diferença entre o risco empírico $R_{\text{emp}}(f)$ e o risco verdadeiro $R(f)$. Quanto menor a dimensão VC em relação ao número de amostras, mais provável que o risco empírico seja uma boa aproximação do risco verdadeiro. Este fato é fundamental na teoria do aprendizado estatístico, onde se busca o equilíbrio entre a capacidade do modelo e a generalização.

Neste contexto, as Máquinas de Vetores de Suporte (SVMs) se destacam como um dos modelos que equilibram a capacidade de generalização e a minimização do risco empírico. As SVMs aplicam o conceito de margem máxima, utilizando funções de perda como a *hinge loss* para garantir que a margem entre as classes seja maximizada. Isso

está diretamente relacionado à ideia de minimizar o risco empírico enquanto se controla a dimensão VC, o que ajuda a garantir uma boa generalização.

As Máquinas de Vetores de Suporte (SVMs) são modelos de aprendizado supervisionado aplicáveis tanto à classificação de padrões quanto à aproximação de funções (ROCHA NETO, 2011). Elas se destacam pela capacidade de criar hiperplanos de separação otimizados entre classes distintas. No contexto da classificação de padrões, o objetivo principal das SVMs é encontrar um hiperplano que maximize a distância entre os padrões das classes positiva e negativa.

2.1.2 SVM com Margem Rígida

A formulação matemática do SVM para dados linearmente separáveis pode ser expressa da seguinte forma: Dado um conjunto de dados de treinamento $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, onde $\mathbf{x}_i$ é um vetor de características $\mathbf{x}_i \in \mathbb{R}^n$ e $y_i \in \{+1, -1\}$ é a classe associada, o objetivo é encontrar o vetor de pesos $\mathbf{w}$ e o termo de *bias* b que definem o hiperplano de separação $\mathbf{w} \cdot \mathbf{x} + b = 0$. A distância d entre um ponto $\mathbf{x}$ e o hiperplano é dada por:

$$d = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|}$$

As condições para a classificação dos pontos em SVM podem ser expressas como:

$$\text{Se } y_i = +1, \quad \mathbf{w} \cdot \mathbf{x}_i + b \geq 1,$$

$$\text{Se } y_i = -1, \quad \mathbf{w} \cdot \mathbf{x}_i + b \leq -1.$$

A margem entre os dois hiperplanos é:

$$\text{Margem} = \frac{2}{\|\mathbf{w}\|}$$

O problema de otimização do SVM, que busca maximizar a margem, pode ser formulado como:

$$\text{Minimizar} \quad \frac{1}{2} \|\mathbf{w}\|^2$$

Sujeito a:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \text{para } i = 1, 2, \dots, n.$$

Essa formulação garante a maximização da margem entre as classes, o que é equivalente a minimizar a norma do vetor $\mathbf{w}$.

2.2 Condições de Karush-Kuhn-Tucker (KKT)

As condições de Karush-Kuhn-Tucker (KKT) são um conjunto de condições necessárias para otimização em problemas de programação matemática com restrições, incluindo

a otimização Lagrangeana. Essas condições desempenham um papel crucial na resolução eficaz de problemas de otimização com restrições, pois fornecem critérios para determinar quando um conjunto de valores é uma solução viável e ótima.

Problema de Otimização Genérico com Restrições

Consideremos um problema de otimização genérico com restrições:

$$\text{Min } \mathbf{f}(\mathbf{x}) \quad (2.6)$$

$$\text{sujeito a } \mathbf{g}_i(\mathbf{x}) \leq \mathbf{0}, \quad \text{para } i = 1, 2, \dots, m \quad (2.7)$$

$$\mathbf{h}_j(\mathbf{x}) = \mathbf{0}, \quad \text{para } j = 1, 2, \dots, p \quad (2.8)$$

Onde:

- $f(x)$ é a função objetivo que queremos minimizar.
- $g_i(x)$ são as restrições de desigualdade.
- $h_j(x)$ são as restrições de igualdade.
- x é o vetor de variáveis de otimização.

2.2.1 Condições Primal, Dual e de Estacionariedade

Condição de Primalidade

As derivadas das restrições de desigualdade são não-positivas:

$$\mathbf{g}_i(\mathbf{x}^*) \leq \mathbf{0}, \quad \text{para } i = 1, 2, \dots, m$$

Condição de Dualidade

As variáveis de multiplicador de Lagrange associadas às restrições de desigualdade são não-negativas:

$$\lambda_i \geq \mathbf{0}, \quad \text{para } i = 1, 2, \dots, m$$

2.2.2 Classificador SVM com Margem Rígida

O problema dual do classificador SVM envolve a obtenção dos multiplicadores de Lagrange α_i associados às restrições do problema de otimização (ROCHA NETO, 2011). Lagrangeano para o problema é dado por:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \quad (2.9)$$

O problema dual é dado por:

$$\text{Maximizar } D(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (2.10)$$

sujeito a:

$$\alpha_i \geq 0 \quad \forall i \quad (2.11)$$

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (2.12)$$

A solução do problema dual fornece os valores ótimos de α , e o vetor de pesos pode ser obtido como:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \quad (2.13)$$

Uma vez que os multiplicadores de Lagrange são encontrados, os vetores de suporte são aqueles com $\alpha_i > 0$, e o termo de viés b pode ser calculado usando um dos vetores de suporte. O problema dual do classificador SVM envolve a obtenção dos multiplicadores de Lagrange α_i associados às restrições do problema de otimização (ROCHA NETO, 2011). O Lagrangeano para o problema é dado por:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \quad (2.14)$$

A solução é determinada pelo ponto de sela da função Lagrangeana $L(\mathbf{w}, b, \alpha)$, que deve ser minimizada em relação a $\mathbf{w}$ e b e maximizada em relação a α . Portanto, ao diferenciar em relação a $\mathbf{w}$ e b e igualar a zero, obtemos as seguintes condições de otimização:

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0, \quad (2.15)$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0 \quad (2.16)$$

O que resulta em:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i, \quad (2.17)$$

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (2.18)$$

É importante observar que o terceiro termo $(-b \sum_{i=1}^n \alpha_i y_i)$ da forma expandida da função Lagrangeana apresentada na equação é zero. Assim, podemos reescrever:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{w}^T \mathbf{x}_i + \sum_{i=1}^n \alpha_i \quad (2.19)$$

Continuando o processo de resolução da função Lagrangeana, podemos notar que o segundo termo $(-\sum_{i=1}^n \alpha_i y_i \mathbf{w}^T \mathbf{x}_i)$, quando aplicamos o resultado obtido na equação, equivale a $-\mathbf{w}^T \mathbf{w}$. Portanto, podemos mostrar que:

$$L(\mathbf{w}, b, \alpha) = -\frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \alpha_i \quad (2.20)$$

Ou ainda, substituindo a equação acima, obtemos:

$$L(\mathbf{w}, b, \alpha) = -\frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \alpha_i \quad (2.21)$$

No processo de continuação da resolução da função Lagrangeana, chegamos a:

$$L(\alpha) = -\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^n \alpha_i \quad (2.22)$$

Ou ainda:

$$L(\alpha) = \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad (2.23)$$

O problema de otimização dual é formulado como:

$$\max L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad (2.24)$$

sujeito a:

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (2.25)$$

$$\alpha_i \geq 0 \quad \text{para } i = 1, \dots, n \quad (2.26)$$

No processo de resolução do problema dual, são obtidos os multiplicadores de Lagrange ótimos $\{\alpha_i^o\}_{i=1}^n$. Em seguida, o vetor de pesos $\mathbf{w}^o$ e o viés b^o podem ser computados por:

$$\mathbf{w}^o = \sum_{i=1}^n \alpha_i^o y_i \mathbf{x}_i \quad (2.27)$$

Podemos encontrar b da seguinte forma:

$$b^o = \frac{1}{n} \sum_i (y_i - \mathbf{w} \cdot \mathbf{x}_i) \quad (2.28)$$

Ou também como definido por (CRISTIANINI; SHAWE-TAYLOR, 2000),

$$b^o = -\frac{\max_{y_i=-1}(\mathbf{w} \cdot \mathbf{x}_i) + \min_{y_i=1}(\mathbf{w} \cdot \mathbf{x}_i)}{2} \quad (2.29)$$

A função discriminante, como definida no problema primal para o hiperplano ótimo, pode ser reescrita com base no problema dual, resultando em:

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i^o y_i \mathbf{x}_i^T \mathbf{x} + b^o \quad (2.30)$$

2.3 SVM com Margem Flexível

O classificador SVM com margem flexível busca encontrar um hiperplano que separe os dados em diferentes classes. Quando os dados são linearmente separáveis, o objetivo é maximizar a margem entre as classes. No entanto, quando os dados não são

perfeitamente separáveis, é necessário permitir alguns erros de classificação, o que leva ao conceito de "margem flexível".

Esse tipo de classificador permite que alguns pontos de dados fiquem dentro da margem ou até mesmo do lado errado do hiperplano. Isso torna o SVM mais flexível para lidar com dados complexos e ruidosos. O problema de otimização para o SVM de margem flexível pode ser formulado usando argumentos geométricos ou funções de perda, que quantificam os erros de classificação (DIAS; NETO, 2017).

Uma técnica comum para resolver esse problema é o uso de multiplicadores de Lagrange, que incorporam as restrições do problema de otimização (DIAS; NETO, 2017). Isso resulta em um problema de otimização dual, cujo objetivo é encontrar os vetores de suporte – os pontos de dados mais relevantes para a construção do hiperplano de separação (SHALEV-SHWARTZ; BEN-DAVID, 2014).

Geometricamente, o SVM de margem flexível pode ser visto como um hiperplano que separa os envólucros convexos correspondentes aos exemplos de dados positivos e negativos (ROCHA NETO, 2011). Essa interpretação fornece uma visão adicional sobre o funcionamento do SVM em cenários mais complexos.

Em resumo, o classificador SVM de margem flexível é uma extensão do SVM clássico que permite erros de classificação e lida com dados não linearmente separáveis. A abordagem para resolver o problema de otimização envolve a utilização de multiplicadores de Lagrange, resultando em um problema de otimização dual. Essa técnica é poderosa para lidar com problemas de classificação mais desafiadores, onde uma separação perfeita dos dados não é possível.

2.3.1 Formulação do Problema de Otimização

Queremos minimizar:

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad (2.31)$$

Sujeito a:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{para } i = 1, \dots, N \quad (2.32)$$

$$\xi_i \geq 0 \quad \text{para } i = 1, \dots, N \quad (2.33)$$

O parâmetro de regularização C controla o equilíbrio entre a busca por uma margem mais ampla e a minimização das violações da margem. Pequenos valores de C permitem mais violações de margem, enquanto valores maiores de C reduzem as violações de margem, tornando a superfície de decisão mais flexível.

As SVM com margem flexível permitem que amostras fiquem dentro da margem ou até mesmo do lado errado da margem, desde que essas violações sejam penalizadas na função de custo. Isso permite que o modelo seja mais robusto a dados ruidosos ou com

sobreposição entre classes. Vamos começar com a formulação do problema de otimização das SVM com margem flexível. Queremos encontrar $\mathbf{w}$, b e ξ_i que minimizem a função de custo:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad (2.34)$$

Sujeito a:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{para } i = 1, \dots, N \quad (2.35)$$

$$\xi_i \geq 0 \quad \text{para } i = 1, \dots, N \quad (2.36)$$

Essa função de custo combina a busca por uma margem mais ampla (primeiro termo) com a penalização das violações da margem (segundo termo). O parâmetro C controla o trade-off entre esses dois objetivos.

2.3.2 Função Lagrangeana

A função Lagrangeana para o classificador SVM com margem flexível é dada por:

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y_i(\mathbf{x}_i^T \mathbf{w} + b) - 1 + \xi_i] - \sum_{i=1}^n \beta_i \xi_i, \quad (2.37)$$

onde todos os elementos nos conjuntos $\alpha = \{\alpha_i\}_{i=1}^n$, $\beta = \{\beta_i\}_{i=1}^n$ e $\xi = \{\xi_i\}_{i=1}^n$ são valores não negativos. A forma expandida da função Lagrangeana é:

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T \mathbf{w} - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \alpha_i \xi_i - \sum_{i=1}^n \beta_i \xi_i. \quad (2.38)$$

Para resolver o problema de otimização, precisamos diferenciar a função de custo em relação a $\mathbf{w}$, b e ξ , e igualá-los a zero, resultando nas seguintes condições de otimização:

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial \mathbf{w}} = 0, \quad (2.39)$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial b} = 0, \quad (2.40)$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial \xi} = 0. \quad (2.41)$$

Resolver a primeira condição nos leva a:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i. \quad (2.42)$$

Resolver a segunda condição resulta em:

$$\sum_{i=1}^n \alpha_i y_i = 0. \quad (2.43)$$

E finalmente, ao resolver a terceira condição, obtemos:

$$C = \alpha_i + \beta_i. \quad (2.44)$$

Ao aplicar o resultado acima ao segundo termo $C \sum_{i=1}^n \xi_i$ na expressão, e reorganizar os termos $\sum_{i=1}^n \alpha_i \xi_i$ e $\sum_{i=1}^n \beta_i \xi_i$ como $\sum_{i=1}^n \xi_i (\alpha_i + \beta_i)$, podemos reescrever a expressão como:

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \xi_i (\alpha_i + \beta_i) - \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i^T \mathbf{w}) - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \xi_i (\alpha_i + \beta_i). \quad (2.45)$$

Podemos eliminar o segundo e o último termos dessa expressão, resultando em:

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i^T \mathbf{w}) - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i. \quad (2.46)$$

A partir daqui, o processo de resolução é similar ao utilizado para o classificador SVM com margem rígida. Note que o terceiro termo $-b \sum_{i=1}^n \alpha_i y_i$ na expressão é zero devido ao resultado anterior. Portanto, a expressão pode ser simplificada ainda mais como:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \mathbf{w}^T \left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \right) + \sum_{i=1}^n \alpha_i. \quad (2.47)$$

Reconhecendo que $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$, podemos simplificar ainda mais a expressão:

$$L(\mathbf{w}, b, \alpha) = -\frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \alpha_i. \quad (2.48)$$

Substituindo o valor de $\mathbf{w}$ na expressão, obtemos:

$$L(\alpha) = -\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^n \alpha_i. \quad (2.49)$$

Finalmente, chegamos à forma da função Lagrangeana em termos dos multiplicadores de Lagrange α_i :

$$L(\alpha) = -\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^n \alpha_i. \quad (2.50)$$

Este é o problema de otimização dual para o classificador SVM com margem flexível, onde o objetivo é maximizar a função $L(\alpha)$ sujeita às condições $\sum_{i=1}^n \alpha_i y_i = 0$ e $0 \leq \alpha_i \leq C$ para $i = 1, \dots, n$.

Ao analisar a expressão, observa-se uma única diferença em relação a um caso anterior: a restrição superior imposta aos multiplicadores de Lagrange, $0 \leq \alpha_i \leq C$. Todas as outras partes da expressão permanecem iguais ao que foi apresentado anteriormente. Essa distinção decorre da relação $\alpha_i + \beta_i = C$. Como $\beta_i \geq 0$, o menor valor possível para β_i é zero, o que ocorre quando $\alpha_i = C$.

Embora o uso da margem flexível no classificador SVM seja uma melhoria, há momentos em que o problema não pode ser resolvido satisfatoriamente, mesmo com a permissão de alguns erros de classificação. Uma ilustração desse tipo de desafio ocorre em situações em que as duas classes de um cenário de classificação binária não podem ser separadas de maneira linear usando uma única linha ou hiperplano (ROCHA NETO, 2011).

No entanto, existe uma solução para lidar com esse cenário. Os "kernels" podem ser empregados para transformar o espaço original dos dados em um espaço de dimensão superior, onde as classes podem ser separadas de forma mais clara (ROCHA NETO, 2011). Um "kernel" é uma função que calcula o produto interno entre dois pontos no espaço transformado sem a necessidade de efetivamente transformar todos os dados.

Usar um "kernel" permite que o SVM seja aplicado em um espaço de alta dimensão sem a complexidade computacional associada à transformação direta dos dados. Isso permite encontrar fronteiras de decisão mais complexas e flexíveis, que são capazes de lidar com problemas de classificação não lineares.

2.4 Métodos de Treinamento

A programação quadrática é uma ferramenta eficiente para encontrar soluções ótimas em problemas de otimização convexa, como o SVM (SHALEV-SHWARTZ; BEN-DAVID, 2014). Vamos considerar o caso do SVM com margem máxima, onde temos um conjunto de treinamento linearmente separável $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\}$, com $x_i \in \mathbb{R}^n$ sendo os vetores de características dos dados e $y_i \in \{-1, 1\}$ sendo as classes. O objetivo do SVM com margem máxima é encontrar o hiperplano de separação ótimo $w \cdot x = b$, onde $w \in \mathbb{R}^n$ é o vetor de pesos e b é o termo de viés.

Formulação do Problema

A função objetivo do SVM com margem máxima é definida como:

$$J(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2$$

A função objetivo é convexa, o que implica que podemos encontrar o mínimo global do problema.

2.4.1 Programação Quadrática

A abordagem para resolver o SVM com margem máxima é através da utilização de técnicas de otimização quadrática. O problema de otimização quadrática geralmente tem a forma:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\frac{1}{2} \mathbf{w}^T Q \mathbf{w} - \mathbf{q}^T \mathbf{w} \right) \quad (2.51)$$

sujeito às restrições:

$$X^T \mathbf{w} \geq \mathbf{c} \quad (2.52)$$

onde Q é uma matriz definida positiva, $\mathbf{q}$ é um vetor e X e $\mathbf{c}$ são matrizes e vetores que descrevem as restrições do problema.

2.4.2 Aplicando ao SVM

Para resolver o SVM com margem máxima usando programação quadrática, podemos definir Q como a matriz identidade I e $\mathbf{q}$ como o vetor nulo $\mathbf{0}$. Além disso, as restrições podem ser expressas como (ROCHA NETO, 2011).:

$$(y_i \mathbf{x}_i) \cdot \mathbf{w} \geq 1 + y_i b \quad \text{para } i = 1, \dots, l \quad (2.53)$$

Podemos construir a matriz X da seguinte forma:

$$X = \begin{pmatrix} y_1 \mathbf{x}_1^T \\ y_2 \mathbf{x}_2^T \\ \vdots \\ y_l \mathbf{x}_l^T \end{pmatrix} \quad (2.54)$$

e o vetor $\mathbf{c}$ como:

$$\mathbf{c} = \begin{pmatrix} 1 + y_1 b \\ 1 + y_2 b \\ \vdots \\ 1 + y_l b \end{pmatrix} \quad (2.55)$$

2.4.3 Kernel Adatron

O Kernel Adatron é um modelo adaptativo para treinar máquinas de vetores de suporte (SVMs), desenvolvido a partir do clássico modelo Adatron, que foi desenhado originalmente para treinar redes neurais tipo perceptron, ajustando os pesos até a convergência (FREUND; SCHAPIRE, 1996). A ideia por trás do Adatron é ajustar iterativamente os coeficientes para maximizar a margem do classificador.

2.4.4 Kernel Adatron em SVMs

O Kernel Adatron adapta o conceito do Adatron ao problema de otimização das SVMs, ajustando os multiplicadores de Lagrange (α) associados a cada vetor de suporte. Utiliza-se um kernel para projetar os dados num espaço de características de maior dimensão, esperando que as classes sejam linearmente separáveis (VAPNIK, 1995). O modelo ajusta iterativamente os coeficientes de Lagrange para melhor separar as classes neste espaço.

2.4.5 Processo do modelo Kernel Adatron

1. **Inicialização:** Os coeficientes de Lagrange (α) são inicializados, geralmente como zero ou um pequeno valor aleatório (DIAS; NETO, 2017).

2. **Atualização Iterativa:** Em cada iteração, os coeficientes de Lagrange são ajustados com a seguinte regra:

$$\alpha_i \leftarrow \alpha_i + \eta \cdot \left(1 - y_i \sum_j \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right)$$

onde η é a taxa de aprendizado, y_i são os rótulos de classe, $K(\mathbf{x}_i, \mathbf{x}_j)$ é a função *kernel* que mede a similaridade entre os vetores de características $\mathbf{x}_i$ e $\mathbf{x}_j$, e α_j são os multiplicadores de Lagrange atuais (BURGES, 1998).

3. **Condição de Parada:** O processo continua até que os coeficientes converjam ou até que um número máximo de iterações seja alcançado (PLATT, 1998a).
4. **Classificação:** Uma vez que os coeficientes de Lagrange estão determinados, o classificador pode ser construído e utilizado para fazer previsões.

Vantagens do Kernel Adatron

- **Eficiência Computacional:** O Kernel Adatron pode ser mais rápido que os métodos de otimização quadrática tradicionais, especialmente em grandes conjuntos de dados (KEERTHI et al., 2001).
- **Flexibilidade:** Permite o uso de diferentes funções de kernel, adaptando o modelo a diversos tipos de dados e características específicas dos problemas (SCHÖLKOPF; SMOLA, 2002).

2.4.6 Desafios e Considerações

- **Seleção do Kernel:** A escolha do kernel é crucial, pois afeta diretamente a capacidade do modelo de separar as classes no espaço projetado (DIAS; NETO, 2017).
- **Ajuste de Hiperparâmetros:** Parâmetros como a taxa de aprendizado (η) e o número máximo de iterações precisam ser cuidadosamente ajustados para evitar o sobreajuste e garantir a convergência do modelo (DIAS; NETO, 2017). A seguir temos o pseudocódigo

Algorithm 1 Kernel Adatron para SVM

```

1: Dado: Conjunto de treinamento  $(x_i, y_i)$ , onde  $x_i$  são vetores de características e  $y_i$  são os
   rótulos de classe
2: Dado: Taxa de aprendizado  $\eta$ , Máximo de iterações  $max\_iter$ , Tolerância  $\epsilon$ , Penalidade  $C$ 
3: Resultado: Vetor de pesos  $w$  e viés  $b$ 
4: Inicialização:
5: Inicialize os multiplicadores de Lagrange  $\alpha_i \leftarrow 0$  para cada  $i$ 
6: Defina a iteração  $\leftarrow 0$ 
7: while iterações  $< max\_iter$  do
8:   for cada amostra  $i$  do
9:     Calcule o gradiente:  $grad_i = 1 - y_i \sum_j \alpha_j y_j K(x_i, x_j)$ 
10:    if gradiente  $> 0$  then
11:      Atualize  $\alpha_i \leftarrow \alpha_i + \eta \cdot grad_i$ 
12:       $\alpha_i \leftarrow \max(0, \min(\alpha_i, C))$ 
13:    end if
14:  end for
15:  Calcule a variação total:  $\Delta\alpha = \sum_i |\alpha_i^{atual} - \alpha_i^{anterior}|$ 
16:  if  $\Delta\alpha < \epsilon$  then
17:    break
18:  end if
19: end while
20: Compilação do Modelo:
21: Calcule  $w = \sum_i \alpha_i y_i x_i$ 
22: Calcule o viés  $b$  usando os vetores de suporte:  $b = \frac{1}{\#SV} \sum_{i \in SV} (y_i - w^T x_i)$ 

```

2.4.7 Kernels

No contexto de problemas de classificação, como aqueles em que se deseja separar dados em diferentes categorias, às vezes os dados não podem ser separados de forma simples usando uma linha reta ou um hiperplano. Esses tipos de problemas são chamados de não-linearmente separáveis (DIAS; NETO, 2017). Um conjunto de dados que não pode ser dividido por uma única linha reta de maneira satisfatória.

Os kernels são conceitos fundamentais em aprendizado de máquina, especialmente em técnicas como classificadores SVMs e outras abordagens que envolvem transformações de dados em espaços de características de dimensões mais altas (SHALEV-SHWARTZ; BEN-DAVID, 2014). Um kernel é uma função matemática que calcula o produto interno entre dois vetores no espaço de características, mesmo que esses vetores não sejam explicitamente transformados para esse espaço.

Um exemplo clássico é o kernel gaussiano (também conhecido como kernel RBF - Radial Basis Function). Ele mede a similaridade entre dois pontos no espaço original e mapeia essa similaridade para um valor no espaço de características de dimensão mais alta (ROCHA NETO, 2011). O kernel gaussiano atribui maior valor a pontos mais próximos e valores menores a pontos mais distantes. Ao usar esse kernel em um modelo como o classificador SVM, você pode calcular produtos internos no espaço de características sem

realmente realizar a transformação explícita dos dados.

O grande benefício dos kernels é que eles permitem lidar com problemas complexos de separação sem a necessidade de conhecer explicitamente a transformação de características ou os próprios valores transformados. Isso economiza tempo computacional e simplifica a implementação de modelos que podem operar em espaços de dimensões muito altas.

Os kernels são funções matemáticas que nos permitem medir similaridades entre pontos em espaços de características de dimensões mais altas, sem a necessidade de realizar a transformação completa dos dados. Eles desempenham um papel essencial na transformação de problemas não-linearmente separáveis em problemas mais fáceis de serem tratados pelo classificador SVM.

Espaços de Hilbert fornecem a base matemática necessária para entender kernels em SVMs. A completude dos espaços de Hilbert garante que certas operações matemáticas sejam bem definidas e estáveis. Usando produtos internos em espaços de Hilbert, podemos definir kernels que permitem a separação de dados complexos em um espaço de maior dimensão, facilitando a construção de classificadores eficazes. Para uma melhor compreensão dos espaços de Hilbert, recomenda-se a leitura do Apêndice A.

Kernel Linear

O kernel linear é a forma mais simples de kernel e é usado quando os dados são linearmente separáveis. O produto interno padrão é utilizado para mapear os dados para um espaço de características de maior dimensão. A função do kernel linear é dada por:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

onde $\mathbf{x}_i, \mathbf{x}_j$ são os vetores de características de dois pontos de dados no espaço original.

Kernel Polinomial

O kernel polinomial é usado quando os dados não são linearmente separáveis, mas podem ser separados por um hiperplano de dimensão maior. Ele mapeia os dados para um espaço de características de maior dimensão usando uma função polinomial. O kernel polinomial é definido como:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d \quad (2.56)$$

onde x_i e x_j são os vetores de características de dois pontos de dados no espaço original, c é uma constante e d é a ordem do polinômio.

Kernel RBF (Radial Basis Function)

O kernel RBF é um dos kernels mais populares e flexíveis usados no SVM. Ele mapeia os dados para um espaço de características de dimensão infinita usando funções de base radial. O kernel RBF é definido como:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \quad (2.57)$$

onde x_i e x_j são os vetores de características de dois pontos de dados no espaço original, e σ é um parâmetro de largura que controla a influência dos pontos vizinhos. Os kernels são essenciais para o SVM, pois permitem que o modelo encontre soluções mesmo quando os dados não são linearmente separáveis no espaço original. O pseudocódigo do algoritmo SVM com uso kernel é descrito a seguir.

Algorithm 2 SVM

- 1: **Dado:** X (conjunto de dados de entrada)
 - 2: **Dado:** Y (vetor de classes correspondentes aos dados de entrada)
 - 3: **Dado:** C (parâmetro de penalização)
 - 4: **Dado:** $\text{Kernel}(\cdot, \cdot)$ (função de kernel)
 - 5: **Maximize:**
 - 6: $L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j)$
 - 7: **Sujeito a:**
 - 8: $\sum_{i=1}^n \alpha_i y_i = 0$
 - 9: $0 \leq \alpha_i \leq C$, para $i = 1, 2, \dots, n$
 - 10: **Encontre:**
 - 11: $\alpha^* = \arg \max_{\alpha} L(\alpha)$
 - 12: **Calcule:**
 - 13: $w = \sum_{i=1}^n \alpha_i^* y_i x_i$
 - 14: $b = y_k - \sum_{i=1}^n \alpha_i^* y_i K(x_i, x_k)$, onde α_k^* é um valor não nulo no intervalo $0 < \alpha_k^* < C$
 - 15: **Classifique novo ponto:**
 - 16: **Dado um novo ponto de entrada X_{new} , classifique-o usando:**
 - 17: $g(x_{\text{new}}) = \text{sign}(\sum_{i=1}^n \alpha_i^* y_i K(x_i, x_{\text{new}}) + b) \geq 0$
-

Comportamento dos Vetores de Suporte no SVM com Kernel RBF

No gráfico gerado pelo modelo SVM com kernel *Radial Basis Function* (RBF), os **vetores de suporte** desempenham um papel crucial na determinação da fronteira de decisão. Os vetores de suporte são os pontos de dados que estão localizados mais próximos da fronteira de decisão e, portanto, são críticos para a definição da margem que separa as duas classes.

Os vetores de suporte são representados no gráfico como pontos destacados, normalmente com círculos ao redor, e eles se encontram tanto na borda da margem de cada classe quanto em algumas situações, diretamente na fronteira de decisão. A SVM ajusta essa fronteira de forma a maximizar a distância (margem) entre os vetores de suporte das duas classes.

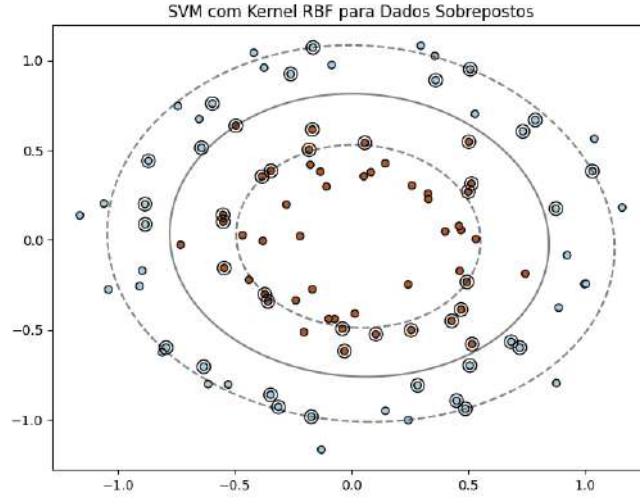


Figura 1 – SVM com Kernel RBF

2.5 Sequential minimal optimization (SMO)

O modelo sequencial minimal optimization (SMO) é uma técnica popular para resolver o problema de otimização do classificador SVM. Ele foi proposto por John Platt em 1998 e é amplamente utilizado devido à sua eficiência para problemas de grande escala. Como mostrado antes nas seções anteriores, o problema de otimização é formulado como:

Minimizar:

$$\frac{1}{2} \|\mathbf{w}\|^2 \quad (2.58)$$

sujeito a

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 0 \quad \text{para } i = 1, 2, \dots, n, \quad (2.59)$$

onde $\mathbf{w} \in \mathbb{R}^d$ é o vetor de pesos, b é o termo de viés e y_i é a classe do ponto de treinamento $\mathbf{x}_i$.

Maximizar:

$$\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (2.60)$$

sujeito a

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (2.61)$$

$$0 \leq \alpha_i \leq C \quad \forall i \quad (2.62)$$

onde C é uma constante de regularização.

O modelo SMO resolve iterativamente este problema de otimização, selecionando um par de multiplicadores α_i e α_j para atualização em cada iteração (PLATT, 1998b). Inicialize os multiplicadores de Lagrange α_i e o termo de viés b com valores iniciais, geralmente zero. Enquanto não convergir (ou até que um critério de parada seja atendido):

1. Para cada par de multiplicadores α_i e α_j , selecione um par para atualização de acordo com uma heurística (como a primeira e segunda heurísticas).
2. Fixe todos os outros multiplicadores α_k para $k \neq i, j$ e resolva o problema de otimização restrito a esses dois multiplicadores. O objetivo é encontrar os valores ótimos de α_i e α_j que maximizem a função objetivo enquanto satisfazem as restrições.
3. Atualize os multiplicadores ótimos α_i e α_j com os novos valores encontrados.

Após a convergência, calcule o vetor de pesos $\mathbf{w}$ e o termo de viés b usando as fórmulas:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \quad (2.63)$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i \quad \text{para qualquer vetor de suporte } \mathbf{x}_i \text{ com } 0 < \alpha_i < C \quad (2.64)$$

2.5.1 Primeira e segunda Heurística

A primeira heurística do SMO escolhe aleatoriamente um multiplicador α_i e, em seguida, procura um segundo multiplicador α_j que maximize a quantidade $|E_i - E_j|$, onde E_i é o erro de predição para o ponto de treinamento $\mathbf{x}_i$. A fórmula para atualizar os multiplicadores é dada por (PLATT, 1998a):

$$\alpha_j^{\text{novo}} = \alpha_j^{\text{antigo}} + \frac{y_j(E_i - E_j)}{\eta} \quad (2.65)$$

onde:

$$\eta = 2K_{ij} - K_{ii} - K_{jj} \quad (2.66)$$

é a taxa de aprendizado calculada a partir dos produtos internos dos vetores de suporte $\mathbf{x}_i$ e $\mathbf{x}_j$, e $K_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$. A ideia intuitiva por trás da primeira heurística é bastante simples. Dessa forma, quando o SMO examina um exemplo, ele verifica se as condições de Karush-Kuhn-Tucker (KKT) estão sendo violadas. Lembrando que pelo menos uma das condições de KKT deve ser violada. Se as condições forem atendidas, ele tenta outro exemplo. Dessa maneira, se existirem milhões de exemplos e apenas alguns deles violarem as condições de KKT, o modelo gastará muito tempo examinando exemplos inúteis (PLATT, 1998a). Para evitar isso, o modelo concentra seu tempo em exemplos nos quais o multiplicador de Lagrange não é igual a 0, porque são os mais propensos a violar as condições. Uma vez que a resolução analítica do problema implica a manipulação de dois multiplicadores de Lagrange, há a possibilidade de que um multiplicador vinculado (cujo valor fique entre 0 e C) possa violar as condições de Karush-Kuhn-Tucker (KKT). Por essa razão, o procedimento principal alterna entre todos os exemplos e o subconjunto de exemplos não vinculados.

A segunda heurística do SMO seleciona os multiplicadores α_i e α_j de modo a manter a quantidade $\sum_{i=1}^n \alpha_i y_i = 0$. Uma vez que α_i é atualizado, α_j é ajustado para

garantir essa propriedade. A fórmula para a atualização de α_j é dada por (PLATT, 1998b):

$$[\alpha_j^{\text{novo}} = \alpha_j^{\text{antigo}} + y_j y_i (\alpha_i^{\text{antigo}} - \alpha_i^{\text{novo}}) \quad (2.67)$$

O modelo SMO é uma técnica eficiente para resolver o problema de otimização do SVM, permitindo a classificação de dados em larga escala. Ele é amplamente utilizado em diversas aplicações de aprendizado de máquina e tem sido fundamental para o sucesso do SVM como um classificador poderoso.

2.5.2 O Hiperplano no SMO e os Vetores de Suporte

Em termos simples, o hiperplano treinado pelo SMO é uma superfície que tenta maximizar a margem de separação entre as duas classes de dados. A margem é a distância mínima entre o hiperplano e os pontos de qualquer classe, e o SMO procura maximizar essa margem, o que resulta em um modelo mais generalizável.

Os **vetores de suporte** são os pontos de dados que estão localizados nas margens ou, em alguns casos, muito próximos a elas. Estes pontos são críticos porque eles são os que definem a posição e a orientação do hiperplano.

Em resumo, o hiperplano no SMO é a fronteira de decisão que separa as duas classes, enquanto os vetores de suporte são os pontos de dados que têm a maior influência na posição e orientação desse hiperplano.

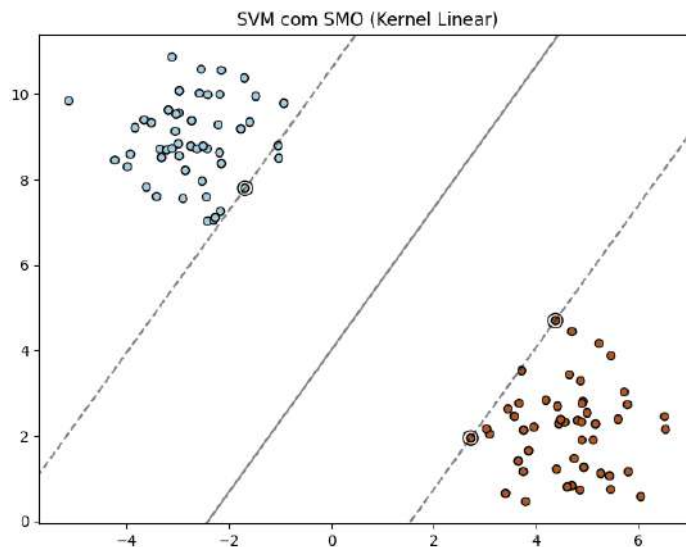


Figura 2 – Gráfico $dmSVM$

Pseudocódigo para treinamento do SMO:

Algorithm 3 Sequential Minimal Optimization (SMO) com Kernel

```

1: Entrada: Conjunto de treinamento  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ , função de kernel  $K$ , parâmetro de regula-
   rização  $C$ , tolerância  $\epsilon$ 
2: Saída: Multiplicadores  $\alpha_i$  e viés  $b$ 
3: Inicialize  $\alpha_i \leftarrow 0$  para todos  $i$ ,  $b \leftarrow 0$ 
4: while não atingir a convergência do
5:   Selecione o par  $(\alpha_i, \alpha_j)$  por heurística
6:   Calcule  $E_i = f(x_i) - y_i$  e  $E_j = f(x_j) - y_j$ 
7:   Salve os valores antigos de  $\alpha_i$  e  $\alpha_j$  como  $\alpha_{i,old}$  e  $\alpha_{j,old}$ 
8:   Calcule os limites  $L$  e  $H$  para  $\alpha_j$ 
9:   if  $y_i \neq y_j$  then
10:     $L \leftarrow \max(0, \alpha_j - \alpha_i)$ ,  $H \leftarrow \min(C, C + \alpha_j - \alpha_i)$ 
11:   else
12:     $L \leftarrow \max(0, \alpha_i + \alpha_j - C)$ ,  $H \leftarrow \min(C, \alpha_i + \alpha_j)$ 
13:   end if
14:   if  $L == H$  then
15:     continue
16:   end if
17:   Calcule  $\eta = 2K(x_i, x_j) - K(x_i, x_i) - K(x_j, x_j)$ 
18:   if  $\eta \geq 0$  then
19:     continue
20:   end if
21:    $\alpha_j \leftarrow \alpha_j - \frac{y_j(E_i - E_j)}{\eta}$ 
22:   Limite  $\alpha_j$  para estar dentro de  $[L, H]$ 
23:    $\alpha_i \leftarrow \alpha_i + y_i y_j (\alpha_{j,old} - \alpha_j)$ 
24:   Atualize o viés  $b$ 
25:   Atualize o cache de erros para todos os exemplos usando os novos  $\alpha_i$  e  $\alpha_j$ 
26: end while
27: retorne  $\alpha_i, b$ 

```

2.6 Conclusão

Neste capítulo, exploramos a fundo as Máquinas de Vetores de Suporte (SVMs). Foi explicado a teoria por trás da construção de hiperplanos de separação ótima, compreendendo a importância dos vetores de suporte e a maximização da margem. Adicionalmente, aprofundamos os métodos de treinamento, como o algoritmo SMO, e a utilização de kernels para lidar com dados não linearmente separáveis. A aplicação do kernel Adatron foi apresentada como um caso particular, demonstrando a flexibilidade das SVMs. Evidenciamos o amplo espectro de aplicações das SVMs, desde problemas de classificação binária até tarefas mais complexas.

3 Novos algoritmos de treinamento para SVM

3.1 Algoritmo de treinamento com base na distância

Este capítulo introduz um novo método para treinar máquinas de vetores de suporte (SVMs), doravante referido como d -SVM. Esta abordagem inovadora se baseia no cálculo da função de decisão, descrita pela função $g(\mathbf{x})$ apresentada na Eq.(17), antes da aplicação da função de sinal. A função $g(\mathbf{x})$ pode ser vista como uma medida de distância, uma vez que o valor absoluto $|g(\mathbf{x})|$ indica quão distante um ponto está do hiperplano de separação: $g(\mathbf{x}) = 0$ se está sobre o hiperplano, $g(\mathbf{x}) > 0$ se está do lado da classe positiva e $g(\mathbf{x}) < 0$ para a classe negativa.

A ideia que fundamenta a proposta reside no uso eficaz da função $g(\mathbf{x})$ para o ajuste preciso dos multiplicadores de Lagrange. Para iniciar este processo, é fundamental obter uma estimativa preliminar de $g(\mathbf{x})$, o que é feito utilizando valores iniciais aleatórios dos multiplicadores, embora um valor constante como C também possa ser usado. Baseado nessa estimativa, desenvolvemos métodos para ajustar os multiplicadores de Lagrange conforme descrito a seguir.

3.1.1 Ajuste dos multiplicadores de Lagrange

Em nossa proposta, todos os multiplicadores de Lagrange são iniciados aleatoriamente entre 0 e C , seguindo uma distribuição uniforme $U(0, C)$. Posteriormente, os multiplicadores de Lagrange de um percentual(%) P de padrões para cada uma das classe positiva e negativa são ajustados iterativamente em duas etapas. Na primeira, como dito, há um ajuste de $P\%$ dos multiplicadores de Lagrange escolhidos com base no menor valor de $g(\mathbf{x})$. Os $P\%$ dos multiplicadores de Lagrange a serem ajustados são selecionados também com base na função $g(\mathbf{x})$. Enquanto na segunda, são ajustados apenas os multiplicadores de Lagrange dos padrões incorretamente classificados.

Ao se analisar a função de decisão para as máquinas de vetores suporte pode-se perceber que a saída/classe é definida com base na média ponderada¹ dos vetores suporte para cada uma das classes. A fim de tornar mais evidente isto, percebe-se que uma vez que o rótulo $y_i \in \{-1, +1\}$ então tem-se que

$$g(\mathbf{x}) = \left[\sum_{i \in SV^+} \alpha_i \mathbf{x}_i^T - \sum_{j \in SV^-} \alpha_j \mathbf{x}_j^T \right] \mathbf{x} + b. \quad (3.1)$$

¹ Média ponderada pelos valores dos multiplicadores de Lagrange não-nulos associados a cada um dos padrões de treinamento.

A equação acima pode ser vista também em função direta de valores médios ponderados como segue

$$g(\mathbf{x}) = [\mu_i^T - \mu_j^T] \mathbf{x} + b \quad (3.2)$$

Em que SV^+ e SV^- denotam os conjuntos de índices dos padrões para as classes positivas e negativas, respectivamente. Já as médias $\mu_i^T = \sum_{i \in SV^+} \alpha_i \mathbf{x}_i^T$ e $\mu_j^T = \sum_{j \in SV^-} \alpha_j \mathbf{x}_j^T$ são as médias ponderadas das classes positiva e negativa, respectivamente. E, com base nesta média ponderada, verifica-se a qual das duas médias (μ_i ou μ_j) o padrão $\mathbf{x}$ é mais similar (i.e., quando tem o maior valor).

3.1.2 Regra de Ajustes dos Multiplicadores de Lagrange

A regra de ajuste dos multiplicadores de Lagrange para a primeira etapa, em nossa proposta, leva em consideração os padrões que estão mais distantes e também os que estão mais próximos do hiperplano ótimo para cada uma das classes. Mais especificamente, para a classe positiva ($y_i = +1$ e $i \in SV^+$), é necessário que se obtenha os valores mínimo

$$g_{\min}^+ = g(\mathbf{x}_{i_{\min}^+}) \quad (3.3)$$

e máximo

$$g_{\max}^+ = g(\mathbf{x}_{i_{\max}^+}), \quad (3.4)$$

em que

$$i_{\min}^+ = \arg \min \{g(\mathbf{x}_i)\} \quad (3.5)$$

e

$$i_{\max}^+ = \arg \max \{g(\mathbf{x}_i)\} \quad (3.6)$$

são os índices dos padrões mais próximos e mais distantes do hiperplano ótimo da classe positiva, respectivamente. De forma similar, em nossa proposta, obtem-se os valores $g_{\min}^-$ e $g_{\max}^-$. Com base nestes valores $g_{\min}^-$, $g_{\max}^-$, $g_{\min}^+$ e $g_{\max}^+$, são utilizados cálculos para indicar a posição relativa de um padrão de treinamento qualquer em relação ao seu pares mais longes e mais próximos. Para tal, são utilizadas as seguintes expressões para as posições relativas de padrões da classe positiva

$$r_i = \frac{g(\mathbf{x}) - g_{\min}^+}{g_{\max}^+ - g_{\min}^+} \quad (3.7)$$

e para posições relativas da negativa

$$r_j = \frac{g_{\max}^- - g(\mathbf{x})}{g_{\max}^- - g_{\min}^-}, \quad (3.8)$$

Pode-se perceber que os valores de $r_i \in [0, 1]$ e $r_j \in [0, 1]^2$; pois, quando o padrão é o mais próximo, o valor de $r_* = 0$. No entanto, quando o padrão é o mais distante o valor de $r_* = 1$.

² r_* significa r_i ou r_j

Com os valores de r_i e r_j , tem-se as seguintes regras de atualização dos multiplicadores de Lagrange das classes positiva

$$\alpha_i = r_i \cdot 0.1C + (1 - r_i) \cdot C \quad (3.9)$$

e negativa

$$\alpha_j = r_j \cdot 0.1C + (1 - r_j) \cdot C, \quad (3.10)$$

respectivamente. Após a atualização, os valores de α_i e α_j encontram-se no intervalo $[0.1C, C]$, ou seja, os P% padrões mais próximos tem valores nesse intervalo e os demais tem valores iguais a zero. Atribuir valor zero aos demais padrões visa a manter esparsidade no vetor de multiplicadores de Lagrange.

Na segunda etapa, busca-se um ajuste fino da superfície de decisão pela atualização dos multiplicadores de Lagrange dos padrões de treinamento que são incorretamente classificados, ou seja, quando

$$E_i = y_i f(\mathbf{x}_i) = -1. \quad (3.11)$$

Na situação oposta, quando o padrão é corretamente classificado, tem-se $E_i = 1$. Nesta etapa de ajuste, busca-se aumentar a influência do padrão, por exemplo, da classe positiva incorretamente classificado pelo aumento do valor do seu multiplicador de Lagrange; fazendo assim com que a média ponderada da classe μ_i (e também o hiperplano de separação) se desloque em direção a classe negativa. Análise similar pode ser feita para o ajuste de um padrão da classe negativa incorretamente classificado. O ajuste dos multiplicadores de Lagrange nessa etapa é aplicado aos padrões cujo os índices pertencem ao conjunto

$$I_{err} = \{k \mid E_k = -1, \forall k = 1, \dots, N\}, \quad (3.12)$$

que contém os índices dos padrões $\mathbf{x}_k$ que são incorretamente classificados, ou seja, que possuem $E_k = -1$. Logo, os padrões $\mathbf{x}_k$ em que $k \in I_{err}$ são ajustados pela seguinte regra de atualização

$$\alpha_k = \min(\alpha_k + \Delta\alpha_k, C) \quad (3.13)$$

em que,

$$\Delta\alpha_k = [1 - y_k g(\mathbf{x}_k)] 0.1C, \quad (3.14)$$

Se $g(\mathbf{x}_k)$ é pequeno e $y_k g(\mathbf{x}_k) < 0$ devido a um erro de classificação, $1 - y_k g(\mathbf{x}_k)$ será um pouco maior que 1. Isso resulta em um aumento de cerca de 10% do valor máximo C para o multiplicador de Lagrange α_k , fazendo com que o padrão incorretamente classificado influencie mais, ajustando o hiperplano para corrigir a classificação. Esse ajuste ocorre iterativamente até que não haja mais erros em uma das classes. As condições de otimalidade são:

$$\left[\sum_{i \in SV^+} \alpha_i - \sum_{j \in SV^-} \alpha_j \right] = 0, \quad (3.15)$$

A diferença $\gamma = \left[\sum_{i \in SV^+} \alpha_i - \sum_{j \in SV^-} \alpha_j \right]$ é subtraída dos vetores suporte da classe positiva se $\gamma > 0$, ou dos da classe negativa se $\gamma < 0$, sempre respeitando $0 \leq \alpha_i \leq C$. Finalmente, com os multiplicadores de Lagrange, calcula-se o viés b para definir a função de decisão do modelo. O pseudocódigo segue abaixo.

Algorithm 4 d -SVM

```

1: Entradas:  $X, y, P$ , e  $C$ 
2:  $N \leftarrow$  número de padrões em  $X$ 
3: Inicie cada  $\alpha_i$  com valores aleatórios entre 0 e  $C$ , usando uma distribuição uniforme  $U(0, C)$ 
4: Faça ajuste nos valores dos  $\alpha_i$ , conforme Eq. (3.15)
5: Calcule  $g(\mathbf{x}_i)$  para todos os padrões em  $X$ 
6: Obtenha  $g_{\min}^+, g_{\max}^+, g_{\min}^-$  e  $g_{\max}^-$ 
7: Selecione os  $P\%$  padrões mais próximos, usando  $|g(\mathbf{x}_i)|$ .
8: Defina  $\alpha_i = 0$  para os padrões não selecionados.
9: Defina  $\alpha_i$  para os padrões selecionados como segue.
10: for cada padrão dentre os selecionados do
11:   if  $y = +1$  then
12:      $r_i \leftarrow \frac{g(\mathbf{x}) - g_{\min}^+}{g_{\max}^+ - g_{\min}^+}$ 
13:      $\alpha_i \leftarrow r_i \cdot \frac{C}{10} + (1 - r_i) \cdot C$ 
14:   else
15:      $r_j \leftarrow \frac{g_{\max}^- - g(\mathbf{x})}{g_{\max}^- - g_{\min}^-}$ 
16:      $\alpha_j \leftarrow r_j \cdot \frac{C}{10} + (1 - r_j) \cdot C$ 
17:   end if
18: end for
19:  $I_{err}^+ \leftarrow \{i \in \{1, \dots, N\} \mid y_i f(\mathbf{x}_i) = -1, y_i = +1\}$ 
20:  $I_{err}^- \leftarrow \{j \in \{1, \dots, N\} \mid y_j f(\mathbf{x}_j) = -1, y_j = -1\}$ 
21: while  $I_{err}^+ \neq \emptyset$  e  $I_{err}^- \neq \emptyset$  do
22:   for  $k \in I_{err}^+ \cup I_{err}^-$  do
23:      $\alpha_k \leftarrow \min(\alpha_k + [1 - y_k g(\mathbf{x}_k)]0.1C, C)$ 
24:   end for
25:    $I_{err}^+ \leftarrow \{i \in \{1, \dots, N\} \mid y_i f(\mathbf{x}_i) = -1, y_i = +1\}$ 
26:    $I_{err}^- \leftarrow \{j \in \{1, \dots, N\} \mid y_j f(\mathbf{x}_j) = -1, y_j = -1\}$ 
27: end while
28: Compute o viés  $b$ 
29: Considere o vetor  $\alpha^T = [\alpha_1, \alpha_2, \dots, \alpha_N]^T$ 
30: Output:  $\alpha$  e  $b$ 

```

3.2 Segunda Proposta: Algoritmo de treinamento com base na distância média

Na segunda proposta do nosso modelo dm -SVM, a necessidade de testar uma porcentagem específica de vetores de suporte é eliminada, pois a quantidade de ajustes é determinada automaticamente com base na dinâmica do problema. Esta abordagem aprimora o processo de ajuste dos multiplicadores de Lagrange (α), utilizando as distâncias calculadas e os centros das classes para refinar a capacidade da máquina de vetores de

suporte de discriminar entre as classes.

Para isso, fazemos o cálculo dos centros das classes ponderados. Os centros das classes, tanto positivos quanto negativos, são calculados de forma ponderada pelos valores de α . A ideia segue a mesma lógica no uso da função $g(\mathbf{x})$ para ajustar os multiplicadores de Lagrange, fazendo uma estimativa aproximada da função $g(\mathbf{x})$, que em nossa proposta é obtida com base em valores iniciais aleatórios para os multiplicadores³. Subsequentemente, funções de ajuste para os multiplicadores de Lagrange são propostas com base na função $g(\mathbf{x})$.

Onde SV^+ e SV^- denotam os conjuntos de índices dos vetores de suporte para as classes positiva e negativa, respectivamente. As médias $\mu_i^T = \sum_{i \in SV^+} \alpha_i \mathbf{x}_i^T$ e $\mu_j^T = \sum_{j \in SV^-} \alpha_j \mathbf{x}_j^T$ são as médias ponderadas das classes positiva e negativa, respectivamente. Com base nesta média ponderada, verifica-se a qual das duas médias (μ_i ou μ_j) o padrão $\mathbf{x}$ é mais semelhante (ou seja, qual tem o valor mais alto).

$$g(\mathbf{x}) = [\mu_i^T - \mu_j^T] \mathbf{x} + b, \quad (3.16)$$

3.2.1 Regra para Ajuste dos Multiplicadores de Lagrange

A regra de ajuste para os multiplicadores de Lagrange, em nossa proposta, leva em conta os padrões mais distantes e mais próximos do hiperplano ótimo para cada classe. Especificamente, para a classe positiva ($y_i = +1$ e $i \in SV^+$), definimos:

$$g_{\min}^+ = g(\mathbf{x}_{i_{\min}^+}), \quad i_{\min}^+ = \arg \min \{g(\mathbf{x}_i) \mid i \in SV^+\} \quad (3.17)$$

$$g_{\max}^+ = g(\mathbf{x}_{i_{\max}^+}), \quad i_{\max}^+ = \arg \max \{g(\mathbf{x}_i) \mid i \in SV^+\} \quad (3.18)$$

Esses índices $i_{\min}^+$ e $i_{\max}^+$ representam os índices dos padrões que estão mais próximos e mais distantes do hiperplano ótimo, respectivamente.

Com base nesses valores $g_{\min}^-$, $g_{\max}^-$, $g_{\min}^+$ e $g_{\max}^+$, são feitas computações para indicar a posição relativa de qualquer padrão de treinamento em relação aos seus pares mais distantes e mais próximos. Para isso, as seguintes expressões para as posições relativas dos padrões da classe positiva são utilizadas:

As posições relativas r_i para a classe positiva e r_j para a classe negativa são dadas por:

³ Um valor constante também pode ser usado inicialmente, por exemplo, igual a C

$$r_i = \frac{g(\mathbf{x}_i) - g_{\min}^+}{g_{\max}^+ - g_{\min}^+}, \quad (3.19)$$

$$r_j = \frac{g_{\max}^- - g(\mathbf{x}_i)}{g_{\max}^- - g_{\min}^-}, \quad (3.20)$$

onde os valores de r_i e r_j são limitados dentro do intervalo $[0, 1]$. Especificamente, r^* é igual a 0 quando o padrão está mais próximo do hiperplano ótimo e 1 quando está mais distante.

Com os valores de r_i e r_j , as seguintes regras para atualizar os multiplicadores de Lagrange das classes positiva e negativa são aplicadas, respectivamente:

$$\alpha_i = r_i \cdot 0.1C + (1 - r_i) \cdot C \quad (3.21)$$

$$\alpha_j = r_j \cdot 0.1C + (1 - r_j) \cdot C \quad (3.22)$$

Após a atualização, os valores de α_i e α_j estão no intervalo $[0.1C, C]$, ou seja, os padrões mais próximos têm valores nesse intervalo e os valores restantes são definidos como zero. Atribuir um valor zero aos outros padrões visa manter a esparsidade no vetor de multiplicadores de Lagrange.

3.2.2 Parte Dois do Ajuste

Na continuação do nosso modelo d -SVM proposto, refinamos o ajuste dos multiplicadores de Lagrange (α) usando as distâncias calculadas e os centros das classes. Esta parte do processo de ajuste é crucial para ajustar efetivamente a máquina de vetores de suporte para discriminar melhor entre as classes.

Cálculo dos Centros das Classes Ponderados: Inicialmente, calculamos os centros das classes ponderados pelos valores correspondentes de α . Isso é feito separadamente para as classes positiva e negativa usando a fórmula:

$$\mathbf{x}_{\text{pos}} = \frac{\sum_{i:y_i>0} \alpha_i \mathbf{x}_i}{\sum_{i:y_i>0} \alpha_i} \quad (3.23)$$

$$\mathbf{x}_{\text{neg}} = \frac{\sum_{i:y_i<0} \alpha_i \mathbf{x}_i}{\sum_{i:y_i<0} \alpha_i} \quad (3.24)$$

Esses centros representam a posição "média" de cada classe no espaço de características, ponderada pela influência de cada padrão conforme ditada pelo seu valor de α . Em seguida, calculamos as predições reais d_{pos} e d_{neg} nesses centros usando a função `predict_real`, que avalia a função de decisão sem aplicar o limiar de sinal:

$$d_{\text{pos}} = g(\mathbf{x}_{\text{pos}}), d_{\text{neg}} = g(\mathbf{x}_{\text{neg}}) \quad (3.25)$$

Atualização Final de α : Em seguida, identificamos os índices onde são necessários ajustes nos valores de α com base em sua desvio em relação a esses valores centrais:

$$\text{ind1} = \{i \mid y_i > 0, g(\mathbf{x}_i) < d_{\text{pos}}\}, \quad (3.26)$$

$$\text{ind2} = \{i \mid y_i < 0, g(\mathbf{x}_i) > d_{\text{neg}}\} \quad (3.27)$$

Os valores de α para esses índices são ajustados para zero para impor as restrições de margem e aumentar a esparsidade do modelo:

$$\alpha_i = 0, \quad \forall i \in \text{ind1} \cup \text{ind2}$$

Após este ajuste, o multiplicador de Lagrange é recalibrado de acordo com as condições de Karush-Kuhn-Tucker (KKT). **Cálculo do Intercepto b :** Finalmente, calculamos o intercepto b do SVM usando os vetores de suporte que estão estritamente dentro da margem:

$$b = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (y_i - g(\mathbf{x}_i))$$

Este cálculo determina o deslocamento médio necessário para alinhar as predições com os rótulos de classe reais, centrando assim a fronteira de decisão. O modelo de treinamento para máquinas de vetores de suporte (SVMs), que é eficiente do ponto de vista computacional em termos de tempo de treinamento do modelo, foi desenvolvido com o objetivo de otimizar o desempenho e reduzir os recursos necessários durante o processo de treinamento. A seguir, apresentamos o pseudocódigo do modelo.

Algorithm 5 *dm-SVM*

```

1: Cálculo de novos valores de  $\alpha$ :
2:  $g(x) \leftarrow \text{predict\_real}(X)$ 
3: Calcular  $g_{\min}^+$  mínimo e  $g_{\max}^+$  máximo de  $g(x)$  para  $y = 1$ 
4: Calcular  $g_{\min}^-$  mínimo e  $g_{\max}^-$  máximo de  $g(x)$  para  $y = -1$ 
5: Normalizar  $g(x)$  em cada classe e reajustar  $\alpha$ :
6: if ( $y = +1$ ) then
7:    $r_i \leftarrow \frac{g(\mathbf{x}) - g_{\min}^+}{g_{\max}^+ - g_{\min}^+}$ 
8:    $\alpha_i \leftarrow r_i \cdot \frac{C}{10} + (1 - r_i) \cdot C$ 
9: else
10:   $r_j \leftarrow \frac{g_{\max}^- - g(\mathbf{x})}{g_{\max}^- - g_{\min}^-}$ 
11:   $\alpha_j \leftarrow r_j \cdot \frac{C}{10} + (1 - r_j) \cdot C$ 

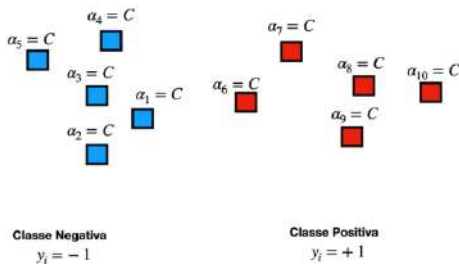
12: Calcular  $x_{\text{pos}}$  e  $x_{\text{neg}}$  usando  $\alpha_i$  para classes positivas e negativas respectivamente
13:  $d_{\text{pos}} \leftarrow g(x_{\text{pos}})$ 
14:  $d_{\text{neg}} \leftarrow g(x_{\text{neg}})$ 
15:  $\text{ind1} \leftarrow \{i \mid y_i > 0 \wedge g(x_i) < d_{\text{pos}}\}$ 
16:  $\text{ind2} \leftarrow \{i \mid y_i < 0 \wedge g(x_i) > d_{\text{neg}}\}$ 
17:  $\alpha \leftarrow \text{adjust\_alphas\_wKKT}(\alpha, \mathbf{y})$ 

18: Cálculo do Intercepto  $b$ :
19: Encontrar suportes onde  $0 < \alpha < C$ 
20: if vetores de suporte existem then
21:    $gx \leftarrow g(X[\text{suporte}])$ 
22:    $b \leftarrow \text{média}(y[\text{suporte}] - gx)$ 
23: end if

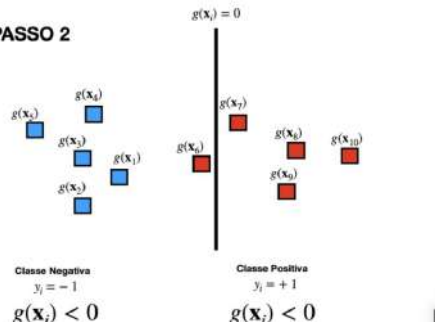
```

3.3 Análise gráfica das etapas do d-SVM

Nas imagens apresentadas nas figuras dessa seção são mostrados de forma ilustrativa os passos do algoritmo de treinamento d-SVM para um problema linear e não linear. -0.1cm

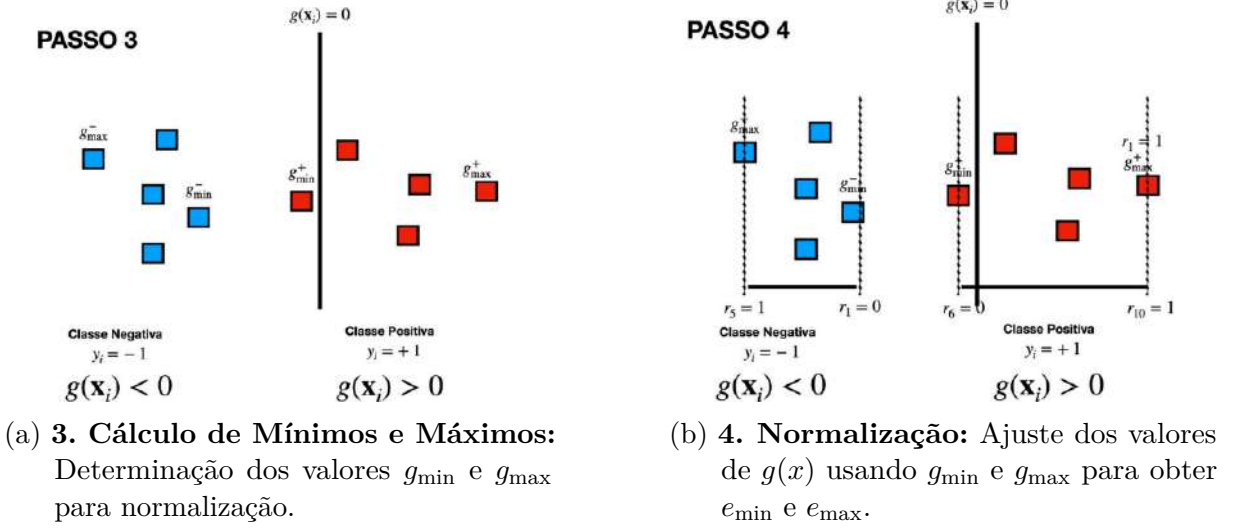
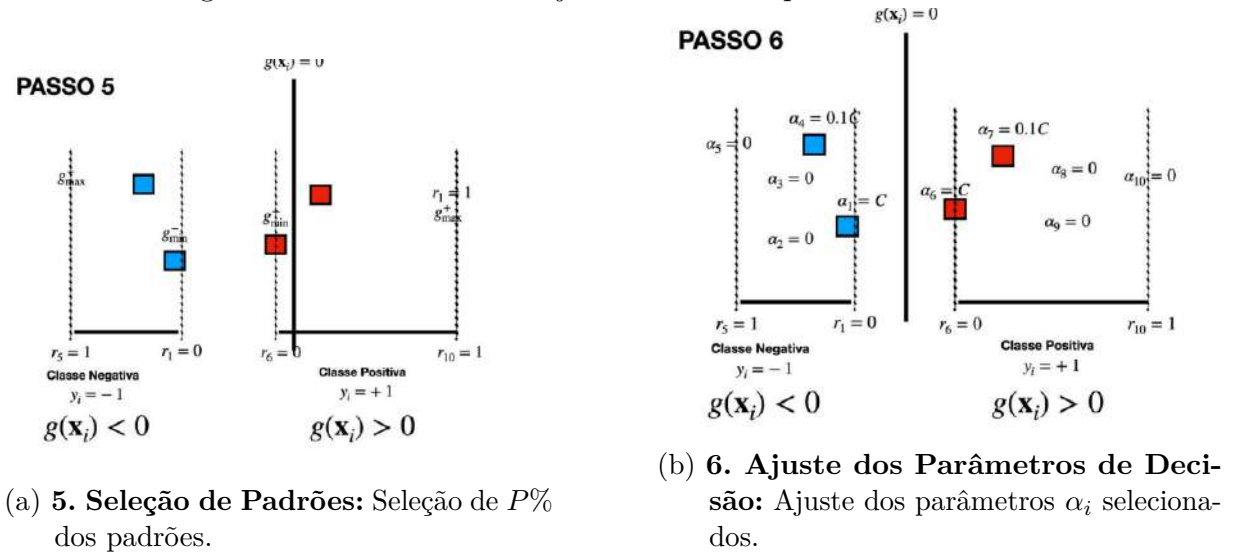
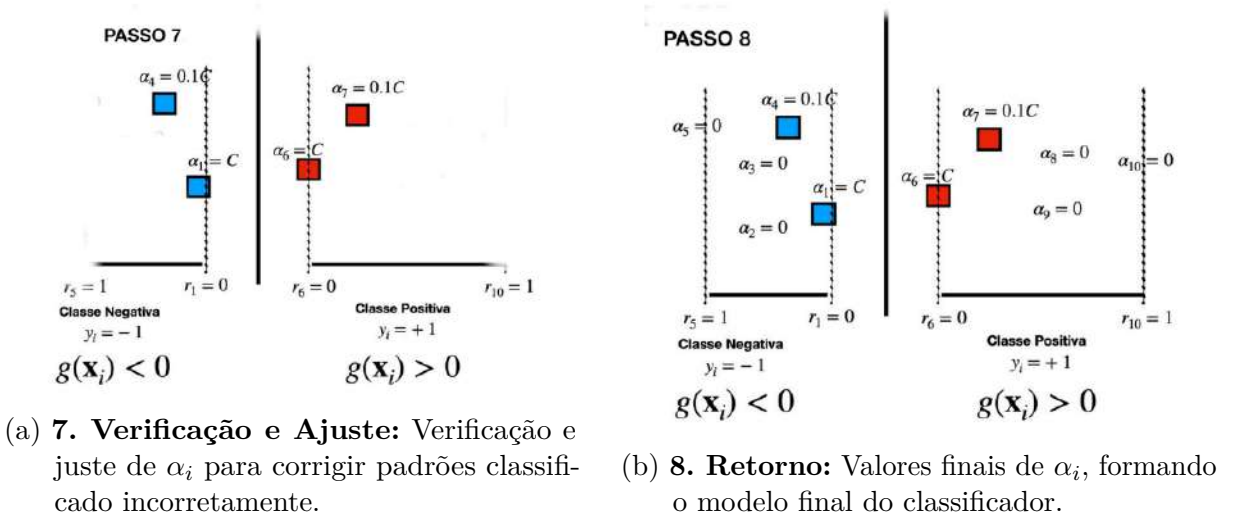
PASSO 1

(a) **1. Estimativas:** $g(x)$ como estimativa inicial para $y = 1$ e $y = -1$.

PASSO 2

(b) **2. Cálculo de Valores para Cada Padrão:** Avaliação de $g(x)$ para cada padrão.

Figura 3 – Passos 1 e 2 do ajuste do *d*-SVM para dados lineares.

Figura 4 – Passos 3 e 4 do ajuste do d -SVM para dados lineares.Figura 5 – Passos 5 e 6 do ajuste do d -SVM para dados lineares.Figura 6 – Passos 7 e 8 do ajuste do d -SVM para dados lineares.



- (a) **1. Estimativas:** $g(x)$ como estimativa inicial para $y = 1$ e $y = -1$.

- (b) **2. Cálculo de Valores:** Normalização de $g(x)$ e ajuste de α .

Figura 7 – Primeiro e segundo passos do ajuste do d-SVM.



- (a) **3. Mínimos e Máximos:** Determinação de $g_{\min}$ e $g_{\max}$.

- (b) **4. Normalização:** Ajuste de $g(x)$ com $g_{\min}$ e $g_{\max}$.

Figura 8 – Terceiro e quarto passos do ajuste do d-SVM.



- (a) **5. Seleção de Padrões:** Seleção de $P\%$ com menor $g(x)$.

- (b) **6. Ajuste dos Parâmetros:** Ajuste de α_i para o hiperplano.

Figura 9 – Quinto e sexto passos do ajuste do d-SVM.



- (a) **7. Verificação e Ajuste:** Verificação de padrões incorretos e ajuste de α_i .

- (b) **8. Otimização:** Ajuste de α_i para soma igual a zero.

Figura 10 – Sétimo e oitavo passos do ajuste do d-SVM.

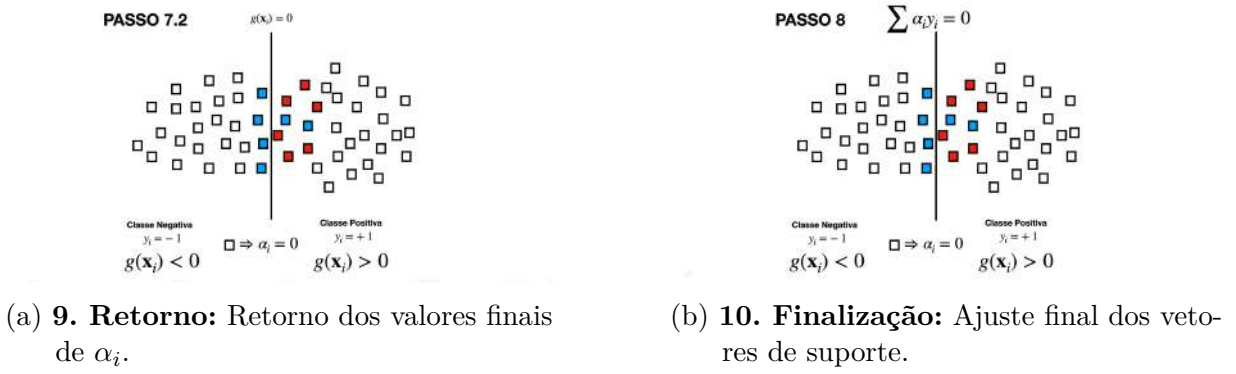


Figura 11 – Nono e décimo passos do ajuste do d-SVM.

3.3.1 Visualização dos modelos d -SVM e dm -SVM após ajuste dos parâmetros utilizando kernels para dados artificiais.

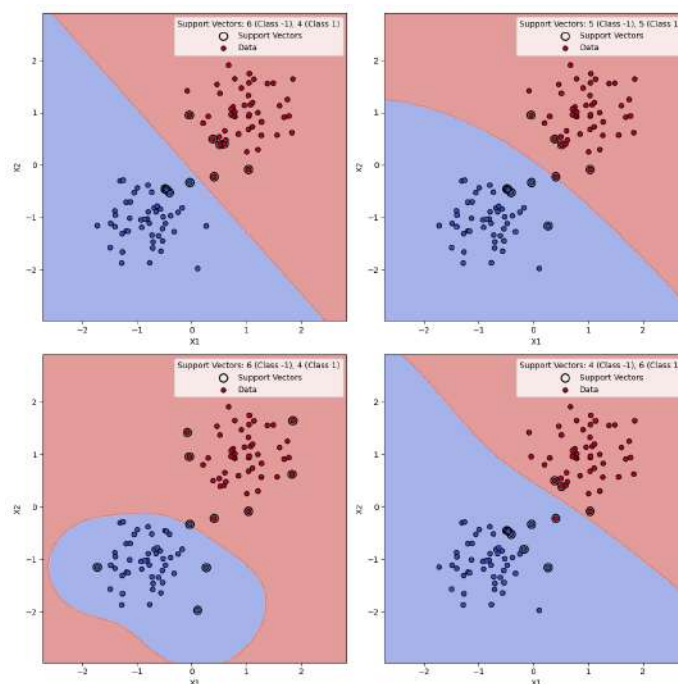


Figura 12 – Visualização do modelo d -SVM após ajuste dos parâmetros

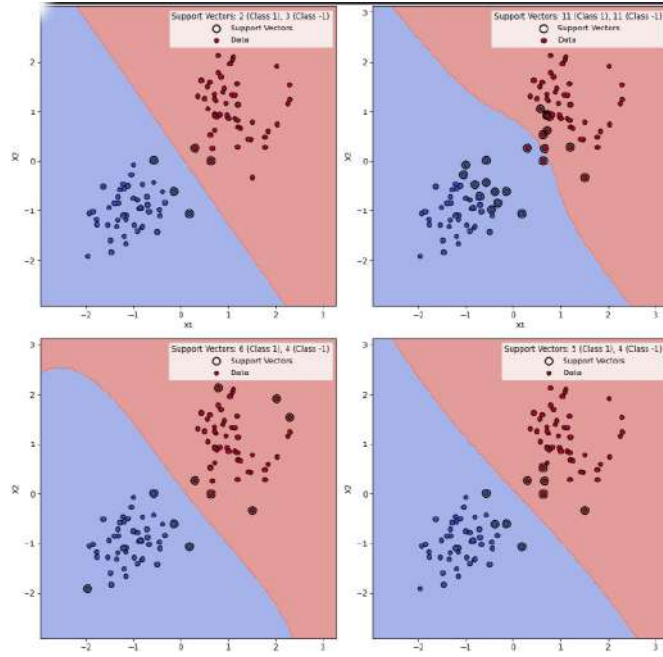


Figura 13 – Visualização do modelo dm -SVM após ajuste dos parâmetros

O d -SVM como visto, ajusta os valores de α_i dos vetores de suporte a partir de um $p\%$ definido como hiperparâmetro. Ele utiliza os padrões mais próximos das margens para definir os vetores de suporte, buscando garantir que apenas os vetores mais relevantes sejam considerados na separação das classes. A quantidade de vetores de suporte é fixa, mesmo podendo variar de acordo com a complexidade do conjunto de dados.

O dm -SVM, por outro lado, permite uma quantidade variável de vetores de suporte. Ele ajusta os valores de α dinamicamente com base na normalização dos valores preditos para cada classe. Esse modelo se adapta melhor às características dos dados ao permitir que a quantidade de vetores de suporte varie conforme necessário, proporcionand maior flexibilidade na modelagem das fronteiras.

Linearidade e Simplicidade: Observando o gráfico 12, o d -SVM tende a criar fronteiras de decisão mais simples e lineares, refletindo a natureza mais física e estática da quantidade de vetores de suporte. A simplicidade da fronteira de decisão pode ser observada na uniformidade das margens e na quantidade de vetores de suporte destacados.

Robustez: Como a quantidade de vetores de suporte é fisicamente limitada, o d -SVM pode ser mais robusto contra outliers, evitando ajustes excessivos que possam prejudicar a generalização do modelo.

Flexibilidade e Adaptação: O dm -SVM, ao permitir que a quantidade de vetores de suporte varie conforme os gráficos representados na figura 13, apresenta fronteiras de decisão mais complexas e potencialmente mais ajustadas às particularidades dos dados. Isso é especialmente visível nos gráficos onde a fronteira de decisão não é linear e envolve regiões mais sinuosas, o que pode ser benéfico para capturar padrões mais complexos.

Risco de Sobreajuste: Embora o *dm*-SVM ofereça maior flexibilidade, ele também pode estar mais suscetível ao sobreajuste, principalmente em conjuntos de dados ruidosos ou com outliers. A variabilidade na quantidade de vetores de suporte pode levar o modelo a se ajustar demais às características específicas dos dados de treino.

3.4 Conclusão

Nesse capítulo foi visto os modelos d-SVM e dm-SVM e suas respectivas abordagens para a seleção e ajuste dos vetores de suporte. O d-SVM, com sua abordagem mais rígida, utiliza um percentual fixo de vetores de suporte, enquanto o dm-SVM oferece maior flexibilidade ao ajustar dinamicamente a quantidade de vetores de suporte.

4 Resultados e Discussões

Nesta seção são apresentados a metodologia de avaliação de nossa proposta e os resultados obtidos. Os algoritmos de treinamento propostos foram comparados com os algoritmos SMO, SVMQP e com o Kernel Adatron, todos implementados em Python. Os conjuntos de dados utilizados na comparação e suas informações são apresentados na tabela 1

Em cada uma das 10 realizações, o conjunto de dados com todos os padrões é separado aleatoriamente em dois conjuntos de dados: um para treinamento com 70% dos padrões e outro para teste com os 30% restantes. Durante o treinamento, para cada conjunto de treinamento utilizado, foi realizada uma busca em grade com validação cruzada de 5-partes (*folds*) para obter os melhores hiperparâmetros do *kernel* e para o termo de regularização C . Em seguida, com tais hiperparâmetros, treina-se o modelo usando todos os padrões do conjunto de treinamento. Posteriormente, com o modelo treinado na etapa anterior, aplica-se tal modelo ao conjunto de teste para avaliação de desempenho. Podendo ser visto os resultados nas Tabelas 2, 3 e 4

Na versão d-SVM, selecionamos uma porcentagem dos vetores de suporte iniciais, optando por 20% para todos os conjuntos de dados avaliados, com 10% provenientes de cada classe. Por outro lado, na versão dm-SVM, os vetores de suporte são identificados automaticamente pelos métodos descritos nas seções anteriores, eliminando a necessidade de definir um hiperparâmetro, como ocorre no d-SVM. Todos os métodos de treinamento utilizaram os kernels linear, polinomial e RBF, com os seguintes hiperparâmetros: o Parâmetro de Regularização (C), variando entre [0.1, 1, 10]; o Grau Polinomial, com valores de [1, 2, 3]; e o Parâmetro Gamma, com valores de [0.1, 1, 10].

4.1 Hiperparâmetros e seus respectivos valores testados

Tabela 1 – Conjuntos de dados utilizados nas simulações computacionais.

Conjunto de Dados.	Abrev.	Padrões	Atributos
Breast Cancer Wisconsin	BCW	569	30
Heart Disease	HEA	303	76
Iris Flower Dataset	IFD	150	4
Pima Indians Diabetes	PID	768	8
Sonar	SON	208	60
Vertebral Column Pat	VCP	310	6
Wine	WIN	178	13
Artificial	ART	20,000	2
Twitter	TWI	49,999	78

Tabela 2 – Resultados para *kernel* Linear.

Método	Acurácia	Precisão	Recall	Tempo (s)
BCW				
<i>d</i> -SVM	95,5±1,6	95,8±1,6	95,9±1,9	2,4±1,3
<i>dm</i> -SVM	96,5±1,6	96,8±1,6	96,9±2,1	2,1±1,1
Adatron	96,4±1,1	96,5±1,3	97,9±1,2	250,0±150,0
SMO	97,2±1,0	97,0±1,1	98,5±1,0	303,4±185,9
SVMQP	90,0±2,9	91,1±2,8	90,0±2,9	840,7±436,0
VCP				
<i>d</i> -svm	76,2±3,3	88,8±4,5	75,3±6,8	5,3±2,9
<i>dm</i> -SVM	77,3±2,3	88,7±3,4	76,3±3,8	4,3±2,9
Adatron	75,9±3,2	88,5±4,2	75,0±6,5	150,0±85,0
SMO	76,3±3,2	88,9±4,3	75,3±6,8	164,8±90,8
SVMQP	71,5±3,0	97,4±2,1	59,2±4,2	284,4±153,9
PID				
<i>d</i> -SVM	74,1±2,4	86,7±1,8	70,8±3,4	3,3±1,7
<i>dm</i> -SVM	75,2±1,4	87,8±2,1	71,7±2,3	2,3±1,6
Adatron	76,5±2,0	77,8±1,7	88,0±2,1	175,0±100,0
SMO	77,2±1,9	78,3±1,8	89,4±2,0	196,1±108,6
SVMQP	74,2±2,2	86,7±16,7	71,3±3,0	3192,4±1880,2
HEA				
<i>d</i> -SVM	89,2±1,3	85,3±3,1	96,9±2,3	31,5±16,5
<i>dm</i> -SVM	91,3±1,8	86,9±2,1	97,9±2,2	3,4±1,3
Adatron	82,5±1,5	79,2±2,5	89,5±3,5	320,0±180,0
SMO	83,0±1,3	79,7±2,7	90,1±3,8	360,1±192,4
SVMQP	82,5±1,7	78,4±18,6	91,03±2,7	8452,9±891,7
SONAR				
<i>d</i> -SVM	74,4±4,1	70,0±4,5	73,7±7,4	2,0±0,5
<i>dm</i> -SVM	74,2±4,2	71,0±4,7	77,9±7,6	1,0±0,5
Adatron	74,8±5,6	73,5±8,5	72,5±8,0	55,0±30,0
SMO	75,2±5,7	74,1±8,8	73,1±8,2	61,9±33,8
SVMQP	76,1±6,5	72,5±8,5	79,6±59,6	218,3±114,7
WIN				
<i>d</i> -SVM	96,7±1,9	95,6±1,6	95,3±1,4	10,13±5,3
<i>dm</i> -SVM	96,4±1,8	96,6±1,7	96,4±1,8	5,13±2,6
Adatron	99,0±0,9	99,6±0,5	98,8±1,3	9,0±29,0
SMO	99,4±0,89	100,0±0,0	99,1±1,34	11,0±31,7
SVMQP	94,4±2,1	88,8±4,2	99,0±1,91	62,3±39,8
IFD				
<i>d</i> --SVM	100,0±0,0	100,0±0,0	100,0±0,0	2,0±1,1
<i>dm</i> -SVM	100,0±0,0	100,0±0,0	100,0±0,0	1,5±1,2
Adatron	100,0±0,0	100,0±0,0	100,0±0,0	19,0±9,5
SMO	100,0±0,0	100,0±0,0	100,0±0,0	21,2±10,8
SVMQP	99,0±1,8	96,3±5,0	100,0±0,0	57,6±12,7
ART				
<i>d</i> -SVM	99,7±0,04	99,8±0,06	99,7±0,08	1819,1±1002,3
<i>dm</i> -SVM	99,7±0,04	99,8±0,06	99,7±0,08	1819,1±1002,3
Adatron	98,5±0,01	98,7±0,01	98,6±2,5	4500,0±2500,0
SMO	99,0±0,01	99,1±0,01	98,9±2,7	4889,2±2705,0
SVMQP	98,8±0,2	98,9±0,2	98,9±2,3	6417,1±3133,8
TWI				
<i>dm</i> -SVM	91,6±0,1	98,3±9,6	74,0±0,4	6015,1±3294,3
Adatron	89,3±0,1	97,9±9,5	73,8±0,4	7702,7±4189,8
SMO	-	-	-	-
SVMQP	-	-	-	-

Tabela 3 – Resultados para *kernel* Polinomial.

Método	Acurácia	Precisão	Recall	Tempo (s)
BCW				
<i>d</i> -SVM	95,2±1,4	95,8±2,2	95,2±2,6	2,1±0,8
<i>dm</i> -SVM	96.4±1.3	95.9±1.8	96.6±1.1	1.2±1.0
Adatron	96.2±1.3	96.0±1.5	95.8±1.6	55.0±25.0
SMO	96.7±1.2	96.1±2.2	98.7±1.0	181.8±60.2
SVMQP	97.0±1.2	97.8±1.3	97.3±2.1	2956.9±1911.9
VCP				
<i>d</i> -SVM	75.6±2.1	81.2±3.1	74.2±5.2	4.8±2.9
<i>dm</i> -SVM	76.4±2.3	87.8±4.4	75.3±6.7	5.3±2.9
Adatron	75.1±4.6	87.0±5.3	75.0±6.4	70.0±40.0
SMO	75.6±4.5	87.4±5.1	75.7±6.5	79.8±44.0
SVMQP	73.3±4.6	97.5±3.3	61.2±7.7	97.9±56.4
PID				
<i>d</i> -SVM	75.4±1.8	86.5±5.9	64.5±4.5	3.6±2.2
<i>dm</i> -SVM	77.8±2.0	88.4±6.1	69.1±3.9	2.9±1.2
Adatron	76.4±2.4	78.0±2.8	88.5±2.8	65.0±35.0
SMO	76.2±2.4	73.4±7.8	52.3±4.6	402.2±131.7
SVMQP	79.8±3.4	83.4±4.5	88.4±2.1	1385.0±717.7
HEA				
<i>d</i> -SVM	84.7±2.1	80.8±3.4	91.8±1.5	8.2±3.8
<i>dm</i> -SVM	84.6±2.2	80.8±3.4	91.8±1.5	7.9±2.7
Adatron	83.0±2.0	79.9±3.2	90.1±3.6	95.0±50.0
SMO	97.5±1.0	97.4±1.8	97.8±1.2	1148.5±219.9
SVMQP	97.9±0.02	98.8±0.02	97.9±2.3	9417.1±5133.8
SONAR				
<i>dm</i> -SVM	84.9±2.8	85.1±4.3	81.3±7.0	2.5±0.9
<i>dm</i> -SVM	85.5±2.7	86.1±3.9	82.4±6.0	2.3±0.6
Adatron	71.3±7.1	70.8±11.0	71.9±9.7	11.0±7.0
SMO	76.5±3.8	96.3±4.8	51.0±8.4	895.7±606.0
SVMQP	73.5±9.5	72.7±9.9	77.2±12.0	82.2±43.2
WIN				
<i>d</i> -SVM	97.8±2.1	96.5±2.2	94.0±3.5	2.3±0.8
<i>dm</i> -SVM	98.9±2.2	97.6±2.4	94.0±3.7	1.4±0.6
Adatron	98.8±1.2	99.6±0.7	98.6±1.4	8.0±4.5
SMO	96.7±2.8	98.4±2.5	92.6±6.3	26.1±14.4
SVMQP	95.6±1.4	91.5±3.2	99.0±2.4	50.4±25.1
IFD				
<i>d</i> -SVM	100.0±0.01	100.0±0.01	100.0±0.0	1.0±0.3
<i>dm</i> -SVM	100.0±0.01	100.0±0.01	100.0±0.0	1.0±0.2
Adatron	100.0±0.01	100.0±0.01	100.0±0.0	2.0±1.0
SMO	100.0±0.01	100.0±0.01	100.0±0.0	37.2±11.9
SVMQP	98.8±1.8	97.0±5.0	100.0±0.0	357.7±21.2
ART				
<i>d</i> -SVM	99.9±0.02	99.8±0.06	99.7±0.08	18.0±10.3
<i>dm</i> -SVM	99.8±0.04	99.8±0.06	99.7±0.08	12.1±8.3
Adatron	98.5±0.07	98.3±0.09	98.7±0.11	1800.0±1000.0
SMO	98.2±0.3	98.2±1.2	99.3±0.7	54358.3±28362.6
SVMQP	99.0±0.12	98.5±0.24	99.2±0.29	2881.0±1328.9
TWI				
<i>d</i> -SVM	90.1±3.1	83.4±9.6	93.4±2.3	7554.9±554.9
<i>dm</i> -SVM	90.4±4.2	84.4±8.4	94.5±2.1	4352.3±354.4
Adatron	85.2±0.2	98.1±0.05	73.0±0.06	8312.3±2224.1
SMO	-	-	-	-
SVMQP	-	-	-	-

Tabela 4 – Resultados para *kernel* RBF.

Método	Acurácia	Precisão	Recall	Tempo (s)
BCW				
<i>d</i> -SVM	95.9±1.3	96.6±1.2	97.8±1.0	12.7±7.3
<i>dm</i> -SVM	96.8±1.0	96.7±1.4	97.9±1.2	11.2±6.9
Adatron	96.5±1.0	97.0±1.1	98.2±1.1	70.0±40.0
SMO	97.1±0.9	97.2±1.0	98.5±1.0	80.6±44.2
SVMQP	95.8±1.6	95.9±1.9	96.3±1.9	580.5±305.0
VCP				
<i>d</i> -SVM	77.8±3.2	88.6±4.4	76.3±6.7	10.2±5.6
<i>dm</i> -SVM	81.9±3.1	89.6±3.5	77.2±5.7	9.2±4.6
Adatron	76.8±3.4	88.8±4.5	75.0±7.1	100.0±50.0
SMO	77.1±3.3	89.1±4.5	75.5±7.0	110.8±60.8
SVMQP	75.3±3.0	97.5±2.3	62.6±6.1	238.1±134.0
PID				
<i>d</i> -SVM	75.1±1.5	85.5±1.6	74.2±2.2	11.1±6.1
<i>dm</i> -SVM	76.0±2.5	86.5±2.1	73.2±2.3	11.1±6.1
Adatron	77.9±2.2	78.3±2.3	88.8±2.5	125.0±70.0
SMO	78.3±2.0	78.7±2.2	89.5±2.4	134.0±73.8
SVMQP	76.0±2.4	87.5±9.5	73.0±2.7	1672.5±981.4
HEA				
<i>d</i> -SVM	89.2±2.1	87.5±2.1	97.1±2.1	42.7±19.3
<i>dm</i> -SVM	92.3±2.0	88.6±2.2	98.1±2.1	36.6±17.9
Adatron	82.9±1.7	79.5±2.9	89.8±4.0	80.0±21.2
SMO	83.2±1.5	79.8±2.8	90.2±4.2	202.5±108.2
SVMQP	84.0±1.9	79.9±13.1	92.5±2.7	3085.0±610.2
SONAR				
<i>d</i> -SVM	75.0±4.5	70.7±4.4	79.1±7.4	4.4±2.4
<i>dm</i> -SVM	76.1±4.1	71.8±3.9	80.1±6.9	4.1±2.5
Adatron	75.8±5.2	74.7±6.5	73.8±6.3	35.0±20.0
SMO	76.2±5.4	75.1±6.8	74.2±6.5	43.2±23.8
SVMQP	76.5±5.2	73.5±7.0	80.5±7.6	84.4±50.2
WIN				
<i>d</i> -SVM	97.0±2.3	96.5±2.7	97.2±2.4	19.1±9.7
<i>dm</i> -SVM	98.5±2.5	97.7±2.8	97.6±2.2	17.2±7.1
Adatron	99.2±1.0	99.7±0.3	98.9±1.2	40.0±20.0
SMO	99.5±0.89	100.0±0.0	99.2±1.0	45.0±24.5
SVMQP	96.8±1.9	92.5±3.4	99.0±2.5	320.1±190.8
IFD				
<i>d</i> -SVM	100.0±0.01	100.0±0.01	100.0±0.0	4.5±2.3
<i>dm</i> -SVM	100.0±0.01	100.0±0.01	100.0±0.0	4.1±2.4
Adatron	100.0±0.01	100.0±0.01	100.0±0.0	10.0±5.0
SMO	100.0±0.01	100.0±0.01	100.0±0.0	12.5±7.2
SVMQP	100.0±0.01	100.0±0.01	100.0±0.0	28.7±15.4
ART				
<i>d</i> -SVM	99.6±0.09	99.7±0.1	99.7±0.2	4216.3±2324.8
<i>dm</i> -SVM	99.7±0.09	99.8±0.1	99.9±0.1	2216.3±723.8
Adatron	98.9±0.1	99.0±0.1	98.8±0.2	5000.0±2500
SMO	99.1±0.08	99.2±0.09	99.0±0.1	5292.7±2961.8
SVMQP	98.7±0.18	98.9±0.3	98.8±0.4	8436.7±3134.6
TWI				
<i>d</i> -SVM	90.9±0.2	98.2±0.06	73.0±0.07	17847.0±9632.2
<i>dm</i> -SVM	91.3±0.4	98.6±0.08	72.9±0.08	12434.1±2632.2
Adatron	88.2±0.2	88.1±0.09	83.0±0.07	24347.0±13632.2
SMO	-	-	-	-
SVMQP	-	-	-	-

4.2 Análise e Desempenho

Analizamos o desempenho dos métodos d-SVM, dm-SVM, SMO, Adatron e SVMQP utilizando três tipos de kernel: Linear, Polinomial e RBF, com base nas métricas de Acurácia, Precisão, Recall e Tempo de execução. Utilizamos o teste de Friedman para avaliar a eficácia dos métodos com base nas diferentes métricas. Os resultados, apresentados nas Tabelas 2, 3 e 4, mostram que os métodos propostos, d-SVM e dm-SVM, alcançaram acurácia comparável, e em muitos casos equivalente, aos métodos tradicionais, com a vantagem de apresentarem uma eficiência de tempo superior.

4.2.1 Kernel Linear

Os rankings médios obtidos para cada método utilizando o kernel Linear são apresentados a seguir:

Métricas	d-SVM	SMO	Adatron	SVMQP
Acurácia	1.5	5.0	10.0	10.25
Precisão	4.0	5.125	9.0	7.0
Recall	4.5	4.5	10.5	6.625
Tempo	1.0	9.625	6.0	13.0

Tabela 5 – Comparação das métricas entre d-SVM, SMO, Adatron e SVMQP

- **Estatística de Friedman:** 62.32
- **p-valor:** 1.00e-07

A estatística de Friedman resultou em um p-valor extremamente baixo (1.00e-07), indicando que existe uma diferença estatisticamente significativa entre os métodos analisados. Isso significa que os métodos não apresentam o mesmo desempenho, e podemos identificar qual deles é superior com base nos rankings médios.

O método com o menor ranking médio é considerado o mais eficaz. Neste caso, o **d-SVM** destacou-se como o melhor método, apresentando os menores rankings médios em várias métricas, especialmente na Acurácia e no Tempo de execução. Portanto, podemos concluir que, para o kernel Linear, o d-SVM é a escolha mais adequada entre os métodos avaliados, oferecendo um equilíbrio superior entre precisão, recall, e eficiência temporal.

4.2.2 Kernel Polinomial

Os rankings médios obtidos para cada método utilizando o kernel Polinomial são apresentados a seguir:

- **Estatística de Friedman:** 48.11

Métricas	d-SVM	SMO	Adatron	SVMQP
Acurácia	4.625	4.0	7.5	10.625
Precisão	5.0	4.625	8.5	7.0
Recall	4.5	4.5	8.5	8.625
Tempo	1.0	9.5	6.0	13.125

Tabela 6 – Comparação das métricas entre d-SVM, SMO, Adatron e SVMQP

- **p-valor:** 2.44e-05

Novamente, o p-valor significativamente baixo (2.44e-05) indica que há uma diferença estatisticamente significativa entre os métodos avaliados com o kernel Polinomial. Embora o **SMO** tenha apresentado o melhor ranking médio em Acurácia, Precisão, Recall em relação aos outros métodos, o **d-SVM** teve um desempenho competitivo, em relação as mesmas métricas e, demonstrou uma superioridade em custo computacional, tornando-o a escolha mais adequada para este tipo de kernel.

4.2.3 Kernel RBF

Os rankings médios obtidos para cada método utilizando o kernel RBF são apresentados a seguir:

Métricas	d-SVM	SMO	Adatron	SVMQP
Acurácia	1.5	5.0	10.0	10.25
Precisão	4.0	5.125	9.0	7.0
Recall	4.5	4.5	10.5	6.625
Tempo	1.0	9.625	6.0	13.0

Tabela 7 – Comparação das métricas entre d-SVM, SMO, Adatron e SVMQP

- **Estatística de Friedman:** 62.32
- **p-valor:** 1.00e-07

Assim como nos casos anteriores, o p-valor extremamente baixo (1.00e-07) indica uma diferença estatisticamente significativa entre os métodos avaliados utilizando o kernel RBF.

O **d-SVM** novamente demonstrou o melhor desempenho, com os menores rankings médios em várias métricas, particularmente em Acurácia e Tempo de execução. Portanto, para o kernel RBF, o d-SVM é, mais uma vez, a escolha mais apropriada entre os métodos avaliados.

4.3 Análise da segunda Proposta

Analisando o *kernel* linear na Tabela 2, o *dm*-SVM corresponde ao SMO em termos de acurácia nos conjuntos de dados VCP, SONAR e IFD, e apresenta um desempenho superior nos conjuntos HEA e ART. O SMO lidera nos conjuntos de dados BCW, PID e WIN. Em comparação com o SVMQP, o *dm*-SVM é superior ou equivalente em acurácia, mais eficiente e mais rápido no treinamento. Notavelmente, o SMO não conseguiu treinar o conjunto de dados Twitter dentro de duas semanas.

Para o *kernel* polinomial (Tabela 3), o *dm*-SVM geralmente supera ou iguala o SMO e o SVMQP, especialmente no conjunto de dados SONAR. Tanto o SMO quanto o SVMQP não conseguiram treinar o conjunto de dados Twitter no prazo de duas semanas.

Com o *kernel* RBF (Tabela 4), o *dm*-SVM apresenta desempenho superior ou similar em comparação ao SMO e ao SVMQP, alcançando maior acurácia nos conjuntos de dados SONAR, HEA e WIN. O treinamento dos conjuntos de dados ART e Twitter levou mais de duas semanas tanto para o SMO quanto para o SVMQP, sem sucesso. O *dm*-SVM oferece eficiência significativa e tempos de treinamento mais rápidos, conforme detalhado nas Tabelas 2, 3, 4

4.3.1 Kernel Linear

Os rankings médios obtidos para cada método utilizando o kernel Linear são apresentados a seguir:

Métricas	dm-SVM	SMO	Adatron	SVMQP
Acurácia	3.125	4.125	7.625	10.75
Precisão	4.5	3.625	8.5	9.5
Recall	5.5	3.625	8.875	6.5
Tempo	1.625	9.75	5.75	13.25

Tabela 8 – Comparação das métricas entre dm-SVM, SMO, Adatron e SVMQP

- **Estatística de Friedman:** 47.94
- **p-valor:** 2.60e-05

O p-valor baixo (2.60e-05) indica que há uma diferença estatisticamente significativa entre os métodos avaliados. Para o kernel Linear, o **SMO** se destacou como o melhor método em várias métricas, particularmente em Acurácia, Precisão, e Recall, embora o dm-SVM tenha sido o mais rápido.

4.3.2 Kernel Polinomial

Os rankings médios obtidos para cada método utilizando o kernel Polinomial são apresentados a seguir:

Métricas	dm-SVM	SMO	Adatron	SVMQP
Acurácia	2.00	6.625	9.625	8.50
Precisão	4.50	7.125	8.625	6.50
Recall	5.50	6.50	7.00	6.625
Tempo	1.00	10.50	5.75	12.125

Tabela 9 – Comparação das métricas entre dm-SVM, SMO, Adatron e SVMQP

- **Estatística de Friedman:** 43.83
- **p-valor:** 0.000117

O p-valor baixo (0.000117) indica uma diferença estatisticamente significativa entre os métodos avaliados. O **dm-SVM** foi o melhor método em Acurácia e Tempo de execução, destacando-se como a escolha mais eficiente para o kernel Polinomial.

4.3.3 Kernel RBF

Os rankings médios obtidos para cada método utilizando o kernel RBF são apresentados a seguir:

Métricas	dm-SVM	SMO	Adatron	SVMQP
Acurácia	2.14	3.86	8.29	9.14
Precisão	2.14	5.71	9.29	6.29
Recall	1.57	5.14	10.14	8.57
Tempo	1.00	10.14	6.43	13.86

Tabela 10 – Comparação das métricas entre dm-SVM, SMO, Adatron e SVMQP

- **Estatística de Friedman:** 63.37
- **p-valor:** 6.58e-08

O p-valor extremamente baixo (6.58e-08) indica uma diferença estatisticamente significativa entre os métodos analisados. O **dm-SVM** se destacou como o melhor método em todas as métricas para o kernel RBF, tornando-se a escolha mais apropriada. Em conclusão, a análise dos três kernels (Linear, Polinomial e RBF) mostra que o **dm-SVM** é consistentemente eficaz, especialmente no kernel RBF, onde apresentou o melhor desempenho geral. O SMO também se destacou no kernel Linear, demonstrando um bom equilíbrio entre precisão e recall. Dependendo das prioridades específicas (como tempo de execução versus acurácia), ambos os métodos são recomendações sólidas para diferentes cenários.

4.4 Conclusão

Neste capítulo, apresentamos e avaliamos uma nova abordagem para o treinamento de Máquinas de Vetores de Suporte (SVMs). Os métodos propostos, d-SVM e dm-SVM, foram comparados com algoritmos tradicionais como SMO, Adatron e SVMQP, utilizando diferentes tipos de kernel e métricas de desempenho.

Os resultados experimentais demonstraram que os métodos propostos alcançam níveis de acurácia e precisão comparáveis aos métodos tradicionais, com a vantagem de serem mais eficientes em termos de tempo de treinamento.

Utilizando o teste de Friedman, realizamos uma análise estatística para avaliar a significância das diferenças entre os métodos. Os rankings médios obtidos para cada método, utilizando os kernels. Essa análise estatística reforça a evidência de que os métodos propostos são uma alternativa eficiente e eficaz para o treinamento de SVMs.

5 Conclusões e Trabalhos Futuros

Neste trabalho, foram propostos dois modelos de treinamento para máquinas de vetores de suporte (SVMs), com foco na eficiência computacional, especialmente no tempo de treinamento. Em cenários com um grande número de padrões, como ART e TWI, esses métodos de treinamento se destacaram como as opções viáveis para a aplicação de SVMs. A segunda proposta simplifica ainda mais o processo ao automatizar completamente o ajuste dos multiplicadores de Lagrange, reduzindo a necessidade de intervenção manual e aumentando a adaptabilidade do modelo a novos dados ou cenários dinâmicos. Os resultados indicam que essa abordagem mantém níveis competitivos de acurácia e precisão, além de otimizar o tempo de treinamento nos conjuntos de dados testados.

As implicações práticas tem potencial vasto, particularmente em campos que requerem rápida adaptação a novos dados, como sistemas de detecção de fraude, diagnósticos médicos e aprendizado online. A capacidade dos modelos de treinar rapidamente sem sacrificar a precisão o torna ideal para tais aplicações. Em sistemas de detecção de fraude, por exemplo, a velocidade e a precisão no treinamento de são cruciais para identificar e mitigar ameaças em tempo real. Da mesma forma, em diagnósticos médicos, a capacidade de processar grandes volumes de dados com alta dimensionalidade e fornecer resultados precisos em tempo hábil pode ser a diferença entre um diagnóstico precoce e um tardio, impactando diretamente os resultados clínicos. No contexto do aprendizado online, onde os modelos devem se adaptar continuamente a novos dados, a eficiência de treinamento oferecida pela segunda proposta é particularmente valiosa. A capacidade de ajustar automaticamente os multiplicadores de Lagrange elimina a necessidade de reconfigurações frequentes, permitindo que o modelo mantenha um bom desempenho em cenários dinâmicos e em constante mudança.

Referências

- BISHOP, C. M.; NASRABADI, N. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2006. Citado 3 vezes nas páginas 29, 35 e 83.
- CHEN, G.; CHENG, Y.; XU, J. Cluster reduction support vector machine for large-scale data set classification. In: IEEE. *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*. Wuhan, China: IEEE, 2008. v. 1, p. 8–12. Citado na página 31.
- CONOVER, W. *Practical Nonparametric Statistics*. New York: John Wiley & Sons, 1999. Citado na página 90.
- CRISTIANINI, N.; SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000. Citado 2 vezes nas páginas 31 e 40.
- DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 2006. Citado na página 91.
- DIAS, M. L. D.; NETO, A. R. R. Training soft margin support vector machines by simulated annealing: A dual approach. *Expert Systems with Applications*, Elsevier, v. 87, p. 157–169, 2017. Citado 5 vezes nas páginas 31, 41, 45, 46 e 47.
- FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, Taylor & Francis, 1937. Citado na página 89.
- HE, H.; GARCIA, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, 2009. Citado 2 vezes nas páginas 87 e 88.
- JI, W. et al. A review of genetic-based evolutionary algorithms in svm parameters optimization. *Evolutionary Intelligence*, Springer, v. 14, p. 1389–1414, 2021. Citado na página 31.
- KEERTHI, S. S. et al. Improvements to platt’s smo algorithm for svm classifier design. *Neural Computation*, v. 13, n. 3, p. 637–649, 2001. Citado na página 46.
- KREYSZIG, E. *Introductory functional analysis with applications*. New York: Wiley, 1978. Citado na página 83.
- LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado na página 29.
- LI, B.; WANG, Q.; HU, J. A fast SVM training method for very large datasets. In: IEEE. *2009 international joint conference on neural networks*. Atlanta, GA, 2009. p. 1784–1789. Citado na página 31.
- LIMA, E. L. *Espaços Métricos*. Rio de Janeiro, Brasil: Instituto de Matemática Pura e Aplicada, CNPq, 1977. v. 4. 299 p. (Projeto Euclides, v. 4). Digitalizado pela Universidade de Michigan em 5 de fevereiro de 2010. Citado 3 vezes nas páginas 83, 84 e 85.

- MANNING, C. D.; SCHÜTZE, H. *Foundations of Statistical Natural Language Processing*. [S.l.]: MIT Press, 1999. Citado na página 88.
- NALEPA, J.; KAWULOK, M. Selecting training sets for support vector machines: a review. *Artificial Intelligence Review*, Springer, v. 52, n. 2, p. 857–900, 2019. Citado na página 31.
- OSUNA, E.; FREUND, R.; GIROSI, F. Decomposition algorithms for training support vector machines. In: IEEE. *IEEE Neural Networks Council Workshop on Support Vector Machines*. [S.l.], 1997. Citado na página 31.
- PLATT, J. C. Sequential minimal optimization: A fast algorithm for training support vector machines. In: *Advances in kernel methods—support vector learning*. [S.l.: s.n.], 1998. p. 185–208. Citado 3 vezes nas páginas 31, 46 e 51.
- PLATT, J. C. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods*, 1998. Citado 3 vezes nas páginas 31, 50 e 52.
- POWERS, D. M. *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation*. 2011. Citado 2 vezes nas páginas 88 e 89.
- ROCHA NETO, A. R. *SINPATCO II: Novas Estratégias de Aprendizado de Máquina para Classificação de Patologias da Coluna Vertebral*. Tese (Doutorado) — Universidade Federal do Ceará, 2011. Citado 5 vezes nas páginas 37, 39, 41, 47 e 86.
- RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2009. Citado na página 29.
- SCHÖLKOPF, B.; SMOLA, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002. Citado 2 vezes nas páginas 46 e 83.
- SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press, 2014. Citado 4 vezes nas páginas 29, 41, 44 e 47.
- SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, Elsevier, Amsterdam, Netherlands, 2009. Citado 2 vezes nas páginas 87 e 88.
- VAPNIK, V. N. *The Nature of Statistical Learning Theory*. New York: Springer Science & Business Media, 1995. Citado 3 vezes nas páginas 31, 35 e 45.

APÊNDICE A – Espaços de Hilbert

Os *espaços de Hilbert* são um conceito fundamental na matemática. Um espaço de Hilbert é um espaço vetorial com um produto interno que é completo com respeito à norma induzida por esse produto interno. Isso significa que todas as sequências de Cauchy nesse espaço convergem para um ponto dentro do espaço. A completude é uma propriedade crucial que permite a aplicação de várias técnicas analíticas e garante que certos limites e somas infinitas se comportem de maneira esperada. Formalmente, se H é um espaço de Hilbert, então para qualquer sequência de Cauchy $\{x_n\} \subset H$, existe um $x \in H$ tal que $\|x_n - x\| \rightarrow 0$ quando $n \rightarrow \infty$ (KREYSZIG, 1978).

Na área de aprendizado de máquina, os *métodos de kernel* são uma técnica poderosa usada para transformar dados em um espaço de maior dimensão onde eles se tornam linearmente separáveis. Esta técnica é a base do *Support Vector Machine* (SVM) com kernel. Essencialmente, a função de kernel $k(x, y)$ calcula o produto interno no espaço de Hilbert reproduzível correspondente, sem nunca precisar computar explicitamente a transformação para esse espaço de maior dimensão (SCHÖLKOPF; SMOLA, 2002). Isso é conhecido como o truque do kernel. A vantagem dos espaços de Hilbert neste contexto é que eles permitem que as operações de produto interno sejam realizadas de forma eficiente e garantem que as propriedades geométricas e topológicas necessárias sejam preservadas.

A aplicação de *kernels* em SVMs é de extrema importância porque permite que os algoritmos lidem com dados não linearmente separáveis. Em vez de trabalhar diretamente no espaço original dos dados, o kernel mapeia os dados para um espaço de Hilbert de dimensão potencialmente infinita, onde um hiperplano pode ser usado para separar as classes de forma eficiente. Esta abordagem é poderosa porque combina a robustez matemática dos espaços de Hilbert com a capacidade prática de SVMs de encontrar margens de separação ótimas (BISHOP; NASRABADI, 2006). Assim, a compreensão dos espaços de Hilbert e seus operadores lineares é essencial para aproveitar plenamente o potencial dos métodos de kernel em aprendizado de máquina.

Os *espaços de Hilbert* são espaços vetoriais com um produto interno, completos em relação à norma induzida por esse produto interno. Formalmente, um espaço de Hilbert H é um espaço vetorial sobre o campo dos números reais $\mathbb{R}$ ou complexos $\mathbb{C}$ com um produto interno $\langle \cdot, \cdot \rangle : H \times H \rightarrow \mathbb{R}$ (ou $\mathbb{C}$) que satisfaz as seguintes propriedades para todos $x, y, z \in H$ e $\alpha \in \mathbb{R}$ (ou $\mathbb{C}$) (LIMA, 1977):

1. $\langle x, x \rangle \geq 0$ e $\langle x, x \rangle = 0 \Leftrightarrow x = 0$ (positividade definida),
2. $\langle x, y \rangle = \langle y, x \rangle$ (simetria ou hermiticidade),

3. $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$ (linearidade no primeiro argumento).

A completude significa que qualquer sequência de Cauchy em H converge para um elemento dentro de H .

A.1 Propriedade de Continuidade Uniforme

Uma função contínua em um conjunto fechado e limitado é uniformemente contínua (LIMA, 1977).

Seja $f : M \rightarrow \mathbb{R}$ uma função contínua definida em um conjunto fechado e limitado M . Como f é contínua, para um $\epsilon > 0$ arbitrário, existe um $\delta(z)$ para cada $z \in M$ tal que

$$d_2(f(x), f(z)) < \epsilon \text{ se } d_1(x, z) < \delta(z).$$

Podemos cobrir M com um número finito de vizinhanças. Sejam $U_1, \dots, U_m$ vizinhanças com centros $z_1, \dots, z_m$ e raios $\delta(z_1)/2, \dots, \delta(z_m)/2$. Suponha que escolhemos $x, y \in M$ tais que

$$d_1(x, y) < \delta := \frac{1}{2} \min_{1 \leq i \leq m} \delta(z_i).$$

Como x pertence a uma das $U_1, \dots, U_m$, sem perda de generalidade, assumimos que $x \in U_i$. Pela propriedade da distância, temos

$$d_1(x, z_i) < \frac{1}{2} \delta(z_i) < \delta(z_i).$$

$$d_1(y, z_i) \leq d_1(x, y) + d_1(x, z_i) < \delta(z_i).$$

Combinando essas desigualdades e a suposição de que f é contínua, temos

$$d_2(f(x), f(y)) \leq d_2(f(x), f(z_i)) + d_2(f(y), f(z_i)) < \epsilon + \epsilon = 2\epsilon.$$

Como $\epsilon > 0$ é arbitrário e δ não depende de x, y , f é uniformemente contínua.

A.2 Aplicação da Continuidade Uniforme

Podemos provar que "uma integral definida existe para uma função contínua definida sobre um intervalo fechado $[a, b]$ (LIMA, 1977). Se dividirmos $a < b$ em n segmentos de comprimento igual como $a = x_0 < \dots < x_n = b$, então para um $\epsilon > 0$ arbitrário, requeremos

$$\sum_{i=1}^n \left(\sup_{x_{i-1} < x < x_i} f(x) - \inf_{x_{i-1} < x < x_i} f(x) \right) \frac{b-a}{n} < \epsilon$$

para definir a integral definida. Como assumimos que f é uniformemente contínua, a condição $|f(x) - f(y)| < \epsilon/(b-a)$, $x, y \in [x_{i-1}, x_i]$ é satisfeita se fizermos $\delta = x_i - x_{i-1} = \frac{b-a}{n}$ menor (ou seja, aumentarmos n).

A.3 Espaços Lineares e Espaços com Produto Interno

Dizemos que um conjunto V é um espaço linear se satisfaz as seguintes condições: para $x, y \in V$ e $\alpha \in \mathbb{R}$,

1. $x + y \in V$,
2. $\alpha x \in V$.

Se definirmos a soma de $x = [x_1, \dots, x_d]$, $y = [y_1, \dots, y_d] \in \mathbb{R}^d$ e a multiplicação por uma constante $\alpha \in \mathbb{R}$ como $x + y = [x_1 + y_1, \dots, x_d + y_d]$ e $\alpha x = [\alpha x_1, \dots, \alpha x_d]$, então o espaço Euclidiano d -dimensional $\mathbb{R}^d$ forma um espaço linear (LIMA, 1977).

[Espaço L^2] O conjunto $L^2[0, 1]$ de funções $f : [0, 1] \rightarrow \mathbb{R}$ para as quais

$$\int_0^1 f(x)^2 dx < \infty$$

é um espaço linear porque

$$\int_0^1 (f(x) + g(x))^2 dx \leq 2 \int_0^1 f(x)^2 dx + 2 \int_0^1 g(x)^2 dx < \infty$$

e

$$\int_0^1 (\alpha f(x))^2 dx = \alpha^2 \int_0^1 f(x)^2 dx < \infty$$

para $\alpha \in \mathbb{R}$ quando $\int_0^1 f(x)^2 dx < \infty$ e $\int_0^1 g(x)^2 dx < \infty$ (LIMA, 1977).

A.4 Produto Interno em Espaços Lineares

Seja V um espaço linear. Dizemos que qualquer função bivariada $\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{R}$ que obedece às quatro condições abaixo é um produto interno (LIMA, 1977):

1. $\langle x, x \rangle \geq 0$,
2. $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$,
3. $\langle x, y \rangle = \langle y, x \rangle$,
4. $\langle x, x \rangle = 0 \Leftrightarrow x = 0$,

para $x, y, z \in V$ e $\alpha, \beta \in \mathbb{R}$.

A.5 Kernels em Espaços de Hilbert

Utilizando o conceito de espaços de Hilbert, podemos entender e definir *kernels* comuns usados em *Support Vector Machines* (SVMs). Um kernel é uma função que corresponde ao produto interno em um espaço de Hilbert de dimensão possivelmente infinita (ROCHA NETO, 2011). Vamos considerar três tipos de kernels: linear, polinomial e RBF (Radial Basis Function).

A.6 Kernel Linear

O kernel linear é o mais simples e é definido como o produto interno padrão no espaço Euclidiano:

$$k(x, y) = \langle x, y \rangle = x^T y.$$

Aqui, x e y são vetores no espaço $\mathbb{R}^d$. Este kernel é usado quando os dados são aproximadamente linearmente separáveis.

A.7 Kernel Polinomial

O kernel polinomial permite a separação não-linear transformando o espaço de entrada em um espaço de maior dimensão:

$$k(x, y) = (\langle x, y \rangle + c)^d.$$

Nesta fórmula, c é uma constante livre e d é o grau do polinômio. Este kernel é útil para problemas onde a relação entre as características é não-linear.

A.8 Kernel RBF (Radial Basis Function)

O kernel RBF, também conhecido como kernel Gaussiano, é definido como:

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right).$$

Aqui, $\|x - y\|$ é a distância Euclidiana entre os vetores x e y , e σ é um parâmetro livre que controla a largura da gaussiana. Este kernel mapeia os vetores de entrada para um espaço de dimensão infinita, permitindo a separação de classes que não são linearmente separáveis no espaço original.

Espaços de Hilbert fornecem a base matemática necessária para entender kernels em SVMs. A completude dos espaços de Hilbert garante que certas operações matemáticas sejam bem definidas e estáveis. Usando produtos internos em espaços de Hilbert, podemos definir kernels que permitem a separação de dados complexos em um espaço de maior dimensão, facilitando a construção de classificadores eficazes (ROCHA NETO, 2011).

APÊNDICE B – Apêndice B

B.1 Acurácia, Precisão e Recall

Este apêndice apresenta as definições matemáticas das métricas de avaliação de modelos de classificação: Acurácia (*Accuracy*), Precisão (*Precision*) e Recall (*Recall*) (SOKOLOVA; LAPALME, 2009).

1. Acurácia (*Accuracy*)

A **Acurácia** é a métrica que mede a proporção de previsões corretas (tanto de verdadeiros positivos quanto de verdadeiros negativos) em relação ao total de previsões feitas pelo modelo.

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \quad (\text{B.1})$$

Onde:

- TP (True Positives) = Verdadeiros Positivos.
- TN (True Negatives) = Verdadeiros Negativos.
- FP (False Positives) = Falsos Positivos.
- FN (False Negatives) = Falsos Negativos.

Interpretação: A acurácia indica a proporção de todas as previsões que estão corretas. No entanto, em conjuntos de dados desbalanceados, onde uma classe é muito mais frequente que a outra, a acurácia pode ser uma métrica enganosa (HE; GARCIA, 2009).

2. Precisão (*Precision*)

A **Precisão** é a métrica que mede a proporção de exemplos classificados corretamente como positivos em relação ao total de exemplos que foram classificados como positivos (tanto os verdadeiros quanto os falsos positivos).

$$\text{Precisão} = \frac{TP}{TP + FP} \quad (\text{B.2})$$

Onde:

- TP (True Positives) = Verdadeiros Positivos.
- FP (False Positives) = Falsos Positivos.

Interpretação: A precisão mostra a qualidade das previsões positivas do modelo. Uma alta precisão indica que a maioria das instâncias classificadas como positivas são realmente positivas (POWERS, 2011).

3. Recall (*Recall*)

A **Recall** (também conhecida como sensibilidade ou taxa de verdadeiros positivos) é a métrica que mede a proporção de exemplos positivos que foram corretamente identificados pelo modelo em relação ao total de exemplos que realmente são positivos.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (\text{B.3})$$

Onde:

- TP (True Positives) = Verdadeiros Positivos.
- FN (False Negatives) = Falsos Negativos.

Interpretação: A Recall reflete a capacidade do modelo de capturar todos os exemplos positivos. Uma alta Recall significa que o modelo é eficiente em identificar a maioria dos exemplos positivos (MANNING; SCHÜTZE, 1999).

4. Relação entre Precisão, Recall e Acurácia

4.1. Cenário de Dados Balanceados

Quando os dados são balanceados (ou seja, o número de exemplos positivos e negativos é aproximadamente igual), a acurácia pode ser uma métrica confiável. Se a precisão e a Recall forem altas, a acurácia também tende a ser alta (SOKOLOVA; LAPALME, 2009).

4.2. Cenário de Dados Desbalanceados

Em casos de dados desbalanceados, a acurácia pode não refletir bem o desempenho do modelo, sendo necessário focar na precisão e na Recall (HE; GARCIA, 2009).

4.3. Trade-off entre Precisão e Recall

Existe um trade-off entre precisão e Recall, e ajustar o limiar de decisão do modelo pode aumentar uma métrica à custa da outra. Para balancear as duas métricas, o F1-score é frequentemente utilizado:

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \quad (\text{B.4})$$

O F1-score é a média harmônica entre precisão e Recall, sendo útil quando é necessário encontrar um equilíbrio entre essas duas métricas (POWERS, 2011).

B.2 Demonstração do Teste de Friedman

O teste de Friedman é um teste estatístico não paramétrico utilizado para comparar três ou mais grupos emparelhados. Ele é particularmente útil em situações onde os pressupostos do teste ANOVA para medidas repetidas (normalidade e homogeneidade das variâncias) não são atendidos (FRIEDMAN, 1937).

Abaixo, apresento a demonstração matemática do teste de Friedman.

B.3 Definições e Notações

Considere que temos k tratamentos (ou grupos) e n blocos (ou sujeitos). Cada bloco contém uma medida correspondente a cada tratamento.

- x_{ij} é o valor da i -ésima observação do j -ésimo tratamento.
- R_{ij} é o posto (rank) de x_{ij} dentro do bloco i , entre os k tratamentos.
- $\bar{R}_j$ é a média dos postos do tratamento j .

B.4 Passos do Teste de Friedman

B.5 Calculando os Postos

Dentro de cada bloco i , ordenamos as observações x_{ij} para $j = 1, 2, \dots, k$, e atribuímos postos R_{ij} . Se houver empates, o posto médio é atribuído.

B.6 Soma dos Postos por Tratamento

A soma dos postos para o j -ésimo tratamento é dada por:

$$R_j = \sum_{i=1}^n R_{ij}$$

B.7 Estatística de Teste de Friedman

A estatística de teste de Friedman Q é dada por:

$$Q = \frac{12n}{k(k+1)} \sum_{j=1}^k \left(\bar{R}_j - \frac{k+1}{2} \right)^2$$

onde $\bar{R}_j = \frac{R_j}{n}$ é a média dos postos para o tratamento j .

Expansão da soma dos quadrados:

$$Q = \frac{12}{nk(k+1)} \sum_{j=1}^k R_j^2 - \frac{3n(k+1)}{2}$$

Essa fórmula é usada para calcular a estatística do teste a partir das somas dos postos.

B.8 Distribuição da Estatística de Teste

Para $k \geq 5$, Q aproximadamente segue uma distribuição qui-quadrado (χ^2) com $k - 1$ graus de liberdade. O valor-p é determinado a partir dessa distribuição, comparando Q com o valor crítico correspondente.

Se o valor-p for menor que o nível de significância α (geralmente 0,05), rejeitamos a hipótese nula (CONOVER, 1999).

B.9 Hipóteses

- **Hipótese nula (H_0):** Os k tratamentos têm o mesmo efeito, ou seja, as distribuições de R_{ij} são idênticas para todos os tratamentos.
- **Hipótese alternativa (H_1):** Pelo menos um dos tratamentos tem um efeito diferente, ou seja, as distribuições de R_{ij} diferem para pelo menos um tratamento.

B.10 Considerações

- O teste de Friedman é aplicável em situações onde temos blocos emparelhados e queremos testar a diferença entre mais de dois grupos.

-
- Se o teste de Friedman indica diferenças significativas, testes post-hoc, como o teste de Wilcoxon pareado, podem ser usados para comparações múltiplas entre os tratamentos (DEMŠAR, 2006).