UNIVERSIDADE FEDERAL DE ALAGOAS
CAMPUS A. C. SIMÕES
INSTITUTO DE COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

JOSE AUGUSTO DOS SANTOS SILVA

OTIMIZANDO FUNÇÕES DE *KERNEL* RACIONAIS PARA MÁQUINAS DE VETORES DE SUPORTE

Maceió
2023

JOSE AUGUSTO DOS SANTOS SILVA

OTIMIZANDO FUNÇÕES DE *KERNEL* RACIONAIS PARA MÁQUINAS DE VETORES DE SUPORTE

Trabalho de Conclusão de Curso de Graduação em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas como requisito para a obtenção do título de Engenheiro de Computação.
Orientador: Prof. Erick de Andrade Barboza, Dr.

Maceió
2023

Catalogação na fonte
Universidade Federal de Alagoas
Divisão de Tratamento Técnico
Biblioteca Central
Bibliotecário: Jorge Raimundo da Silva – CRB-4 - 1528

S586o Silva, José Augusto dos Santos

Otimizando funções de Kernel racionais para máquinas de vetores de suporte. / José Augusto dos Santos Silva. – 2023.
59 f. : il., grafs., tabs.

Orientador: Erick de Andrade Barboza.
Monografia (Trabalho de Conclusão de Curso em Engenharia de computação) – Universidade Federal de Alagoas. Instituto de Computação, Maceió. 2023.

Bibliografia: f. 53-59.

1. Máquinas de Vetores de Suporte. 2. Funções de Kernel. 3. Funções Racionais. 4. Evolução Diferencial. 5. Aprendizado de Máquina Automatizado. I. Título.

CDU: 004.383.8

Este trabalho é dedicado integralmente à minha família e,
especialmente, aos meus pais. É graças aos seus esforços,
sacrifícios e incentivos que hoje concluo o meu curso.

AGRADECIMENTOS

À minha família e aos meus queridos pais, cujo amor incondicional, apoio incessante e sacrifícios incalculáveis foram alicerces sólidos na construção de cada página deste trabalho e, indubitavelmente, de cada passo da minha jornada. Suas lições de vida e valores foram a luz orientadora nos momentos de incerteza e desafio.

Aos meus inestimáveis amigos, companheiros leais, que com sua amizade genuína, compreensão e apoio inabalável, tornaram a travessia acadêmica mais leve. Cada alegria vivida, cada angústia compartilhada e cada pequeno gesto de solidariedade foram imprescindíveis para que este dia chegasse.

Aos meus colegas de curso, Pareias do IC, com quem convivi durante os últimos anos, pela compreensão, companhia e pela troca de experiências que me permitiram crescer não só como pessoa, mas também como formando.

Aos meus professores, em especial ao meu orientador e à banca examinadora, mentores dedicados, que com paciência, conhecimento e compromisso, não apenas transmitiram a matéria de seus respectivos campos, mas instigaram em mim a paixão pelo saber, o respeito pela ciência e o compromisso com a excelência.

À Universidade Federal de Alagoas, instituição pública e de qualidade, essencial no meu processo de formação profissional e acadêmica, expresso minha gratidão pela dedicação e por tudo o que aprendi ao longo dos anos do curso.

Este trabalho é um tributo à contribuição que vocês generosamente me ofereceram. Que estas páginas reflitam não apenas o conhecimento adquirido, mas principalmente o caráter, a resiliência e a gratidão cultivados através do convívio e do exemplo inspirador de cada um de vocês.

"A ciência é mais do que um corpo de conhecimento. É uma forma de pensar." (Sagan, 1996)

RESUMO

Aprendizado de máquina, um subconjunto da inteligência artificial, foca no desenvolvimento de modelos matemáticos que permitem que computadores aprendam e tomem decisões a partir do uso de dados. Em vez de serem explicitamente programados para executar uma tarefa, esses modelos usam métodos estatísticos para identificar padrões a partir dos dados e fazer predições. No centro deste paradigma está o desenvolvimento de algoritmos capazes de modelar relações não lineares pelo uso dos dados. Um desses proeminentes modelos são as Máquinas de Vetores de Suporte, com raízes na teoria do aprendizado estatístico, que são modelos de aprendizagem supervisionada que se destacam em tarefas de classificação ao encontrar o melhor hiperplano que separa as classes distintas de um conjunto de dados rotulado. Máquinas de Vetores de Suporte têm notória popularidade em diferentes cenários devido a sua habilidade de generalização e de lidar com funções de decisão não lineares. Um dos fatores críticos que influenciam a sua função de decisão para lidar com problemas não lineares é a escolha da função de *kernel*, que representa o resultado do produto interno no espaço de alta dimensão onde os dados são implicitamente transformados para que modelos lineares possam encontrar funções de decisão não lineares. Enquanto o *kernel* de Função de Base Radial tem sido uma escolha tradicional e amplamente utilizada, este trabalho explora a viabilidade de funções racionais, que são funções definidas pelo quociente entre dois polinômios, como funções de *kernel*, otimizadas pelo uso da Evolução Diferencial. Deste modo, o objetivo principal deste trabalho é construir uma metodologia de aprendizado automatizado de funções de *kernel* modeladas por funções racionais para investigar a viabilidade de obter desempenho comparável em relação ao *kernel* de Função de Base Radial, uma das funções de *kernel* mais utilizadas, ao passo que podem oferecer vantagens em termos de convergência, complexidade e capacidade de generalização. Os experimentos foram conduzidos utilizando o conjunto de dados Pima e os resultados mostram que as funções de *kernel* racionais foram capazes de alcançar acurácia balanceada estatisticamente igual ao *kernel* de Função de Base Radial. Esse resultado demonstra o potencial de funções racionais, de coeficientes otimizados pela Evolução Diferencial, como alternativa viável para o *kernel* de Máquinas de Vetores de Suporte. Além disso, as funções racionais como *kernel* utilizaram um número menor de iterações para convergir (84,14% menor) e um número menor de vetores de suporte (3,84% menor), o que indica um modelo menos complexo. Os resultados foram comparados por meio de testes estatísticos com nível de confiança igual a 95% e indicam que ainda há espaço na literatura para explorar aprimoramentos no uso de funções de *kernel* em Máquinas de Vetores de Suporte.

Palavras-chave: Máquinas de Vetores de Suporte; Funções de *Kernel*; Funções Racionais; Evolução Diferencial; Aprendizado de Máquina Automatizado.

ABSTRACT

Machine learning, a subset of artificial intelligence, focuses on the development of mathematical models that allow computers to learn and make decisions from data. Instead of being explicitly programmed to perform a task, these models employ statistical methods to identify patterns from data and make predictions. At the heart of this paradigm is the development of algorithms capable of modeling non-linear relationships through data. One of these prominent models is the Support Vector Machines, with roots in statistical learning theory, which are supervised learning models that excel in classification tasks by finding the best hyperplane that separates the distinct classes of a labeled dataset. Support Vector Machines have notable popularity in various scenarios due to their generalization ability and handling non-linear decision functions. One of the critical factors influencing its decision function to deal with non-linear problems is the choice of the kernel function, which represents the outcome of the inner product in the high-dimensional space where data is implicitly transformed so that linear models can find non-linear decision functions. While the Radial Basis Function kernel has been a traditional choice and widely used, this work explores the feasibility of rational functions, which are functions defined by the quotient between two polynomials, as kernel functions, optimized by the use of Differential Evolution. Thus, the main objective of this work is to build an automated learning methodology for rational functions as kernel functions to investigate the feasibility of achieving comparable performance to the Radial Basis Function kernel, one of the most used kernel functions, while possibly offering advantages in terms of convergence, complexity and generalization ability. Experiments were conducted using the Pima dataset, and the results show that the rational kernel functions were able to achieve balanced accuracy statistically equal to the Radial Basis Function kernel. This outcome demonstrates the potential of rational functions, with coefficients optimized by Differential Evolution, as a viable alternative for the Support Vector Machines kernel. Moreover, rational functions as a kernel used fewer iterations to converge (84.14% lower) and fewer support vectors (3.84% lower), indicating a less complex model. The results were compared through statistical tests with a confidence level equal to 95%, and they indicate that there is still room in the literature to explore improvements in the use of kernel functions in Support Vector Machines.

Keywords: Support Vector Machines; Kernel Functions; Rational Functions; Differential Evolution; Automated Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de máquina
BAS	Pontuação de acurácia balanceada
DE	Evolução diferencial
EA	Algoritmo evolucionário
EDA	Algoritmo de estimação de distribuição
EPT	Espessura da pele do tríceps
ERD	Estimativa de risco à diabetes baseada no histórico familiar
FN	Falso negativo
FP	Falso positivo
GA	Algoritmo genético
GP	Programação genética
Id	Idade
IMC	Índice de massa corpora
IS	Insulina sérica
ITR	Número de iterações
KKT	Karush-kuhn-tucker
MSE	Erro médio quadrático
NG	Número de gestações
NIDDK	Instituto Nacional de Diabetes e Doenças Digestivas e Renais
NSV	Número de vetores de suporte
PDM	Processo de decisão de markov
PSD	Pressão sangúinea diastólica
PSO	Otimização por enxame de partículas
RBF	Função de base radial
RKF	Função de *kernel* racional
SA	Recozimento simulado
SGD	Gradiente descendente estocástico
SMO	Otimização sequencial mínima
SV	vetores de suporte
SVM	Máquina de vetores de suporte
TN	Verdadeiro negativo
TOTG	Teste oral de tolerância à glicose
TP	Verdadeiro positivo
TVN	Taxa de verdadeiro negativo
TVP	Taxa de verdadeiro positivo

LISTA DE SÍMBOLOS

α	Multiplicador de Lagrange
C	Coeficiente de regularização da SVM
γ	Argumento para o *kernel* RBF
$\mathbb{R}$	Conjunto dos números reais
K	Função de *kernel*
κ	Argumento para o *kernel* sigmoide
k	Quantidade de conjuntos na validação cruzada
μ	Média de uma população
σ	Desvio padrão de uma população
w	Vetor de pesos da SVM
b	Termo viés da SVM
$\mathbb{D}$	Conjunto de dados
ρ	Distância do vetor mais próximo ao hiperplano
ξ	Variáveis de folga
L	Função Lagrangiana
β	Multiplicador de Lagrange
ϕ	Mapeamento de um vetor para um espaço de dimensão mais alta
Q	Matriz de Gram
M	Margem completa da SVM
e	Número de Euler
ϵ	Representação de uma pequena distância
$\mathbb{P}$	Conjunto de indivíduos de um algoritmo evolucionário
P	Número de indivíduos de um algoritmo evolucionário
A	Operador de aptidão
R	Operador de recombinação
T	Operador de mutação
N	Número de gerações de um algoritmo evolucionário
F	Fator de escalada da DE
CR	Taxa de recombinação da DE
G	Número de coeficientes do polinômio

SUMÁRIO

1 INTRODUÇÃO

Aprendizado de Máquina (AM) se tornou inevitável em aplicações na ciência e na indústria, incluindo processamento de linguagem natural, visão computacional, medicina e finanças, dentre outros. No conjunto dos algoritmos de AM, Máquinas de Vetores de Suporte (SVMs) são reconhecidas pela sua robustez e versatilidade em resolver tanto problemas de classificação quanto problemas de regressão (VAPNIK, 1998). A fundação das SVMs está na teoria do aprendizado estatístico e, essencialmente, tem como objetivo encontrar o hiperplano ótimo que segrega os dados de classes distintas. Essa separação geralmente é resolvida em um espaço de características de dimensão mais alta, implicitamente mapeado pelo uso de uma função de *kernel* (SCHÖLKOPF; SMOLA, Alexander J, 2002).

As funções de *kernel*, com seu papel central na desempenho das SVMs, ditam o quão bem os modelos pode ser aprimorados pela transformação implícita dos dados para um espaço onde são linearmente separáveis. Neste contexto, as funções de *kernel* clássicas utilizadas, como a linear, sigmoide, polinomial e a Função de Base Radial (RBF), têm sido investigadas e amplamente utilizadas (CRISTIANINI; SHAWE-TAYLOR, 2000; SCHÖLKOPF; SMOLA, Alexander J, 2002). Apesar da eficácia em várias aplicações, não são universalmente adequadas na hipótese de que os dados são linearmente separáveis em outro espaço de características, particularmente quando o conjunto de dados contém relações não lineares no seu espaço de características original, exigindo um *kernel* mais flexível que possa se adaptar à estrutura intrínseca dos dados.

Pesquisas sobre teoria de aproximação e otimização matemática se mostram relevantes na exploração de novas funções que possam servir como alternativas para funções de *kernel* tradicionais nas SVMs. Uma das famílias de interesse é a família das funções racionais. Essas funções são definidas como o quociente de dois polinômios e têm sido promissoras ao se disporem como um mecanismo alternativo que oferece flexibilidade na representação de relacionamentos funcionais complexos (PARKS; BURRUS, 1987; BULTHEEL; VAN BAREL *et al.*, 2004).

Funções racionais, sendo uma razão de funções polinomiais, podem ser úteis ao usar sua flexibilidade para aprender mapeamento de dados em altas dimensões. O poder das funções racionais está na sua habilidade de aproximar uma ampla gama de funções e, consequentemente, lidar com padrões de dados que podem ser intangíveis por funções tradicionais (HU; TAO; CROITORU, 2004). Apesar disso, o uso de funções racionais pode introduzir complexidade adicional em termos de otimização, pois as funções são definidas por coeficientes que precisam ser apropriadamente selecionados para atingir um desempenho adequado.

Para lidar com o caótico cenário de otimização associado às funções de *kernel* racionais, este trabalho emprega o uso da Evolução Diferencial (DE), um algoritmo evo-

lucionário (EA) robusto e eficiente (STORN, R.; PRICE, K., 1997). DE é reconhecido no campo da otimização pela sua capacidade de exploração em espaços complexos de otimização, multimodais e de alta dimensão, que são comumente encontrados em cenários do mundo real (DAS *et al.*, 2011). Suas vantagens incluem poucos hiperparâmetros, operações simples e habilidade de escapar de mínimos locais, fazendo com que sejam uma opção factível para a otimização dos coeficientes de funções racionais.

Uma consideração crucial nos EAs é o design de uma função de aptidão compatível para guiar a otimização. No contexto da otimização de funções de *kernel* para SVMs, a escolha mais natural seria a combinação de métricas intrínsecas das SVMs e métricas de AM, como acurácia, especificidade, sensitividade, entre outras (SOKOLOVA; LAPALME, 2009). Porém, sem a correta calibração, a função de aptidão pode não capturar exatamente a performance da função de *kernel*. A função de aptidão deve ser, portanto, uma ponderação entre métricas de performance de AM e do próprio modelo SVM, de maneira a refletir a capacidade real da função de *kernel* racional.

1.1 TRABALHOS RELACIONADOS

O emprego de técnicas e métodos de otimização para a escolha de funções de *kernel* em SVMs não é uma área de pesquisa recente. Particularmente, existe uma extensa literatura científica sobre a matéria, que também incluí tanto o desenvolvimento de metodologias de otimização automática para o melhoramento de funções de *kernel* já existentes, como o desenvolvimento de modelagens de novas funções e seleção automática destas funções.

Ayat, Cheriet e Suen (AYAT; CHERIET; SUEN, 2005) propuseram uma metodologia de aprendizagem automática dos parâmetros de uma função de *kernel* para reduzir o número de vetores de suporte, o que teoricamente implica a redução do erro de generalização do modelo. O critério de seleção foi baseado em uma estimativa da probabilidade de erro do classificador SVM e, para comparação, utilizaram *Generalized Approximate Cross-Validation* e a dimensão Vapnik-Chernovenkis, que são limitantes superiores para o erro esperado. Os autores também propuseram um esquema de minimização baseado em gradientes pra encontrar os melhores parâmetros para as funções de *kernel*, testaram a metodologia tanto em dados sintéticos como dados reais e mostraram que sua abordagem conseguia modelos menos complexos com uma quantidade menor de vetores de suporte.

Howley e Madden (HOWLEY; MADDEN, 2006) elaboraram uma abordagem evolucionária guiada por dados para automaticamente construir funções de *kernel*, denominando-a *KTree*, que é a representação de um *kernel* em árvore. Ao caminhar na árvore é possível compor outras funções de *kernel* ou operadores matemáticos, tornando possível o aprendizado de uma função adequada para o problema atacado. Diferentes funções de aptidão foram testadas, combinando o erro de classificação do conjunto de treino, o tamanho da árvore, a soma dos multiplicadores (α_i) dos vetores de suporte ou validação cruzada com

três subconjuntos. A abordagem foi testada em dados sintéticos e em dados reais, mostrando capacidade de superar ou manter a performance em comparação com as funções de *kernel* tradicionais.

Koch e seus colegas (KOCH *et al.*, 2012) propuseram uma forma de evoluir funções de *kernel* baseada em Programação Genética (GP). A representação também é em árvore e a otimização usa as propriedades do *kernel*: fechada para soma, fechada para multiplicação, fechada para a multiplicação por um escalar positivo e fechada para a exponenciação. A GP empregada, então, busca árvores de expressões que utilizam as propriedades de fechamento para combinar operações e tem profundidade máxima de dez níveis. São definidos critérios de aceitação para as novas funções, que devem ter dois vetores de entrada, no máximo quatro constantes multiplicativas e a matriz de Gram associada não deve conceber problemas numéricos. Os autores desenvolveram novos métodos de mutação e recombinação, e a função de aptidão é baseada nos critérios de aceitação definidos associados com a performance em um conjunto de teste. No entanto, concluem que não conseguiram encontrar melhorias significativas em comparação a funções de *kernel* tradicionais.

Dioşan, Rogozan e Pecuchet (DIOŞAN; ROGOZAN; PECUCHET, 2012) também propuseram uma abordagem de GP para o aprendizado de novas funções de *kernel* e chamaram sua abordagem de *eKoK* (*Evolutionary Kernel of Kernels*). Um *eKoK* é uma função de *kernel* que respeita as propriedades do teorema de Mercer e também é representada em forma de árvore, onde os autores desenvolvem métodos de mutação e recombinação próprios que preservam as estruturas de árvore e os critérios do teorema de Mercer. Para a função de aptidão, os autores separaram os dados em 80% para treino e 20% pra teste. O conjunto de treino é dividido utilizando o método *holdout* de um terço e a métrica de desempenho utilizada na função de aptidão é a acurácia. Nas conclusões, os autores afirmam que a sua abordagem é mais performática que as funções de *kernel* clássicas, apesar de ter um custo computacionalmente maior.

Já no trabalho de Mantovani e colegas (MANTOVANI *et al.*, 2015), investiga-se a hipótese de que uma simples busca randômica seja suficiente para otimizar os parâmetros C da SVM e γ da RBF. Para os experimentos, os autores utilizaram, além da busca randômica, a busca em grade, Algoritmo Genético (GA), Otimização por Enxame de Partículas (PSO) e Algoritmos de Estimação de Distribuição (EDA) na otimização dos parâmetros, aplicando em 70 conjuntos de dados distintos. Os resultados experimentais do *kernel* RBF corroboram com a hipótese de que a busca randômica é capaz de produzir modelos com capacidade preditiva equivalente aos modelos otimizados pelas meta-heurísticas.

Syarif, Prugel-Bennett e Wills (SYARIF; PRUGEL-BENNETT; WILLS, 2016), em seu trabalho, comparam o custo computacional do uso da busca em grade em comparação com o uso de meta-heurísticas (PSO e GA) para otimizar os parâmetros das funções de *kernel* clássicas. Os autores fizeram os experimentos em nove conjuntos de dados distintos,

e concluem que a busca em grade para otimizar os melhores parâmetros é viável para conjuntos de dados de dimensionalidade baixa com poucos parâmetros, porém os seu custo computacional não escala bem quando a quantidade de parâmetros aumenta ou a dimensionalidade do conjunto de dados cresce. Ao contrário, o uso de meta-heurísticas se mostrou mais estável e cerca de 16 vezes mais rápido. Os autores ainda concluem que os resultados de performance obtidos com o uso das meta-heurísticas foram ligeiramente superiores aos resultados encontrados na busca em grade.

Yu e Wang (YU; WANG, 2017) abordam a otimização dos parâmetros de funções de *kernel* utilizando uma versão da DE modificada proposta pelos próprios autores. Duas diferentes estratégias de construção dos vetores de teste foram integradas na modificação, e foram utilizados dez conjuntos de dados do mundo real para realizar os experimentos. Antes de executar a otimização, os autores escolhem uma função de *kernel* dentre as funções tradicionais. Depois, o algoritmo modificado otimiza os parâmetros da função de *kernel* escolhida. Os autores ainda comparam os resultados obtidos com os resultados obtidos por Tian e seus colegas (TIAN *et al.*, 2013), mostrando que sua abordagem foi capaz de obter melhores resultados em sete dos dez conjuntos de dados.

Roman, Santana e Mendiburu (ROMAN *et al.*, 2021) fazem um trabalho minucioso ao fazer uma análise do conjunto de componentes que influenciam o comportamento das SVMs e sua interação com a função de *kernel*. Além de proporem uma estrutura gramática própria para a busca em árvore aliada a GP para construir novos *kernels*, os autores introduzem uma nova métrica para abordar o problema da acurácia não incluir informação sobre a generalização do modelo e exigir o uso da validação cruzada. A nova métrica é baseada no critério de informação bayesiano, é contínua e penaliza o *kernel* pela sua quantidade de parâmetros. Concluem que o uso da GP provê resultados marginalmente melhores, identificam a necessidade de estender os conjuntos de dados referência para aumentar a variedade de características e mostram que uma gramática bem definida para a busca pode ser mais importante que o método de busca em si.

Do exposto, nota-se o uso extensivo de técnicas de otimização para melhorar a performance de funções de *kernel* tradicionais ou para encontrar novas funções de *kernel* mais adequadas aos diversos cenários de dados, ainda que a maioria das abordagens utilizem estruturas de árvore e gramáticas construtivas para funções de *kernel*. Contudo, percebe-se que há espaço para novas abordagens e melhorias dos métodos de aprendizagem automática de funções de *kernel*.

Sejam $x \in \mathbb{R}$ um número real e $x_i, x_j \in \mathbb{R}^d$, isto é, vetores de dimensão d com entradas reais. Com uma simples análise morfológica das funções de *kernel* tradicionais, percebe-se que podem ser vistas como uma composição de duas funções, $f(x)$ e $g(x_i, x_j)$, onde $g(x_i, x_j)$ é uma função real de duas variáveis que assume o produto interno euclidiano ou a distância euclidiana e $f(x)$ é uma função real de uma variável:

- Linear: $K(x_i, x_j) = x_i \cdot x_j$, com $f(x) = x$ e $g(x_i, x_j) = x_i \cdot x_j$;

- Polinomial: $K(x_i, x_j) = (x_i \cdot x_j + r)^d$, com $f(x) = (x + r)^d$ e $g(x_i, x_j) = x_i \cdot x_j$;
- RBF: $K(x_i, x_j) = exp(-\gamma \|x_j - x_i\|^2)$ para $\gamma > 0$, com $f(x) = exp(-\gamma x^2)$ e $g(x_i, x_j) = \|x_j - x_i\|$;
- Sigmoide: $K(x_i, x_j) = \tanh(\kappa x_i \cdot x_j + c)$ para alguns $\kappa > 0$ e $c < 0$, com $f(x) = \tanh(\kappa x + c)$ e $g(x_i, x_j) = x_i \cdot x_j$.

A análise morfológica permite perceber que a função $f(x)$ é passível à novas formulações, já que o produto interno ou a distância euclidiana estão dispostos como operação básica entre os dois vetores. O presente trabalho, então, lida primariamente com o uso das funções racionais como modelo para $f(x)$, que terá seus coeficientes aprendidos pela DE. A função $g(x_i, x_j)$ escolhida é a distância euclidiana por ser a função utilizada no *kernel* RBF, função de *kernel* mais popular entre os utilizadores das SVMs e que será utilizada como base de comparação para verificar a viabilidade do modelo idealizado.

A viabilidade das funções racionais como *kernel* de SVMs podem, portanto, ser uma área de interesse na comunidade do AM. Se por um lado a flexibilidade intrínseca das funções racionais oferece um caminho para capturar funções de decisão complexas e não lineares, por outro lado a escolha do *kernel* tem implicações substanciais para a capacidade de generalização da SVM, especialmente quando os coeficientes da função racional $f(x)$ modelada são otimizados utilizando meta-heurísticas, como a DE.

Embora a DE seja capaz de explorar o espaço de busca para encontrar ótimos globais ou locais, o potencial de sobreajuste aumenta ao usar *kernels* altamente flexíveis, como os construídos a partir de funções racionais. Entretanto, com uma regularização criteriosa e validação cruzada, é concebível que funções de *kernel* racionais, otimizados por DE, possam produzir resultados quem ampliem o entendimento em certas aplicações. Para isso, experimentos controlados são necessários para afirmar conclusivamente o seu potencial em conjuntos de dados e domínios de problemas variados.

1.2 OBJETIVOS

O objetivo primário deste trabalho é desenvolver uma nova metodologia de busca automática de novas funções de *kernel* para SVMs baseadas em funções racionais pela utilização da DE na otimização dos coeficientes de seus polinômios. Métodos tradicionais de otimização como busca em grade ou busca aleatória manifestam limitações na sua habilidade de efetivamente explorar o complexo espaço de parâmetros das funções de *kernel*. A DE, em contra partida, têm mostrado seu valor em vários desafios de otimização dentre vários domínios (STORN, R.; PRICE, K., 1997; DAS *et al.*, 2011).

Além do objetivo principal, este trabalho visa também atingir metas secundárias, que são listadas como se segue:

- Executar simulações e experimentos que avaliam o potencial das novas funções de *kernel* sob vários hiperparâmetros;

- Desenvolver uma função de aptidão para medir a performance de várias funções de *kernel* e guiar a otimização;
- Aplicar a metodologia para fazer um estudo de caso com dados reais, analisando as propriedades do resultado.

Por fim, e não menos importante, o trabalho também tem como objetivo demonstrar o aprendizado ao longo do curso e sua capacidade de aplicação em situações reais.

1.3 CONTRIBUIÇÕES

As contribuições deste trabalho visam introduzir o uso de funções racionais como classe viável de funções de *kernel* para SVMs, expandindo a caixa de ferramentas disponível tanto para pesquisadores quanto para profissionais da área. Também visa adotar o algoritmo DE como alternativa para o ajuste fino dos coeficientes das funções de *kernel* racionais propostas, fornecendo uma estratégia de otimização robusta e eficiente comparado com a busca em grade ou busca randômica para funções de *kernel* tradicionais. Além do mais, também contribui na pluralidade das funções de aptidão que possam capturar aspectos relevantes da performance de funções de *kernel* sobre aplicações de otimização envolvendo SVMs.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho é dividido da seguinte maneira: o Capítulo 2 apresenta os fundamentos teóricos que servirão de base para o entendimento da proposta deste trabalho. Capítulo 3 servirá para o detalhamento da proposta, com características específicas de implementação e conjunto de dados utilizado. Capítulo 4 apresenta os resultados da abordagem deste trabalho. Por fim, o Capítulo 5 será responsável por apresentar o que é possível concluir da abordagem proposta e suas limitações, bem como os trabalhos que podem ser explorados em ocasiões futuras.

2 BASE TEÓRICA

O campo do Aprendizado de Máquina (AM) continua a se diversificar e a se aprofundar, alimentado pelo fluxo constante de dados complexos e pela procura de algoritmos inteligentes capazes de os decifrar. No centro deste paradigma está a ciência de construir modelos que podem generalizar a partir do passado para prever o futuro. Um desses modelos que tem mantido a sua importância na comunidade científica é a Máquina de Vetores de Suporte (SVM).

Como acontece em muitas áreas de aprendizado de máquina, as SVMs apresentam complexidades específicas, especialmente quando se trata das funções de *kernel*. É aqui que as aproximações de funções racionais podem ser úteis. Essas aproximações são uma alternativa para introduzir flexibilidade no processo de aprendizagem, mantendo ou possivelmente melhorando a qualidade do modelo. No entanto, a utilização de funções racionais em problemas do mundo real muitas vezes necessita de um distanciamento do campo conveniente da otimização convexa para a realidade mais desafiadora da otimização não convexa.

Com este fim, o presente capítulo tem como objetivo apresentar o referencial teórico como base de uma confluência destes domínios interconectados: aprendizado de máquina, aproximações por funções racionais e otimização não convexa. Cada tópico, acompanhado pelo seu corpo de conhecimento, desempenha um papel num quadro mais abrangente que visa introduzir e melhorar a compreensão sobre novos métodos de aprendizagem automática.

2.1 APRENDIZADO DE MÁQUINA

Aprendizado de Máquina (AM) é um subcampo da inteligência artificial que foca no desenvolvimento de modelos estatísticos capazes de aprender, fazer predições ou tomar decisões baseados em dados. Mitchell, em seu trabalho de 1997, formaliza que, dado a tripla (T, P, E), onde T é uma tarefa, P é uma métrica de desempenho e E é a experiência disponível para o aprendizado, um modelo só é aprimorado com a experiência E com relação a alguma tarefa T e alguma métrica de desempenho P se o desempenho na tarefa T, medida por P, melhora com a experiência E (MITCHELL, 1997).

Um exemplo desta formalização se segue: a experiência E pode ser um conjunto de dados das preferências para o ajuste da temperatura do termostato e da temperatura externa de uma residência medidas várias vezes no dia, em vários dias. A tarefa T pode ser visto como a tarefa de controlar o termostato para manter uma temperatura confortável dentro da residências, que pode ser aquecer ou resfriar. A métrica de desempenho P, então, pode ser visto como o quão perto o termostato mantém a temperatura ao nível esperado sem desnecessárias flutuações ou uso excessivo de energia.

Ainda, faz-se necessário declarar o significado de vetor que será utilizada ao longo

deste trabalho. Vetor é um objeto matemático que representa um elemento do conjunto $\mathbb{R}^d$, onde d é a dimensão ou "tamanho" do vetor e cada entrada do vetor é um valor contido em $\mathbb{R}$. Soma, subtração e produto interno de vetores serão as mesmas operações usuais do espaço euclidiano, bem como a multiplicação por um escalar também pertencente ao conjunto dos reais $\mathbb{R}$.

O processo de AM pode ser categorizado em três subdivisões: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. No aprendizado supervisionado, o modelo é treinado com dados previamente rotulados, o que significa dizer que cada exemplo de treino está associado a um rótulo de saída. O modelo, então, iterativamente, faz predições nos dados de treino e é corrigido quando a predição está incorreta, tipicamente por funções de perda que quantificam o erro entre a predição e o rótulo de fato. Os modelos de regressão linear, árvores de decisão e redes neurais são bem conhecidos na área de aprendizagem supervisionada (NASTESKI, 2017).

Na aprendizagem não supervisionada, é apresentado um conjunto de dados ao modelo sem instruções explicitas do que fazer com ele, isto é, o modelo atua em um conjunto de dados sem rótulos ou uma saída específica a ser obtida. Assim, o sistema tenta aprender os padrões e a estruturar dos dados sem nenhuma supervisão, isto é, rotulação. Análises de agrupamento de dados, aprendizado de regras de associação e análises dos componentes principais são técnicas comuns na área de aprendizado não supervisionado (GHAHRAMANI, 2003).

Aprendizado por reforço lida com agentes que tomam ações em um ambiente para atingirem um objetivo específico. Os agentes aprendem a como atingir o objetivo por tentativa e erro, efetivamente aprendendo a melhor política para uma dada tarefa. Neste paradigma, o processo de aprendizado é formulado como um Processo de Decisão de Markov (PDM), matematicamente descrito por uma tupla (S, A, P, R), onde S é o conjunto de estados possíveis, A é o conjunto de ações, P é a probabilidade de transição de estado e R é a função de recompensa (SUTTON; BARTO, 2018).

2.1.1 Aprendizado Supervisionado

Aprendizado supervisionado é um termo guarda chuva que encapsula algoritmos e métodos designados a modelar relações entre variáveis de entrada e saídas rotuladas. Este tipo de AM necessita da existência de um conjunto de dados rotulado, que é um conjunto de instâncias onde cada vetor de entrada está emparelhado a um rótulo de saída. O principal objetivo do aprendizado supervisionado é inferir um função de mapeamento do tipo $f : X \to Y$ tal que, para cada x no espaço de entrada X, a função $f(x)$ aproxima sua saída $y \in Y$ da saída real ou esperada (BISHOP; NASRABADI, 2006).

O aprendizado supervisionado lida fundamentalmente com dois tipos de tarefas: classificação e regressão. Classificação envolve a atribuição de um rótulo de um conjunto discreto a uma entrada. Um exemplo clássico de aplicação é o filtro de *spam*, onde o

algoritmo de aprendizado de máquina classifica a mensagem como "*spam*" ou "não *spam*" baseado no seu conteúdo. Numa tarefa de classificação binária com as classes C_1 e C_2, o modelo tenta encontrar uma função de decisão que separe as duas classes. Classificação multiclasse, onde o número de classes é maior que dois, é uma extensão da classificação binária, frequentemente abordada pelo uso de técnicas como *One-vs-All* (KOLO, 2011).

Regressão, por outro lado, envolve predizer um valor de saída contínuo. Predição de valores de imóveis é um exemplo canônico de uma tarefa de regressão, onde o objetivo é predizer o valor de venda de uma casa baseado em suas características como tamanho em metros quadrados, números de quartos, etc. Matematicamente, dado um vetor de entrada x, a tarefa é aproximar a função $f(x)$ tal que ela produza um valor contínuo y que seja tão perto quanto possível do valor de saída verdadeiro. A performance é frequentemente calculada usando métricas como Erro Quadrático Médio MSE (HERBRICH; GRAEPEL; OBERMAYER *et al.*, 1999).

Portanto, o aprendizado de máquina supervisionado serve de base para várias domínios que envolvem dados rotulados. O campo oferece uma infinidade de técnicas para abordar classificação e regressão tarefas, cada uma com suas nuances, vantagens e desvantagens. À medida que a pesquisa neste campo continua a acontecer, promove a inovação em inúmeros setores, reforçando a importância e a onipresença de aprendizado de máquina supervisionado na tecnologia contemporânea.

2.1.2 Métricas

Em AM, métricas para avaliação da performance de tarefas de classificação servem como uma ferramenta indispensável para a quantificação da capacidade preditiva de um modelo. Tais métricas proveem uma base matemática para a identificação de pontos positivos e negativos de um modelo, possibilitando que seja feito o ajuste fino nos algoritmos ao passo que decisões informadas sejam tomadas. Esta seção tem como objetivo elucidar as métricas utilizadas no problema de classificação deste trabalho, com atenção especial em sua formulação e aplicabilidade.

Uma das mais rudimentares e ao mesmo tempo poderosa é a métrica de acurácia, que calcula a proporção de verdadeiros positivos e verdadeiros negativos sobre o total de instâncias do conjunto. Matematicamente, é definida como

$$\text{Acurácia} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

onde TP representa os verdadeiro positivos, TN representa os verdadeiros negativos, FP representa os falso positivos e FN representa os falsos negativos. A acurácia é amplamente utilizada, porém sabe-se que não é bem adequada a conjunto de dados desbalanceados (PROVOST; FAWCETT; KOHAVI *et al.*, 1998).

Sensitividade e especificidade são outras duas métricas chaves que lidam com algumas das limitações da acurácia. Sensitividade, ou Taxa de Verdadeiros Positivos

(TVP), é definida como

$$\text{TVP} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

e indica a proporção da assertividade nas instâncias positivas. Especificidade, ou Taxa de Verdadeiros Negativos (TVN), é definida como

$$\text{TVN} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3}$$

e mede a proporção de verdadeiros negativos dentre todos as instâncias negativas de fato. No caso de um problema de classificação binária, a pontuação de acurácia balanceada (BAS) pode definida como a média aritmética entre TVP e TVN

$$BAS = \frac{1}{2}(TVP + TVN) \tag{4}$$

e é particularmente útil em conjunto de dados desbalanceados (URBANOWICZ; MOORE, 2015), pois garante que a proporção de acertos de ambas as classes terão pesos iguais e a métrica não será enviesada pela desproporção das classes.

2.1.3 Validação

A validação no contexto de AM se refere ao processo de avaliar a performance de um modelo treinado em um conjunto de dados que é independente dos dados utilizados para o treinamento. O principal objetivo da validação é determinar a capacidade do modelo de generalizar a predição para dados não vistos, desta forma fornecendo percepções sobre como o modelo provavelmente performará em dados futuros (HASTIE *et al.*, 2009). Particionando o conjunto de dados em subconjuntos distintos, frequentemente chamados de conjunto de treino, validação e teste, é possível mitigar o sobreajuste, um fenômeno onde o modelo aprende com o ruído dos dados de treino ao invés da verdadeira função de decisão (BISHOP; NASRABADI, 2006).

Uma técnica consolidada para particionar os dados se chama método *holdout*, onde uma razão dos dados, proporção tipicamente variando no intervalo compreendido entre 20% e 40%, é separada para teste, ao passo que o restante dos dados é utilizado para treino. O modelo é treinado com os dados de treino e, em seguida, avaliado no conjunto *holdout* de teste para calcular sua performance de generalização. O uso de um conjunto de teste é recomendado para uma avaliação final do modelo sem viés (KOHAVI *et al.*, 1995). Há, ainda, a sua versão estratificada, onde os dois conjuntos apresentam proporções similares entre as classes.

A validação cruzada é outra técnica de validação conhecida, oferecendo uma avaliação confiável por particionar os dados em k conjuntos de tamanhos iguais. O modelo é treinado k vezes, cada vez deixando um conjunto de fora para validação enquanto o treinamento acontece nos $k - 1$ conjuntos restantes. A performance do modelo é a média

da performance dos k conjuntos, fornecendo uma estimativa mais robusta do erro de generalização (GEISSER, 1993).

A validação cruzada estratificada é uma variação da validação cruzada de k conjuntos, onde cada subconjunto mantêm a mesma distribuição de instâncias por classe do conjunto de dados original, o que é particularmente útil para lidar com conjuntos de dados desbalanceados e é crucial para modelos que são sensíveis à distribuição das classes, assegurando que cada subconjunto da validação é uma amostra representativa do conjunto de dados completo (KOHAVI *et al.*, 1995).

2.1.4 Busca em grade

Em AM, o ajuste fino de hiperparâmetros dos modelos é um passo fundamental que pode influenciar significantemente o desempenho de um modelo. A busca em grade é um método tradicional utilizado para o ajuste fino de hiperparâmetros, em que um conjunto predefinido de valores para os hiperparâmetros é exaustivamente testado para encontrar a melhor combinação que minimiza (ou maximiza) alguma métrica de desempenho especificada para o modelo em questão. Pode ser utilizada de maneira efetiva em cenários onde o número de hiperparâmetros dos modelos e os seus potenciais valores são limitados (BERGSTRA; BENGIO, 2012).

O processo de busca em grade envolve a definição de uma grade de valores de hiperparâmetros e a avaliação do desempenho do modelo para cada combinação de hiperparâmetros da grade. Para um determinado modelo de aprendizado de máquina M com hiperparâmetros $h_1, h_2, \ldots, h_n$, uma grade pode ser definida como o produto cartesiano dos valores predefinidos dos hiperparâmetros. Se S_{h_i} representar o conjunto de valores para o hiperparâmetro h_i, a grade B pode ser representada como: $B = S_{h_1} \times S_{h_2} \times \cdots \times S_{h_n}$. Para cada combinação $g \in G$, o modelo M é treinado e seu desempenho é avaliado utilizando um conjunto de validação ou via validação cruzada (KOHAVI *et al.*, 1995).

2.2 TRANSFORMAÇÃO DE DADOS

A transformação dos dados antes de lidar com tarefas de AM é uma etapa fundamental na modelagem de dados por uma séries de razões, incluindo, mas não limitado, a normalização, ajuste de valores discrepantes, preenchimento de valores faltantes e facilita a extração de atributos úteis. Essas transformações influenciam significativamente a eficácia dos modelos de AM, melhorando sua habilidade de generalizar em dados não vistos. Transformar os dados, então, não é uma mera convenção, mas um aspecto bem estudado em inúmeros trabalhos da literatura (LEE *et al.*, 2021).

Normalização ou padronização dos dados, apesar de serem técnicas tradicionais, são técnicas essenciais para a transformação de dados. Algoritmos de AM que são sensíveis a escala dos atributos, como em SVMs, podem se tornar modelos imprecisos ou subótimos

ao utilizar dados fora de escala (SARLE, 1994). Ao transformar os dados para uma escala normalizada, é possível eliminar as influências desproporcionais que os atributos de grandes escalas poderiam exercer. Por exemplo, a normalização *Min-Max* transforma x em y calculando

$$y = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{5}$$

onde x_{min} é o valor mínimo de X e x_{max} é o valor máximo de X, onde X é o conjunto que contém os valores de x e Y é o conjunto dos valores de X normalizados, garantindo que os novos valores estarão entre zero e um.

2.2.1 Padronização dos dados com *z-score*

O *z-score* é uma técnica de padronização dos dados baseada no conceito de padronização estatística e traz consigo um mecanismo para transformar diferentes escalas de atributos para uma escala comparável. O *z-score* y de um valor x em um dado conjunto de dados é calculado como

$$z = \frac{x - \mu_X}{\sigma_X}, \tag{6}$$

onde μ_X e σ_X são, respectivamente, a média e o desvio padrão dos valores do conjunto X a qual o x pertence. A fórmula então normaliza os desvios da média, utilizando o desvio padrão como unidade de medida (EVERITT; SKRONDAL, 2010).

Em um conjunto de dados com vários atributos, este método transforma a distribuição X de um determinado atributo em uma nova distribuição Z que segue uma distribuição com média μ_Z igual a zero e desvio padrão σ_Z igual a um. A padronização alcançada pela transformação *z-score* é essencial para certos tipos de algoritmos de AM, principalmente aqueles que tem como base alguma medida de distância e que dependem da escala da distribuição dos dados. Como os atributos estão na mesma escala, o valor da distância não será enviesado por algum atributo com escala desproporcional em relação as demais.

É preciso notar que, no contexto de modelos de AM, é crítico o cálculo dos parâmetros média e desvio padrão de cada atributo a partir do conjunto de treino sozinho, e só depois aplicar esses parâmetros para padronizar ambos conjuntos de treino e de teste. Isso garante que a transformação não esteja enviesada, testando a capacidade de generalizar em dados novos e desconhecidos e mantendo a integridade do processo de avaliação (GÉRON, 2022).

2.3 MÁQUINA DE VETORES DE SUPORTE

Máquina de Vetores de Suporte (SVMs) são algoritmos de aprendizado de máquina supervisionado extensivamente utilizados para tarefas de classificação e, em menor extensão, de regressão. Introduzidas por Vapnik e Chervonenkis em 1995, seu reconhecimento

foi difundido pela sua robustez, acurácia e eficácia em espaços de alta dimensão (CORTES; VAPNIK, 1995).

Na sua essência, SVMs operam na identificação do hiperplano ótimo que separa o conjunto de dados de diferentes classes no espaço de características. Para o caso de classificação binária linearmente separável, o hiperplano é definido como

$$w \cdot x + b = 0, \tag{7}$$

onde w é o vetor de pesos e b é o viés. A função de decisão para a classificação, então, pode ser denotada como

$$f(x) = \text{sinal}(w \cdot x + b). \tag{8}$$

O hiperplano ótimo maximiza a margem, que é a distância entre os pontos dos dados mais próximos (vetores de suporte) de diferentes classes para o hiperplano. O problema de otimização, abordado pela resolução do problema dual equivalente com uso da programação quadrática, é em suma

$$\min \frac{1}{2}\|w\|^2 \quad \text{sujeito a} \quad y_i(w \cdot x_i + b) \geq 1, \quad i = 1, \ldots, n, \tag{9}$$

onde y_i é o rótulo de x_i e n é o número de exemplos de treino (CRISTIANINI; SHAWE-TAYLOR, 2000).

Em casos onde os dados não são linearmente separáveis ou quando é mais conveniente trabalhar num espaço de características de dimensão mais alta, SVMs podem utilizar funções de *kernel* que calcula o resultado do produto interno implicitamente mapeando os dados em um espaço de dimensão mais alta. O uso de funções de *kernel* permite a construção de funções de decisão não lineares sem explicitamente calcular as coordenadas no espaço de dimensão mais alta, não altera as equações de otimização das SVMs e preserva as características de convergência da programação quadrática (SCHÖLKOPF; SMOLA, Alexander J, 2002).

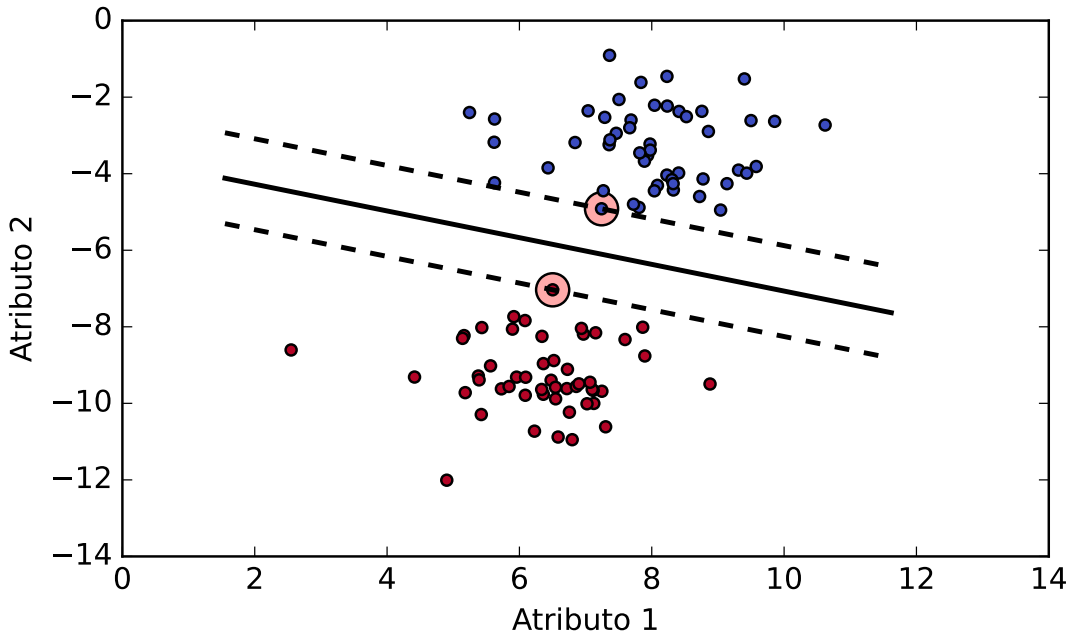
Suplementarmente, SVMs podem ser estendidas para problemas de múltiplas classes pelo uso de estratégias como *One-vs-One* ou *One-vs-All*. Ainda mais, pode ser adaptada para tarefas de regressão (*Support Vector Regression*), detecção de anomalias e outras tarefas ao modificar apropriadamente a função de perda e suas restrições (SMOLA, Alex J; SCHÖLKOPF, 2004).

Nas seções que se seguem, serão apresentadas as formulações matemáticas da SVM de margem rígida, de margem flexível e a sua forma dual. Depois, o truque do *kernel* e o teorema de Mercer serão explicados com mais detalhes, sucedido pela importância do número de interações utilizados na otimização de programação quadrática e o que significa o número de vetores de suporte ao fim do treino de uma SVM.

2.3.1 Margem Rígida

A variante da SVM de margem rígida é particularmente efetiva quando os dados são linearmente separáveis, otimizado o hiperplano para atingir a separação máxima entre as diferentes classes no espaço de características. SVMs são construídas com a intuição de que uma margem larga entre a função de decisão e os pontos mais próximos de cada classe levam a um classificador com menor erro. Uma margem larga significa que o hiperplano é posicionado tão longe quanto possível dos pontos mais próximos de cada classe, dessa forma maximizando a separação entre as classes (SCHÖLKOPF; SMOLA, Alexander J, 2002). A Figura 1 mostra um exemplo da SVM de margem rígida.

Figura 1 – SVM de margem rígida.



Fonte: De autoria própria.

Considere um conjunto de dados

$$\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{n} \tag{10}$$

onde $x_i \in \mathbb{R}^d$ são os vetores do espaço de característica e $y_i \in \{-1, 1\}$ são os rótulos correspondentes. A função de decisão é representada pela equação do plano definido por

$$w \cdot x + b = 0 \tag{11}$$

onde w é o vetor de pesos, x é qualquer ponto no hiperplano e b é o termo de viés.

O objetivo é encontrar w e b tal que a margem ρ, definida pela menor distância dos pontos em relação ao hiperplano, é maximizada. Seja x_o um ponto no hiperplano e x_c um ponto com a menor distância ao plano. Com isso, o vetor x_{oc}, começando em x_o

e término em x_c, pode ser definido e a sua distância ao plano pode ser calculada pelo tamanho do vetor projetado sobre w:

$$\rho = \|proj_w(x_{oc})\| = \tag{12}$$

$$= \left\| \frac{w \cdot x_{oc}}{\|w\|} \frac{w}{\|w\|} \right\| = \tag{13}$$

$$= \frac{|w \cdot x_{oc}|}{\|w\|} \left\| \frac{w}{\|w\|} \right\|. \tag{14}$$

Como $\frac{w}{\|w\|}$ é um vetor unitário, sua norma é igual a 1. Substituindo x_{oc} por $(x_c - x_o)$, obtêm-se:

$$\rho = \frac{|w \cdot (x_c - x_o)|}{\|w\|}. \tag{15}$$

Adicionando e somando b, têm-se:

$$\rho = \frac{|w \cdot (x_c - x_o) + b - b|}{\|w\|} = \tag{16}$$

$$= \frac{|w \cdot x_c + b - (w \cdot x_o + b)|}{\|w\|}. \tag{17}$$

Note que $w \cdot x_o + b = 0$, pois x_o está contido no plano. Logo:

$$\rho = \frac{|w \cdot x_c + b|}{\|w\|}. \tag{18}$$

Sem perda de generalidade, é possível forçar que a menor distância para o plano seja igual a 1. Com isso, $w \cdot x_c + b = 1$ e, por substituição, a margem ρ é definida como:

$$\rho = \frac{1}{\|w\|}. \tag{19}$$

Ainda mais, como o objetivo inicial é maximizar a margem ρ, pode-se notar que as seguintes cláusulas são equivalentes:

$$\max \rho \equiv \max \frac{1}{\|w\|} \equiv \min \|w\| \equiv \min \|w\|^2 \equiv \min \frac{\|w\|^2}{2}. \tag{20}$$

Como $\|w\|$ é estritamente positivo, $1/\|w\|$ é estritamente monótona decrescente quando $\|w\|$ aumenta e $\|w\|$ é estritamente monótona crescente quando $\|w\|$ aumenta. Para maximizar $1/\|w\|$ é preciso que $\|w\|$ seja o menor possível. Para minimizar $\|w\|$ é preciso que $\|w\|$ seja o menor possível, justificando a equivalência entre $\max 1/\|w\|$ e $\min \|w\|$.

Ainda, quando $\|w\|$ decresce, $\|w\|^2$ também decresce. Quando $\|w\|$ aumenta, $\|w\|^2$ também aumenta. Ambas $\|w\|$ e $\|w\|^2$ são estritamente monótonas, justificando que $\min \|w\|$ é equivalente a $\min \|w\|^2$. Por fim, $\min \|w\|^2/2 = (\min \|w\|^2)/2$, que possui a mesma solução de $\min \|w\|^2$. Essas equivalências são importantes pois são matematicamente convenientes para a formulação do problema dual da SVM por meio de seu primal (Seção 2.3.3).

Definindo a classe dos pontos x_+ que estão acima do hiperplano como a classe positiva, isto é, $y_+ = +1$, e a classe dos pontos x_- que estão abaixo do hiperplano como a classe negativa, isto é, $y_- = -1$, é possível deduzir que

$$y_+(w \cdot x_+ + b) > 0, \quad \text{pois} \quad y_+ = +1 \quad \text{e} \quad w \cdot x_+ + b > 0 \tag{21}$$

$$y_-(w \cdot x_- + b) > 0, \quad \text{pois} \quad y_- = -1 \quad \text{e} \quad w \cdot x_- + b < 0. \tag{22}$$

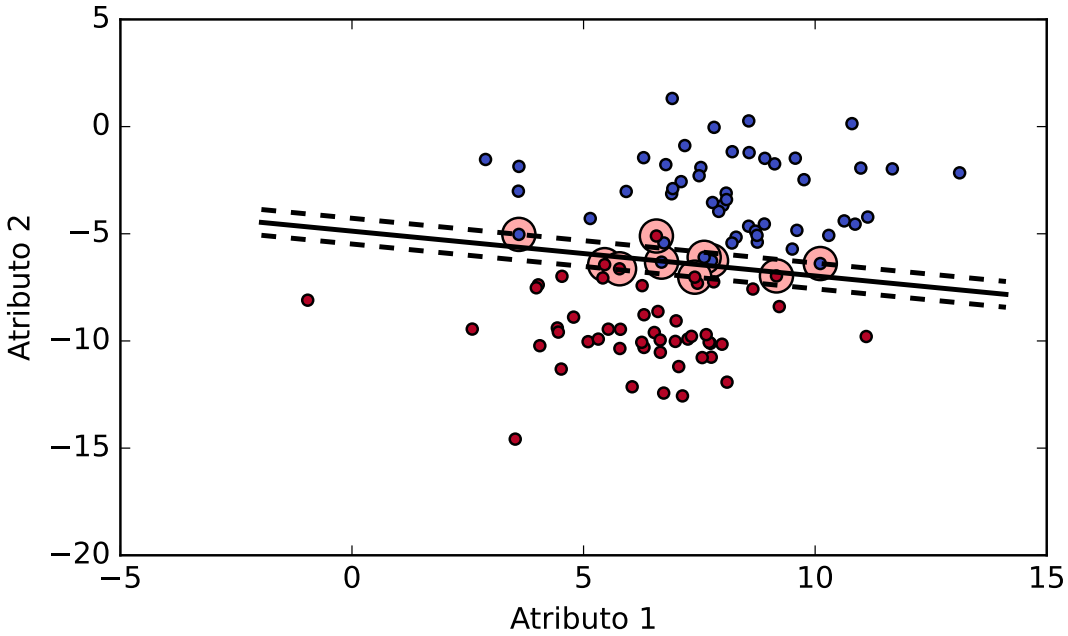
Assim, independente da classe, os vetores x_i obedecem a inequação:

$$y_i(w \cdot x_i + b) > 0, \quad i = 1, 2, \ldots, n. \tag{23}$$

Como $w \cdot x_c + b = 1$, sendo x_c o ponto de menor distância ao plano, e com a Equação (20) e a Equação (23), é possível derivar a Equação (9). Os dados são linearmente separáveis, portanto mirar a maximização de ρ é garantir a classificação perfeita na SVM de margem rígida, conectando a noção geométrica com princípios matemáticos robustos e lançando as bases para várias extensões e aplicações do conceito em áreas do AM.

2.3.2 Margem Flexível

Figura 2 – SVM de margem flexível.



Fonte: De autoria própria.

Na extensão da margem flexível, variáveis de folga ξ_i são introduzidas para permitir que alguns pontos do conjunto de dados possam cair dentro da margem ou até serem classificadas incorretamente. A otimização incorpora esses termos de regularização para

balancear a troca entre maximizar a margem e minimizar os erros de classificação

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \tag{24}$$

sob as restrições

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \text{e} \quad \xi_i \geq 0, \quad i = 1, 2, \ldots, n. \tag{25}$$

O parâmetro C controla a troca e é geralmente escolhido pelo processo de validação cruzada. As variáveis de folga ξ_i medem o grau de classificação incorreta para cada ponto i do conjunto de dados.

SVMs de margem flexível são validadas tanto empiricamente quanto teoricamente pela sua habilidade de generalizar bem em uma variedade de distribuição de dados (SCHÖLKOPF; SMOLA, Alexander J, 2002). A Figura 2 traz um exemplo de sua aplicação. A introdução das variáveis de folga e do parâmetros de regularização equipam as SVMs com a flexibilidade pra lidar com dados ligeiramente não linearmente separados e com ruídos de maneira eficiente.

2.3.3 Forma dual

Utilizando as equações (24) e (25) previamente discutidas, é possível formular a função Lagrangiana (BEAVIS; DOBBS, 1990) para o dado problema de otimização, como se segue:

$$L(w, b, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i[y_i(w \cdot x_i + b) - 1 + \xi_i] - \sum_{i=1}^{n} \beta_i \xi_i. \tag{26}$$

Aqui, α e β são os multiplicadores de Lagrange. Com o uso das condições de Karush-kuhn-tucker (BAZARAA; SHERALI; SHETTY, 2013) , ou condições de KKT, pode-se achar a forma dual tomando a derivada de L com respeito a w, b e ξ_i e as igualando a zero:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \tag{27}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{28}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0. \tag{29}$$

Substituindo as equações de volta na função Lagrangiana, obtêm-se o problema dual:

$$\text{maximize} \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{30}$$

$$\text{sob as restrições} \quad 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{31}$$

Deve ser notado que o problema dual é um problema de programação quadrática, garantindo que suas soluções globais sejam computacionalmente tratáveis sob condições específicas (BOYD; VANDENBERGHE, 2004). A forma dual é particularmente vantajosa porque só envolve os pontos dos dados através de seus produtos internos, desta forma possibilitando o uso do truque do *kernel* para classificação não linear.

2.3.4 O Truque do *kernel* e o teorema de Mercer

O truque do *kernel* e o teorema de Mercer são conceitos pertinentes em AM, particularmente para SVMs e vários outros algoritmos que operam com espaços de características de alta dimensão. O truque do *kernel* é uma técnica computacional que possibilita um mapeamento de dados implícito para um espaço de alta dimensão sem explicitamente calcular as coordenadas naquele espaço. Isso contorna a sobrecarga computacional frequentemente associada com esse tipo de mapeamento. O teorema de Mercer fornece a estrutura matemática para validar se uma dada função pode servir como *kernel*, ou seja, que a função corresponde a um produto interno válido em um espaço transformado (VAPNIK, 1998; SCHÖLKOPF; SMOLA, Alexander J, 2002).

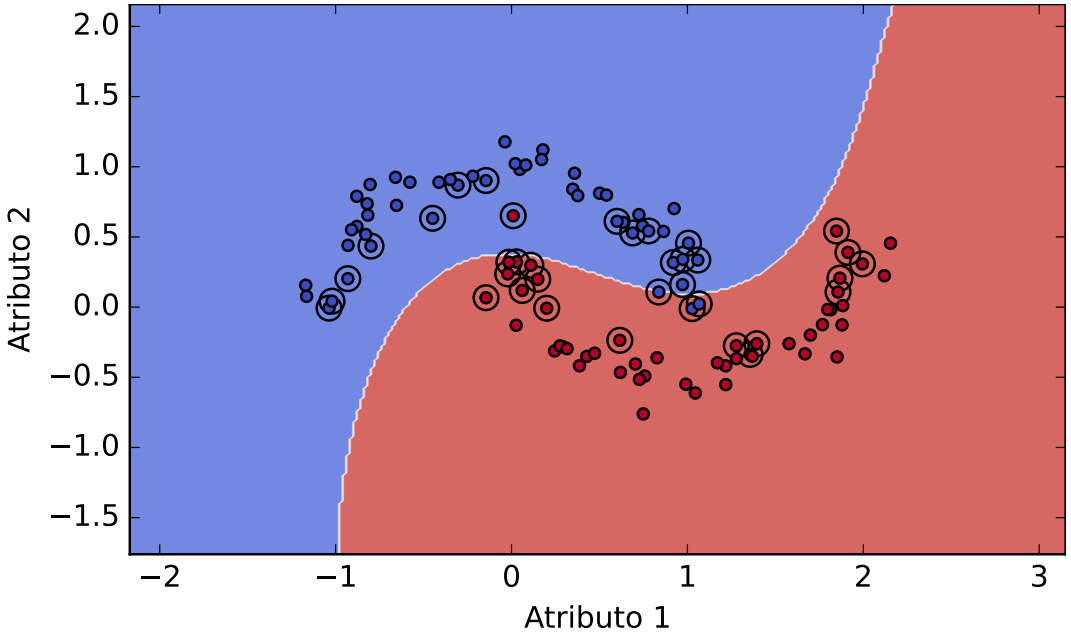
Sejam x, x_i e x_j vetores de dimensão d com entradas reais, isto é, $\{x, x_i, x_j\} \subset \mathbb{R}^d$. Dada a função $K(x_i, x_j)$, o truque do *kernel* implica a substituição do produto interno $x_i \cdot x_j$ no espaço original por $K(x_i, x_j)$ tal que $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, onde ϕ é uma função de mapeamento para um espaço de dimensão mais alta. Os benefícios dessa técnicas se manifesta na eficiência computacional que ela fornece. Calculando $\phi(x)$ explicitamente pode ser computacionalmente caro ou até intratável para espaços de dimensão infinita, mas $K(x_i, x_j)$ pode, frequentemente, ser calculado diretamente em complexidade de tempo constante ou linear, desta forma contornando a necessidade de conhecer $\phi(x)$.

No entanto, nem toda função $K(x_i, x_j)$ pode servir como um *kernel* válido. O teorema de Mercer estabelece as condições necessárias e suficientes para que uma função seja considerada um *kernel* válido. Especificamente, para $K(x_i, x_j)$ ser um *kernel* de Mercer, deve ser simétrica e deve produzir uma matriz de Gram que seja positiva semidefinida para qualquer conjunto finito de pontos do conjunto de dados. Matematicamente, a função $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ é dita ser um *kernel* de Mercer se satisfaz duas condições:

1. $K(x_i, x_j) = K(x_j, x_i)$, $\forall x_i, x_j \in \mathbb{R}^d$, garantindo a simetria;
2. Para qualquer conjunto finito $\{x_1, x_2, \ldots, x_n\}$, a matriz de Gram associada Q definida como $Q_{ij} = K(x_i, x_j)$ precisa ser positiva semidefinida.

O teorema de Mercer afirma ainda que, se $K(x_i, x_j)$ é um *kernel* de Mercer, então existe um mapa ϕ tal que $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ (SHAWE-TAYLOR; CRISTIANINI, 2004). Esse teorema forma o alicerce para a construção e validação de funções de *kernel* complexas para o uso em AM. A Figura 3 é um exemplo da aplicação do teorema de Mercer na Equação (30), substituindo $x_i \cdot x_j$ por $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, assegurando

Figura 3 – SVM de margem flexível e não linear pelo uso da função de *kernel*



Fonte: De autoria própria.

que o problema de otimização dual continue convexo na sua definição e, como resultado, garantindo o ótimo global.

2.3.5 Número de Iterações

A convergência das SVMs é um aspecto crítico da sua implantação em aplicações de AM. Em particular, o número de iterações necessários para a convergência é uma consideração substancial tanto para eficiência quanto efetividade do modelo. A discussão que segue mira na elucidação da importância do número de iterações na convergência da SVM tendo como base as fundações matemáticas do algoritmo.

Solucionadores como o Otimização Sequencial Mínima (SMO) (PLATT, 1998) e os métodos baseados em gradiente, no contexto de SVMs, tem como objetivo minimizar a função objetivo iterativamente. O número de iterações é crítico por duas razões: eficiência computacional e acurácia do modelo. Quando controlado apropriadamente, um número grande de iterações tipicamente leva a um modelo mais assertivo, mas com custo de recursos computacionais e tempo. Por outro lado, poucas iterações podem levar a resultados mais rápidos, porém com o potencial comprometimento da acurácia ou capacidade de generalização do modelo (SHALEV-SHWARTZ; SINGER; SREBRO, 2007).

A taxa de convergência pode ser influenciada por diversos fatores, incluindo, mas não limitado, às condições iniciais, a função de *kernel* utilizada e os parâmetros de tolerância no algoritmo de otimização. A literatura indica que a escolha do algoritmo de otimização pode ter um papel significativo. Por exemplo, o Gradiente Descendente Es-

tocástico (SGD) pode ser mais rápido, porém requer mais iterações para convergir para uma solução ótima ou próxima da otimalidade (BOTTOU, 2010).

O número de iterações para a convergência em SVMs é um ponto válido de debate por impactar tanto a eficiência computacional quanto a efetividade do modelo. Encontrar o equilíbrio certo é crucial e depende de múltiplo fatores, incluindo a complexidade dos dados, a escolha do algoritmo de otimização e os recursos computacionais evolvidos. À medida em que as aplicações de AM evoluem, obter uma compreensão da interação entre esses fatores continua como uma área relevante na comunidade científica.

2.3.6 Número de Vetores de Suporte

Outro aspecto importante das SVMs em relação a sua complexidade e performance de generalização é o número de vetores de suporte. Os vetores de suporte são os pontos do conjunto de dados que moram na margem ou dentro dos limites da margem, e tem um papel vital na determinação do hiperplano ótimo que separa as classes no espaço de características.

A importância do número de vetores de suporte na convergência do algoritmo SVM pode ser explicado de múltiplas perspectivas: complexidade computacional, generalização do modelo e otimização da margem. Quando a folga complementar das condições de KKT é analisada, é possível mostrar que alguns α_i da Equação (31) são iguais a zero. Isto é, a condição não impõe nenhuma restrição sobre o vetor que tem o seu α_i correspondente a zero. Para os $\alpha_i > 0$, o vetor x_i associado está dentro da margem ou está classificado incorretamente, e são os vetores (de suporte) que determinam o hiperplano.

Portanto, a função de decisão é inteiramente definida pelos vetores de suporte, isto é, pelos vetores que têm os seus respectivos α_i positivos. Para determinar a classe $\hat{y}$ de um vetor de teste com a função de decisão, é preciso resolver o vetor de pesos w com a Equação (27) substituindo na função de decisão descrita pela Equação (8), como se segue:

$$f(\mathbf{x}) = \sum_{(x_i,\alpha_i)\in SV} \alpha_i y_i (x_i \cdot \mathbf{x}) + b, \tag{32}$$

onde SV é o conjunto dos vetores de suporte e seus respectivos α_i. Utilizando o truque do *kernel*, finalmente obtemos:

$$f(\mathbf{x}) = \sum_{(\alpha_i,x_i)\in SV} \alpha_i y_i K(x_i, \mathbf{x}) + b \tag{33}$$

$$\hat{y} = \text{sinal}(f(\mathbf{x})). \tag{34}$$

Percebe-se que, como a função de decisão utiliza os vetores de suporte, uma quantidade maior de vetores de suporte aumenta a complexidade e utilização de recursos computacionais para a predição. Outro ponto é a capacidade de generalização do modelo. Um alto número de vetores de suporte pode implicar que o modelo capturou além da estrutura intrínseca dos dados, embora com risco de sobreajuste (CORTES; VAPNIK, 1995).

Todavia, um conjunto pequeno de vetores de suporte indica um modelo mais robusto e mais generalizado, porém potencialmente subajustado em relação aos dados. Um equilíbrio entre estes dois extremos é muitas vezes determinado por técnicas de regularização e estratégias de seleção de modelos.

Um número alto de vetores de suporte implicaria em uma função de decisão complexa, portanto potencialmente reduzindo a margem. Uma margem pequena pode resultado em uma generalização pobre em um conjunto de dados de teste (BURGES, 1998). Em contraste, um número pequeno de vetores de suporte podem implicar em uma margem larga, o que geralmente é considerado uma vantagem para a capacidade de generalização do modelo. A relação entre toda a margem M e os vetores de suporte pode ser expressada como:

$$M = 2\rho = \frac{2}{\|w\|} = \frac{1}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)}} \tag{35}$$

onde w é o vetor de pesos, α_i e α_j são os multiplicadores de Lagrange para os vetores de suporte x_i e x_j, e K é a função de *kernel*. A Equação (35) por ser deduzida a partir das equações (27) e (19).

2.4 APROXIMAÇÕES POR FUNÇÕES RACIONAIS

Aproximação de função utilizando funções racionais constituem uma metodologia importante na análise numérica. Funções racionais são expressões da forma $\frac{P(x)}{Q(x)}$, onde $P(x)$ e $Q(x)$ são polinômios. O fascínio das funções racionais na aproximação se sustenta na sua flexibilidade e poder representativo: pode capturar características complexas, incluindo polos e zeros, melhor do que aproximações polinomiais com o mesmo grau (STAHL; SCHMELZER, 2009). Além disso, funções racionais geralmente mostram propriedades de convergência superiores quando comparadas com aproximações polinomiais, especialmente para funções com singularidades (NEWMAN; SHAPIRO, 1964).

Uma metodologia clássica para aproximar uma função $f(x)$ sobre um domínio $[a, b]$ usando funções racionais é o aproximante de Padé (BREZINSKI, 1994), definido como a função racional $\frac{P(x)}{Q(x)}$ que concorda com $f(x)$ em tantos termos quanto possível para a sua respectiva série de Taylor. Matematicamente, o aproximante $[m/n]$ de Padé é uma função racional $R(x) = \frac{P(x)}{Q(x)}$, onde $P(x)$ é um polinômio de grau m e $Q(x)$ é um polinômio de grau n, tal que a expansão da série Taylor para $f(x) - R(x)$ começa com o termo de ordem $m + n + 1$ (BAKER; GRAVES-MORRIS, 1996). Os aproximantes de Padé tem aplicações em diversos campos, como teoria de controle e processamento de sinais, devido a suas características robustas de aproximação.

Como exemplo, considere uma aproximação para $f(x) = e^x$ sobre o intervalo $[0, 1]$ usando um aproximante de Padé de ordem $[1/1]$. O aproximante

$$R(x) = \frac{a_0 + a_1 x}{1 + b_1 x} \tag{36}$$

pode ser obtido ao coincidir as séries de Taylor de $f(x)$ e $R(x)$ até o termo de segunda ordem, o que traz um sistema de equações para resolver os coeficientes a_0, a_1 e b_1. O aproximante resultado terá uma boa aproximação de e^x no intervalo $[0, 1]$, ilustrando a eficácia da aproximação de funções racionais.

2.5 OTIMIZAÇÃO NÃO CONVEXA

Otimização não convexa é uma área de pesquisa que se preocupa em encontrar soluções para problemas não convexos, ou seja, quando sua função objetivo não for convexa ou quando qualquer uma de suas restrições introduzir não convexidade. Diferente da otimização convexa, problemas não convexos têm a tendência de conter múltiplas soluções ótimas locais e pode ser intratáveis computacionalmente para encontrar a solução global.

No estudo de problemas de otimização não convexos, a noção de um espaço conexo é crucial para entender a topologia das regiões de soluções viáveis onde o mínimo global reside. Do ponto de vista topológico, conexidade implica que o espaço não pode ser particionado em dois conjuntos disjuntos abertos e não vazios (LEFSCHETZ, 2015).

A relevância de espaços conexos em otimização não convexa surge quando as funções objetivo do problema otimização em questão são consideradas. Funções não convexas são caracterizadas pela presença de múltiplos mínimos locais e, possivelmente, um mínimo global, fazendo com sua otimização seja desafiadora. Se a região de soluções viáveis $\Omega \subset \mathbb{R}^n$ na qual a função é definida for conexa, é garantido que quaisquer dois pontos de Ω pode ser unidos por um caminho contínuo inteiramente contido em Ω (CROOM, 2016). Essa propriedade é fundamental para formular algoritmos de otimização, pois impacta a capacidade do algoritmo de explorar a região de soluções viáveis sem encontrar partes desconectadas.

Nesse contexto, formalmente, diz-se que a função $f : \mathbb{R}^n \to \mathbb{R}$ é não convexa se não satisfaz a condição de convexidade, ou seja, existem $x, y \in \mathbb{R}^n$ no domínio conexo de f tais que exista um valor $t \in [0, 1]$ onde a inequação:

$$f(tx + (1 - t)y) > tf(x) + (1 - t)f(y) \tag{37}$$

é verdadeira. A complexidade das funções não convexas tem um impacto crítico na convergência dos algoritmos de otimização construídos para resolvê-las (BOYD; VANDENBERGHE, 2004).

Dada a onipresença de problemas não convexos em vários domínios de aprendizado de máquina, pesquisa operacional e engenharia, várias técnicas têm sido desenvolvidas para aborda-los. Uma abordagem tradicional é utilizar o Gradiente Descendente Estocástico (SGD), que, no entanto, pode tão somente garantir a convergência para o mínimo local. Outra direção pode ser tomada: utilizar técnicas meta-heurísticas como Recozimento Simulado (SA), Algoritmos Genéticos (GA) ou Otimização por Enxame de Partículas

(PSO), que oferecem uma probabilidade maior de escapar de mínimos locais, mas sem garantias de convergência para o ótimo global (BLUM; ROLI, 2003).

Apesar dos avanços, resolver problemas de otimização não convexa para encontrar a solução ótima continua sendo um dificultoso desafio, amplamente por conta da explosão combinatorial das possíveis soluções. Na prática, obter uma solução $\epsilon-$ótima, isto é, uma solução que esteja a uma distância ϵ do ótimo global, geralmente é considerada satisfatória.

2.5.1 Algoritmos Evolucionários

Os Algoritmos Evolucionários (EAs) são uma família de algoritmos de otimização inspirados no processo de seleção natural. Eles têm ganhado significativa atenção no campo da inteligência artificial por resolver problemas de otimização complexos que são computacionalmente intratáveis por métodos tradicionais (EIBEN *et al.*, 2003). Um algoritmo evolucionário opera em uma população de potenciais soluções para o problema em questão. Essas soluções potenciais são submetidas a operações que simulam processos de evolução natural, como a seleção, a recombinação e a mutação para evoluir ao longo das gerações.

O mecanismo fundamental em EAs é a avaliação dos indivíduos de uma população através de uma função de aptidão. A função de aptidão representa quantitativamente o quão bem uma solução individual satisfaz os critérios de otimização (DE JONG, 2017). Formalmente, dada uma população $\mathbb{P}$ com P indivíduos, a função de aptidão $A : \mathbb{P} \to \mathbb{R}$ mapeia cada indivíduo x a um número real, que significa sua qualidade (ou aptidão) para o problema de otimização em questão.

Subsequentemente, mecanismos de seleção escolhem indivíduos baseados na sua aptidão para serem os parentes, gerando descendentes para a próxima geração. Existem vários métodos de seleção, como a seleção aleatória, seleção roleta, seleção torneio e seleções baseadas em ranqueamento, cada uma com suas vantagens e desvantagens.

Após a seleção, as operações de recombinação e mutação são executadas. A recombinação mescla os traços de duas soluções parentes para produzir um ou mais descendentes. Em termos matemáticos, para os parentes x_1 e x_2, o operador de recombinação $R : \mathbb{P} \times \mathbb{P} \to \mathbb{P}$ gera o descendente y, isto é, $y = R(x_1, x_2)$.

A mutação, em contrapartida, introduz pequenas mudanças randômicas em uma solução individual. Formalmente, o operador de mutação $T : \mathbb{P} \to \mathbb{P}$ altera um indivíduo x para produzir um indivíduo mutante x', ou seja, $x' = T(x)$. O processo evolucionário é iterativo, evoluindo a população ao longo de um número de gerações até que um critério de término seja satisfeito.

2.5.2 Evolução Diferencial

A Evolução Diferencial (DE) é um algoritmo de otimização baseado em população que pertence a ampla classe dos algoritmos evolucionários. Introduzida por Storn e Price

em 1997, a DE é primariamente utilizada para otimizar funções multidimensionais com valor real (STORN, R.; PRICE, K., 1997). O algoritmo é particularmente efetivo pra otimização de problemas que são não diferenciáveis, não lineares e multimodais, além de ser simples e de fácil aplicabilidade.

O elemento crucial da DE envolve evoluir uma população de candidatos a solução iterativamente. A evolução é guiada por operações matemáticas simples, como mutação, recombinação e seleção. Ao contrário de outros algoritmos evolucionários que requerem operações distintas para parâmetros reais ou inteiros, DE opera diretamente com parâmetros reais, simplificando sua aplicabilidade em vários domínios (PRICE, K. V.; STORN, R. M.; LAMPINEN, 2005).

O passo a passo do algoritmo para a versão tradicional "DE/rand/1/bin" da Evolução Diferencial pode ser descrito como se segue:

1. Inicialize a população $\mathbb{P}$ com P vetores $[x_1, x_2, ..., x_P]$, onde cada $x_i \in \mathbb{R}^d$;
2. Para cada geração n até N:
    a) Para cada vetor x_i:
        i. Selecione estocasticamente três vetores distintos a, b, c de $\mathbb{P}$;
        ii. Crie um vetor mutante v utilizando: $v = a + F \times (b - c)$, onde $F \in \mathbb{R}$ é um fator de escala;
        iii. Crie um vetor de teste u calculado a recombinação binomial entre x_i e v;
        iv. Calcule a aptidão de u e x_i;
        v. Se u é melhor que x_i, substitua x_i por u;
    b) Atualize $\mathbb{P}$ baseado nas substituições feitas;
3. O melhor vetor na população final $\mathbb{P}$ é a solução.

A variação "DE/rand/1/bin" é uma das estratégias mais populares em DEs e influencia como os vetores de teste e mutantes são gerados. Nessa estratégia, "rand" implica que os vetores a, b e c são aleatoriamente escolhidos da população. O "1" significa que uma única diferença de vetor, $(b - c)$, é usada na mutação. O termo "bin" se refere a recombinação binomial, que determina que o vetor de teste u é formado x_i e v.

O vetor mutante $v \in \mathbb{R}^d$ para a "DE/rand/1" é calculado como:

$$v = a + F \times (b - c), \tag{38}$$

onde $F \in \mathbb{R}$ é um fator de escala definido pelo usuário. Cada entrada $u_j \in \mathbb{R}$ do vetor de teste $u \in \mathbb{R}^d$, com $j = 1, 2, \ldots, d$, é gerada pelo uso da recombinação binomial, que obedece a seguinte lei de construção:

$$u_j = \begin{cases} v_j, & \text{se} \quad sorteio\_real(0,1) \leq CR \quad \text{ou se} \quad j = sorteio\_inteiro(1,d) \\ x_j, & \text{caso contrário} \end{cases}, \tag{39}$$

onde CR é a taxa de recombinação, j é o índice da dimensão, d é a dimensionalidade do vetor, $sorteio\_real(0, 1)$ é uma função que retorna um número real aleatório entre 0 e 1 e $sorteio\_inteiro(1, d)$ é uma função que retorna um número inteiro aleatório entre 1 e d.

3 METODOLOGIA

Para prosseguir com o detalhamento da metodologia, é preciso entender a simbologia (Figura 4) utilizada nos fluxogramas deste capítulo. O símbolo "Dados" representa o conjunto ou partições do conjunto de dados que serão utilizados nos fluxos. O símbolo "Entrada" descreve configurações que são necessárias para a execução de alguma parte do fluxo. O ícone "Processo" representa uma execução ou uma série de execuções dentro do fluxo e o ícone "Saída" reflete saídas de processos que são de interesse na representação. Os processos podem ter saídas ou subprocessos omitidos em favor do entendimento holístico do fluxo.

Figura 4 – Simbologia da metodologia.

Fonte: De autoria própria.

3.1 PROPOSTA

Essa seção é dividia entre duas abordagens, a função de *kernel* RBF como base de comparação (Seção 3.1.1) e a função de *kernel* racional (Seção 3.1.2), que é a proposta deste trabalho. As duas abordagens assumem a modelagem de margem flexível (Seção 2.3.2) e utilizam a SVM na sua versão dual (Seção 2.3.3). O objetivo é garantir que as duas abordagens serão devidamente explicadas, portando como base o conteúdo apresentado no Capítulo 2 e proporcionando um alicerce experimental para que as abordagens sejam comparadas de maneira adequada.

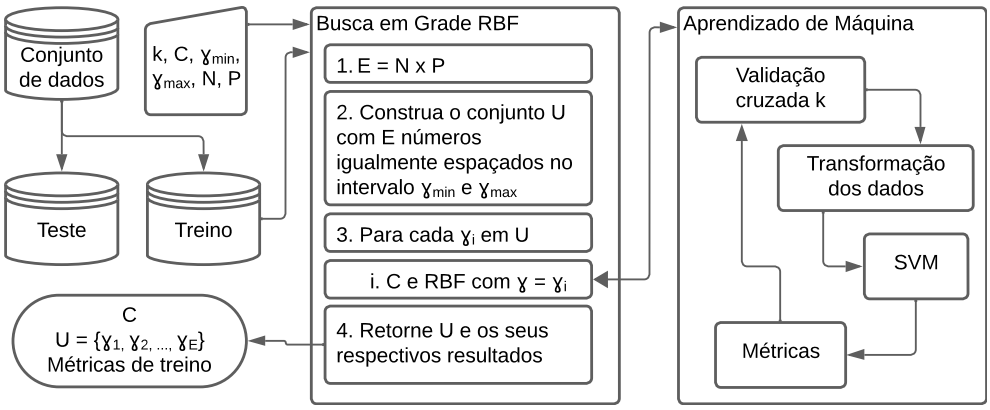
3.1.1 *Kernel* RBF como base de comparação

Por ser uma função popular entre os praticantes da SVM (Seção 2.3), o *kernel* RBF foi escolhido como base de comparação. A Figura 5 mostra o fluxo de treino para a RBF. Inicialmente, os dados deve ser dividido em dois subconjuntos, um para treino e o outro para teste, pela utilização do método *holdout* (Seção 2.1.3) na proporção escolhida. É necessário definir o k da validação cruzada e o hiperparâmetro C da SVM, bem como o intervalo $[\gamma_{min}, \gamma_{max}]$ de γ, onde a busca em grade (Seção 2.1.4) será realizada. Ainda, deve-se fornecer o número N de gerações e o tamanho P da população, parâmetros utilizados na Seção 3.1.2, assegurando que as duas abordagens perfaçam o mesmo número de SVMs realizadas.

Dentro do fluxo da busca em grade, calcula-se o número $E = NP$ de execuções, que será utilizado na construção do conjunto U. O conjunto U é um conjunto com E números igualmente espaçados em uma escala logarítmica (similar a uma progressão geométrica) dentro do intervalo $[\gamma_{min}, \gamma_{max}]$, que representarão cada valor testado para γ do *kernel* RBF. Em seguida, para cada $\gamma_i \in U$, a tripla (k, C, γ_i) de parâmetros é submetida a um processo de aprendizado de máquina (Seção 2.1).

O processo de aprendizado de máquina é composto por outros quatro subprocessos: validação cruzada k, transformação dos dados, SVM e métricas. O processo de validação cruzada k se refere a uma validação cruzada estratificada de k conjuntos (Seção 2.1.3), onde $k-1$ subconjuntos são utilizados para transformação e aprendizado dos parâmetros de transformação dos dados (Seção 2.2), bem como o treino da SVM. O conjunto residual, então, é transformado com o uso dos parâmetros de transformação aprendidos e são utilizados para validação. Este processo tenta preservar a proporção entre as classes (classes desbalanceadas) e é repetido k vezes, onde a métrica final é a média das métricas dos subconjuntos de validação.

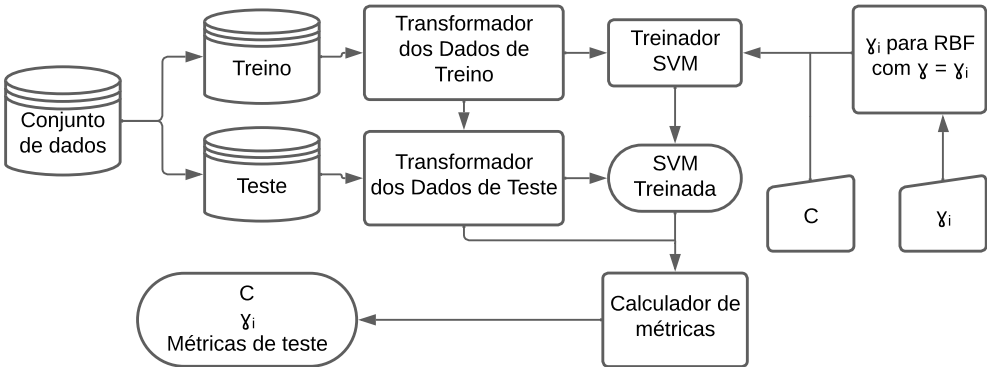
Figura 5 – Fluxo de treino para o *kernel* RBF.



Fonte: De autoria própria.

Várias métricas de AM podem ser implementadas, bem como métricas específicas da SVM. Neste trabalho, optou-se pelas métricas de acurácia balanceada (BAS, Seção 2.1.2), número normalizado de iterações para convergência (ITR_n, Seção 2.3.5) e número normalizado de vetores de suporte (NSV_n, Seção 2.3.6), por serem estimativas de, respectivamente, assertividade, velocidade de convergência e capacidade de generalização. Ao término do aprendizado de máquina para (k, C, γ_i), as métricas associadas são retornadas à busca em grade. Ao exaurir todos os γ_i, o processo de busca em grade se encerra retornando o conjunto de resultados $R_{RBF} = \{(C, \gamma_i, M_i)\}$, onde C é o hiperparâmetro de regularização da SVM, γ_i representa o *kernel* RBF com $\gamma = \gamma_i$ e M_i são as métricas de treino associadas a γ_i.

Figura 6 – Fluxo de teste para o *kernel* RBF.



Fonte: De autoria própria.

Em seguida, cada (C, γ_i) é submetido ao fluxo de teste mostrado na Figura 6. Neste fluxo, todo o conjunto de treino é utilizado para a transformação e aprendizado dos parâmetros de transformação dos dados de treino. O conjunto de teste, outrora esquecido no fluxo de treino, é transformado utilizando os parâmetros de transformação aprendidos. Ao mesmo tempo, um *kernel* RBF de parâmetro γ_i e o hiperparâmetro C são preparados para o treino da SVM.

O treinador SVM utiliza os dados de treino transformados, bem como C e o *kernel* RBF associado a γ_i, treinando um modelo que faz a inferência nos dados de teste transformados baseado nos parâmetros de transformação anteriormente aprendidos. As métricas implementadas no fluxo de treino deverão ser as mesmas utilizadas no fluxo de teste, resultando em um conjunto $\bar{R}_{RBF} = \{(C, \gamma_i, \bar{M}_i)\}$, onde $\bar{M}_i$ são as métricas de teste associadas a γ_i. Os resultados dos fluxos de treino e teste são combinados em um único conjunto $H_{RBF} = \{(C, \gamma_i, M_i, \bar{M}_i)\}$, finalizando o ciclo de avaliação do *kernel* RBF para o conjunto de dados utilizado.

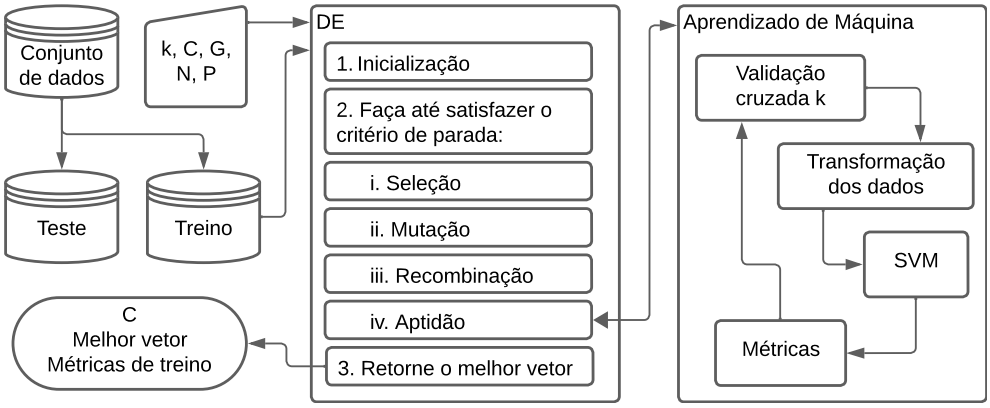
3.1.2 *Kernel* racional

A Figura 7 mostra o fluxo de treino para o *kernel* racional. No início do fluxo, deve-se definir os parâmetros: k para uso na validação cruzada estratificada de k conjuntos, o hiperparâmetro C de regularização da SVM, o número de coeficientes G dos polinômios, bem como o número N de gerações e o número P de indivíduos na população na DE. De maneira equivalente à base de comparação (Seção 3.1.1), o conjunto de dados foi dividido em subconjuntos de treino e teste pelo método *holdout*, onde o conjunto de treino será extensivamente utilizado nas realizações da função de aptidão da DE.

Há de se notar que o número G se refere a quantidade de coeficientes em cada uma das funções polinomiais $P(x)$ e $Q(x)$ que definem a função racional $R(x) = P(x)/Q(x)$ (Seção 2.4), ou seja, o grau de $P(x)$ ou $Q(x)$ é igual a $G - 1$. Aqui, a função de *kernel*

utilizada será uma função $f(x)$ real de uma variável composta com a função $g(x_i, x_j)$, uma função real de duas variáveis que será a função de distância euclidiana, a mesma função utilizada no argumento da exponencial que define o *kernel* RBF. Os coeficientes de $P(x)$ e $Q(x)$ serão otimizados utilizando a DE, com valores limitados pelo intervalo real fechado $[-1, 1]$. A dimensão dos indivíduos será $2G$, G coeficientes para $P(x)$ e G coeficientes para $Q(x)$.

Figura 7 – Fluxo de treino para o *kernel* proposto.



Fonte: De autoria própria.

Seguindo o pseudo-código da DE mostrado na Seção 2.5.2, a otimização começa inicializando P indivíduos de dimensão $2G$ com entradas aleatórias no intervalo $[-1, 1]$. Depois, repete-se, até que o número N de gerações seja alcançado, as quatro operações básicas do algoritmo: seleção, mutação, recombinação e aptidão. A seleção e mutação seguem o que define a variante "DE/rand/1/bin", que seleciona três vetores aleatórios para criar um vetor mutante utilizando um dos vetores somado com a subtração dos outros dois vetores selecionados. Um novo vetor de teste é gerado pela recombinação binomial utilizando cada indivíduo e o vetor mutante associado.

A aptidão do indivíduo é calculada com base no resultado do processo de aprendizado de máquina com a função de *kernel* que o indivíduo representa. Seja I um indivíduo da população de dimensão $2G$ e d a dimensão do espaço de características do conjunto de dados, então o *kernel* que este indivíduo representa pode ser construído como se segue:

$$f_I(x) = \frac{\sum_{i=0}^{G-1} I_i x^i}{\sum_{i=G}^{2G-1} I_i x^{i-G}} \tag{40}$$

$$g(u, v) = \sqrt{\sum_{i=0}^{d-1} (u_i - v_i)^2} \tag{41}$$

$$K_I(u, v) = (f_I \circ g)(u, v). \tag{42}$$

De maneira análoga à base de comparação, este *kernel* é avaliado no processo de aprendizado de máquina, se diferenciando ao receber a tripla $(k, C, K_I(x_i, x_j))$ para direcionar

os processos de validação cruzada (estratificada) com k subconjuntos, transformação dos dados, treino do modelo e retorno das métricas.

O processo de aprendizado de máquina retorna as mesmas três métricas utilizadas na base de comparação: acurácia balanceada (BAS, entre 0 e 1), número normalizado de iterações para convergência (ITR_n, entre 0 e 1) da SVM e número normalizado de vetores de suporte (NSV_n, entre 0 e 1) utilizados na convergência da SVM. A métrica ITR_n é definida como

$$ITR_n = \frac{ITR}{ITR_{max}}, \tag{43}$$

onde ITR_{max} é a quantidade de iterações máxima permitida para o treino da SVM. Como ITR é um número positivo limitado a ITR_{max}, ITR_n estará entre 0 e 1. A métrica NSV_n é definida como

$$NSV_n = \frac{NSV}{|X_{tr}| \times \frac{k-1}{k}}, \tag{44}$$

onde k é a quantidade de partições na validação cruzada e $|X_{tr}|$ é o tamanho do conjunto de treino. Em cada treino da validação cruzada, no máximo $|X_{tr}| \times \frac{k-1}{k}$ vetores podem ser vetores de suporte e, como NSV também é positivo, NSV_n necessariamente estará entre 0 e 1.

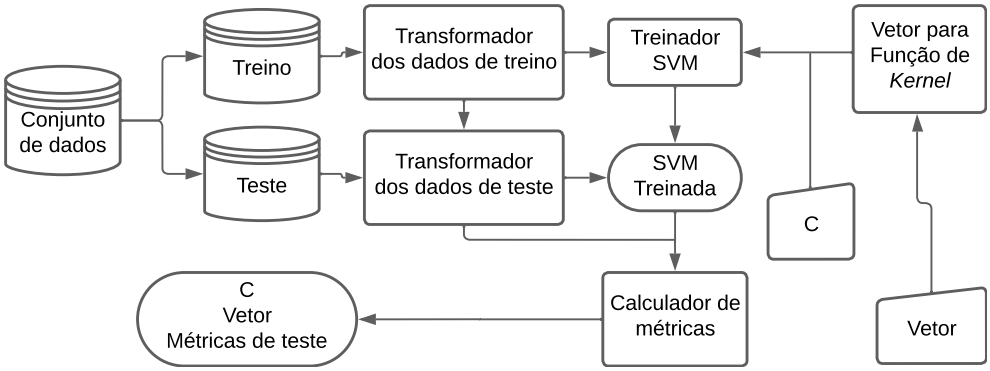
Definido como um problema de minimização, a ideia é tornar mais apto os indivíduos que tenham maior acurácia em treino, ao passo que baixo ITR_n, modelos com rápida convergência, e baixo NSV_n, modelos com boa capacidade de generalização e baixa complexidade. O cálculo da aptidão utiliza essas três métricas e é definido como

$$Aptidão(I) = -BAS + \frac{k}{|X_{tr}|}(ITR_n + NSV_n). \tag{45}$$

O peso de $(ITR_n + NSV_n)$ foi definido para que seja mais valioso aumentar acurária que reduzir ITR_n e NSV_n, otimizados de forma secundária com importância na ordem do impacto de um acerto em relação ao conjunto de validação. Ao fim da DE, $R_{RKF} = (C, I_b, M_{I_b})$ é retornado, onde C é o hiperparâmetro de regularização da SVM, I_b é o melhor vetor da busca e M_{I_b} são as métricas de treino associadas a I_b.

Em seguida, o hiperparâmetro C e o vetor I_b são introduzidos no fluxo ilustrado na Figura 8, para o cálculo das métricas de teste. Os dados de treino são transformados e usados para aprender os parâmetros de transformação empregados na transformação dos dados de teste. O vetor I_b passa pela transformação em uma função de *kernel* que, associada aos dados de treino, será utilizada no treinador SVM. O modelo treinado faz a inferência nos dados de teste e as mesmas três métricas de treino são calculadas, retornando a tripla $\bar{R}_{RKF} = (C, I_b, \bar{M}_{I_b})$, onde $\bar{M}_{I_b}$ representa as métricas de teste associadas ao vetor I_b. A quádrupla $H_{RKF} = (C, I_b, M_i, \bar{M}_i)$ pode ser definida, representando as métricas de treino e teste para o vetor I_b referente ao treino da SVM com regularização C.

Figura 8 – Fluxo de teste para o *kernel* proposto.



Fonte: De autoria própria.

3.2 IMPLEMENTAÇÃO

Os experimentos foram realizados em um computador portátil da Lenovo *PC International*, modelo IdeaPad 3 15ALC6 82MF0004BR, com processador *Ryzen* 7 5700U, 20 GB de memória *RAM* (4 GB + 16 GB) e sistema operacional *Windows* 11 *Home Single Language*, versão 22H2 e compilação 22621-2361. O computador portátil possui oito núcleos e 16 *threads* e, quando possível, os algoritmos foram executados em múltiplos processos para paralelizar as realizações com intenção de reduzir o tempo de execução. O código-fonte se encontra no repositório https://github.com/mrr00b00t/tcc.

A metodologia foi implementada na linguagem de programação *python* (VAN ROSSUM; DRAKE *et al.*, 1995) versão 3.10.11. A biblioteca *scikit-learn* (PEDREGOSA *et al.*, 2011) versão 1.3.0 foi utilizada para calcular a distância euclidiana, os métodos de validação, a transformação de dados, o modelo SVM e as métricas de AM. A biblioteca *pymoo* (BLANK; DEB, 2020) versão 0.6.0.1 foi utilizada para o algoritmo DE. A biblioteca *pmlb* (ROMANO *et al.*, 2021) versão 1.0.1.*post*3 foi utilizada para adquirir o conjunto de dados curado em tempo de execução.

Para reduzir a complexidade computacional da realização de uma função racional na implementação, a estratégia consiste em minimizar o número de multiplicações e adições. Uma forma de conseguir isso é explorar a estrutura e os coeficientes do polinômio com o método de Horner (PAN, 1966). Para uma função racional genérica

$$R(x) = \frac{P(x)}{Q(x)} = \frac{a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0}{b_m x^m + b_{m-1} x^{m-1} + \cdots + b_1 x + b_0}, \tag{46}$$

o método de Horner consiste em aninhar as operações, reduzindo o número total de multiplicações e adições necessárias, como mostra o exemplo

$$P(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + a_n x) \cdots)), \tag{47}$$

que requer n multiplicações e n adições para um polinômio de grau n. O método descrito foi utilizado tanto no polinômio $P(x)$ quanto no polinômio $Q(x)$.

3.2.1 Conjunto de dados

Para testar as abordagens, utilizou-se o conjunto de dados de diabetes dos índios Pima (SMITH *et al.*, 1988; PIMA. . . , 2016), construído pelo Instituto Nacional de Diabetes e Doenças Digestivas e Renais (NIDDK) dos Estados Unidos e curado pelo *Penn Machine Learning Benchmarks* (ROMANO *et al.*, 2021), contendo dados de 768 mulheres de uma população nas proximidades de *Phoenix*, Arizona. Portanto, trata-se de um conjunto de dados com variável binária de predição, em que a saída esperada representa prevalência de diabetes, onde 500 casos testaram negativo e outros 258 testaram positivo. As classes são notoriamente desbalanceadas ($\frac{258}{500} \neq 1$), o que justifica o uso da validação estratificada e acurácia balanceada.

Tabela 1 – Análise descritiva do conjunto de dados Pima.

Estatística	**NG**	**TOTG**	**IS**	**EPT**	**Id**	**PSD**	**IMC**	**ERD**
Média	3,84	120,89	79,80	20,54	33,24	69,10	31,99	0,471
Variância	11,35	1022,25	13281,18	254,47	138,30	374,65	62,16	0,110
Mínimo	0,00	0,00	0,00	0,00	21,00	0,00	0,00	0,078
1º Quartil	1,00	99,00	0,00	0,00	24,00	62,00	27,30	0,244
2º Quartil	3,00	117,00	30,50	23,00	29,00	72,00	32,00	0,372
3º Quartil	6,00	140,25	127,25	32,00	41,00	80,00	36,60	0,626
Máximo	17,00	199,00	846,00	99,00	81,00	122,00	67,10	2,42

Fonte: De autoria própria

O conjunto de dados contém oito atributos (características) que são utilizados para aprender a função de decisão: número de gestações (NG), teste oral de tolerância à glicose (TOTG), insulina sérica (IS), espessura da pele do tríceps (EPT), idade (Id), pressão sanguínea diastólica (PSD), índice de massa corporal (IMC) e estimativa de risco à diabetes baseada no histórico familiar (ERD), descritos na Tabela 1. A população Pima têm sido investigada pelo NIDDK em intervalos de dois anos desde 1965 por terem sobrevivido de uma dieta pobre em carboidratos devido a uma predisposição genética e, pela súbita troca da sua alimentação tradicional por comida processada, apresentam um alto índice de diabetes.

3.2.2 Parâmetros experimentais

Foram utilizados os seguintes parâmetros experimentais: validação *holdout* estratificada de 80% dos dados para treino e 20% para teste, validação cruzada estratificada de $k = 5$ conjuntos nos dados de treino, hiperparâmetros de regularização $C = \{0{,}03125, 0{,}125, 0{,}5, 2, 8, 32, 128, 512, 2048\}$, $\gamma_{min} = 10^{-8}$, $\gamma_{max} = 10^{4}$ e números de coeficientes de cada polinômio da função racional $G = \{3, 4, 5\}$. A quantidade de

iterações possíveis para convergência da SVM foi limitada ao valor $ITR_{max} = 1800$, com intenção de reduzir o tempo das experimentações em todos os casos.

A variante DE escolhida foi a "DE/rand/1/bin", com taxa de recombinação $CR = 0{,}9$, fator de escala $F = 0{,}8$, tamanho da população $P = 30$ e número de gerações $N = 40$. Cada combinação experimental (C e grau da função de *kernel* racional ou C e γ do *kernel* RBF) foi executada 60 vezes, onde cada execução utilizou uma semente aleatória distinta do intervalo $[2, 99999)$ para particionar os dados pelo uso da validação *holdout* estratificada.

4 RESULTADOS

A Tabela 2 apresenta a média dos cinco melhores resultados das diversas configurações experimentadas para o *kernel* RBF, ordenados pelas acurácia balanceada no conjunto de treino. As seguintes colunas são apresentadas: C, γ, acurácia balanceada nos dados de treino (BAS^{tr}), número normalizado de iterações nos dados de treino (ITR_n^{tr}), número normalizado de vetores de suporte no conjunto de treino (NSV_n^{tr}), acurácia balanceada nos dados de teste, número normalizado de iterações nos dados de teste (ITR_n^{te}) e número normalizado de vetores de suporte no conjunto de teste (NSV_n^{te}).

Tabela 2 – Cinco melhores configurações para o *kernel* RBF

Posição	C	γ	BAS^{tr}	ITR_n^{tr}	NSV_n^{tr}	BAS^{te}	ITR_n^{te}	NSV_n^{te}
1º	2048	0,000108	0,724423	0,807496	0,524519	0,717407	0,895833	0,524349
2º	2048	0,000094	0,724386	0,741244	0,525258	0,716849	0,837556	0,524919
3º	2048	0,000096	0,724326	0,758091	0,525123	0,717157	0,831139	0,524837
4º	2048	0,000098	0,724325	0,764593	0,524926	0,716991	0,840926	0,524864
5º	512	0,000461	0,724320	0,788019	0,524817	0,716343	0,895454	0,525000

Fonte: De autoria própria

Para cada configuração, os parâmetros C e γ usados no modelo são anotados. A configuração de modelo mais performática (**1º**) com o *kernel* RBF utiliza o $C = 2048$ e $\gamma = 0{,}000108$. Essa configuração alcançou a acurácia balanceada de 72,4423% e 71,7407% nos conjuntos de treino e teste, respectivamente, requisitando 80,7496% e 89,5833% da quantidade máxima de iterações em treino e teste, respectivamente, e empregando 52,4519% e 52,4349% dos vetores de suporte disponíveis em treino e teste, respectivamente. Essa configuração foi escolhida como a melhor e utilizada como base de comparação para a melhor configuração resultante dos experimentos para o *kernel* racional.

A Tabela 2 também permite uma comparação entre as configurações. Por exemplo, a quinta melhor configuração (**5º**) usou um $C = 512$, o que é menor que as outras configurações, porém utilizou um $\gamma = 0{,}000461$, que é maior que os demais. Também podemos observar que as métricas de treino e teste se comportaram de maneira relativamente estável, apresentando uma faixa compatível de valores ao longo das melhores configurações.

Tabela 3 – Cinco melhores configurações para o *kernel* racional

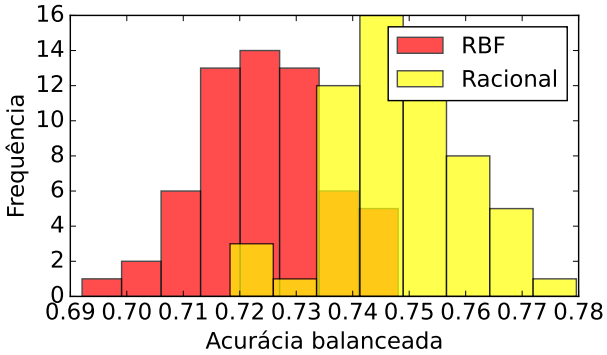
Posição	C	G	F	BAS^{tr}	ITR_n^{tr}	NSV_n^{tr}	BAS^{te}	ITR_n^{te}	NSV_n^{te}
1º	32	3	0,743240	0,748332	0,114598	0,510708	0,706577	0,142065	0,504207
2º	8	4	0,742657	0,748237	0,137626	0,547639	0,707015	0,169889	0,540798
3º	2048	3	0,742835	0,748059	0,130409	0,511128	0,705469	0,162685	0,506053
4º	32	4	0,742285	0,748039	0,170072	0,536544	0,702574	0,207565	0,530646
5º	128	3	0,742105	0,747352	0,126806	0,517509	0,708389	0,158657	0,516069

Fonte: De autoria própria

As melhores configurações para o *kernel* racional podem ser visualizadas na Tabela 3, e são ordenadas pela média de sua aptidão (F) no conjunto de dados de treino. A configuração de modelo mais performática do *kernel* racional utiliza $C = 32$ e $G = 3$, com aptidão $F = 0{,}748332$, que conseguiu atingir acurácia balanceada de 74,8332% e 70,6577% nos conjuntos de treino e teste, respectivamente, utilizando 11,4598% e 14,2065% da quantidade máxima de iterações permitidas em treino e teste, respectivamente, operando com 51,0708% e 50,4207% dos vetores de suporte disponíveis em treino e teste, respectivamente. Essa configuração foi escolhida como a melhor representante para ser comparada com o melhor resultado do *kernel* RBF.

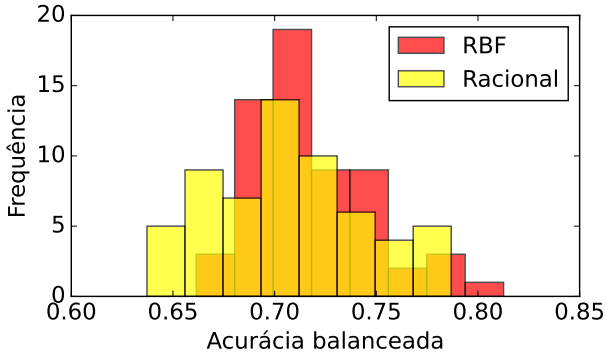
Também é possível notar na Tabela 3 que as melhores configurações apresentam maior diversidade no parâmetro C, diferentemente do *kernel* RBF, o que pode indicar que as funções racionais são flexíveis o suficiente para operar em diferentes parâmetros de regularização. Além disso, nenhuma das configurações da Tabela 3 apresenta $G = 5$, o que pode indicar que uma função racional com mais coeficientes não foi necessária ou que não foi possível otimizar os seus coeficientes eficientemente, necessitando alterações em relação aos parâmetros da DE.

Figura 9 – Histograma da BAS em treino



Fonte: De autoria própria.

Figura 10 – Histograma da BAS em teste

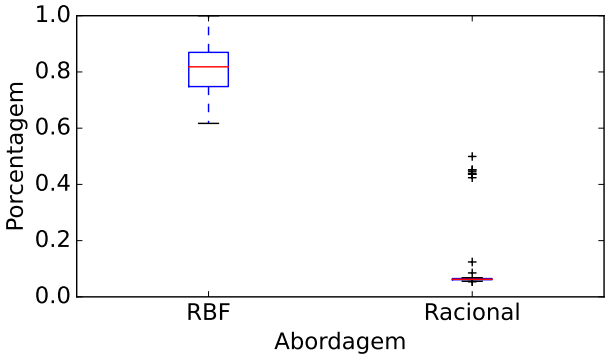


Fonte: De autoria própria.

As Figuras 9 e 10 apresentam a distribuição da acurácia balanceada (BAS) em cenários de treino e teste para os 60 experimentos realizados. A Figura 9 mostra a distribuição de cada abordagem para a métrica BAS no conjunto de treino (BAS^{tr}), enquanto a Figura 10 apresenta a distribuição de cada abordagem utilizando os dados de teste (BAS^{te}).

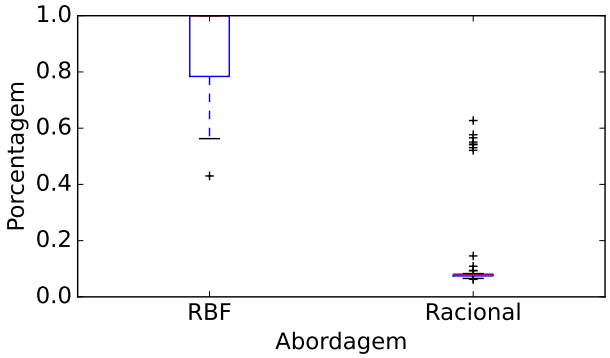
Ao comparar as melhores configurações de cada abordagem, percebe-se que o *kernel* racional tem média BAS^{tr} maior que o *kernel* RBF, porém com média BAS^{te} menor, o que pode ser um indício de sobreajuste, como pode ser verificado nas Tabelas 2 e 3. Contudo, nota-se que as Figuras 9 e 10 mostram sobreposições nas distribuições, menos evidente no conjunto de treino (Figura 9), e com maior intensidade no conjunto de teste (Figura 10).

Figura 11 – *Boxplot* do ITR_n em treino



Fonte: De autoria própria.

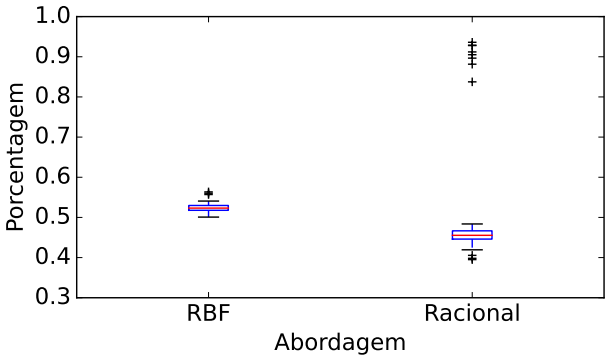
Figura 12 – *Boxplot* do ITR_n em teste



Fonte: De autoria própria.

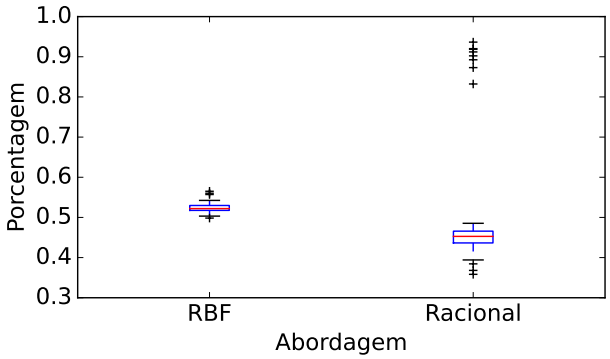
As Figuras 11 e 12 mostram os *boxplots* para as métricas de número normalizado de iterações normalizado para os dados de treino e teste, respectivamente, para os 60 experimentos. Já as Figuras 13 e 14 representam os *boxplots* para as métricas de número normalizado de vetores de suporte para os conjuntos de treino e teste, respectivamente, também nos 60 experimentos realizados.

Figura 13 – *Boxplot* do NSV_n em treino



Fonte: De autoria própria.

Figura 14 – *Boxplot* do NSV_n em teste



Fonte: De autoria própria.

Apesar de alguns *outliers*, o *kernel* racional aparenta ter métricas ITR_n^{tr}, ITR_n^{te}, NSV_n^{tr} e NSV_n^{te} superiores ao *kernel* RBF, hipótese melhor embasada através de uma análise prévia das medianas nas Figuras 11, 12, 13 e 14. Valores baixos de ITR_n indicam maior capacidade de convergência e menor tempo de treinamento, enquanto que valores baixos de NSV_n indicam modelos menos complexos e mais rápidos na inferência, já que os vetores de suporte definem a função de decisão das SVMs.

Não obstante, é necessário verificar os resultados por meio de testes para mensurar a relevância estatística da diferença entre as distribuições dos resultados experimentais. Como se trata de duas abordagens diferentes aplicadas ao mesmo conjunto de dados, torna-se possível o uso de testes pareados, em especial dois deles: o Teste t Pareado, quando as duas distribuições comparadas aparentam ter origem em distribuições normais,

e o teste de Wilcoxon, quando não há garantia de que as distribuições comparadas tem origem em distribuições normais. O teste de normalidade de Shapiro-Wilk foi utilizado para investigar a normalidade das distribuições.

Tabela 4 – Resultado dos testes de normalidade de Shapiro-Wilk

Abordagem	BAS^{tr}	ITR_n^{tr}	NSV_n^{tr}	BAS^{te}	ITR_n^{te}	NSV_n^{te}
RBF	Sim	Sim	Não	Não	Não	Não
Racional	Sim	Não	Não	Sim	Não	Não

Fonte: De autoria própria

Todos os testes foram executados com um nível de significância de 0,05, o que significa dizer que a chance de rejeição da hipótese nula quando ela é verdadeira é de 5%. A Tabela 4 mostra os resultados do teste de normalidade Shapiro-Wilk. A acurácia balanceada no conjunto de treino é a única métrica onde as abordagens têm distribuições aparentemente normais. As demais métricas apresentam pelo menos uma distribuição na qual existe evidência suficiente para rejeitar a hipótese nula, isto é, induzir que a abordagem não tem distribuição procedente de uma distribuição normal.

Tabela 5 – Resultado dos testes pareados de diferença no conjunto de teste

1ª Distribuição	**2ª Distribuição**	**Teste**	**Resultado**
RBF BAS^{tr}	Racional BAS^{tr}	Teste t Pareado	Estatisticamente Diferentes
RBF ITR_n^{tr}	Racional ITR_n^{tr}	Wilcoxon	Estatisticamente Diferentes
RBF NSV_n^{tr}	Racional NSV_n^{tr}	Wilcoxon	Estatisticamente Diferentes
RBF BAS^{te}	Racional BAS^{te}	Wilcoxon	Estatisticamente Iguais
RBF ITR_n^{te}	Racional ITR_n^{te}	Wilcoxon	Estatisticamente Diferentes
RBF NSV_n^{te}	Racional NSV_n^{te}	Wilcoxon	Estatisticamente Diferentes

Fonte: De autoria própria

A Tabela 5 mostra os resultados dos testes para as distribuições pareadas em cada métrica. Com exceção da métrica BAS^{te}, todas as outras métricas demonstraram uma diferença estatisticamente significante entre as distribuições do *kernel* RBF e *kernel* racional. A análise inicial acerca das distribuições BAS^{tr} e BAS^{te} (Figura 9 e Figura 10) se confirma: há evidência suficiente para induzir que a acurácia balanceada em treino é maior na abordagem do *kernel* racional, enquanto que a mesma métrica em teste se mostra equivalente ao *kernel* RBF em termos estatísticos.

Os resultados dos testes também indicam que as métricas ITR_n^{tr} e ITR_n^{te} têm distribuições estatisticamente diferentes, e a hipótese de que ITR_n^{tr} e ITR_n^{te} são menores no *kernel* racional, levantada previamente pela análise das medianas das Figuras 11 e 12, pode ser confirmada. O mesmo ocorre com as métricas NSV_n^{tr} e NSV_n^{te}, os testes apontam que as distribuições das duas abordagens são estatisticamente diferentes e a hipótese de que NSV_n^{tr} e NSV_n^{te} são menores no *kernel* racional pode ser considerada.

Estatisticamente, há evidências para aceitar que a abordagem do *kernel* racional conseguiu uma métrica de acurácia melhor em treino, porém isso não significou um aumento expressivo na acurácia balanceada em teste. Ainda sim, as métricas de convergência (ITR_n) e capacidade de generalização (NSV_n) são melhores no *kernel* racional tanto em treino quanto em teste, quando comparadas ao *kernel* RBF. Mesmo com o indício de sobreajuste indicado pela acurácia de treino e teste para o *kernel* racional, a análise mostra que a abordagem não tem uma diferença estatisticamente significativa quando sua assertividade é comparada ao *kernel* RBF.

5 CONCLUSÃO

Neste trabalho foi proposto uma metodologia automatizada para construir funções de *kernel* baseadas em funções racionais, isto é, funções que são definidas pelo quociente de dois polinômios. Para materializar essas funções, os seus coeficientes foram otimizados utilizando otimização não convexa por meio de um algoritmo evolucionário, chamado Evolução Diferencial (DE) e guiados por uma função de aptidão, que combina tanto a acurácia balanceada quanto o número de iterações para convergência e o número de vetores de suporte do modelo.

Também foi possível investigar o potencial das funções de *kernel* racionais em vários cenários de hiperparâmetros através de experimentos e simulações. Em especial, os experimentos com a mesma configuração foram conduzidos múltiplas vezes com intuito de aumentar a confiança das análises. Uma função de aptidão foi desenvolvida envolvendo as métricas de acurácia balanceada, número de iterações para convergência da SVM e número de vetores de suporte da SVM, além de um estudo de caso com o conjunto de dados Pima.

A análise dos resultados mostrou que há evidência suficiente para induzir que o objetivo principal deste trabalho foi atingido, ou seja, que é possível o desenvolvimento de uma metodologia automática para o aprendizado de funções de *kernel* modeladas como funções racionais, com o emprego da DE na otimização de seus coeficientes.

O presente trabalho também demonstra a aplicabilidade transversal de domínios diversos do conhecimento que foram absorvidos durante o curso. Foi possível associar tanto conteúdos mais basilares da formação, como álgebra, aproximação e cálculo numérico, como conhecimentos mais sofisticados, que é o caso da otimização não convexa pelo uso das meta-heurísticas de busca, aprendizado de máquina e ciência de dados.

É preciso notar o potencial da metodologia: com a otimização dos coeficientes nas funções racionais de *kernel*, é possível abstrair uma morfologia específica da função de *kernel*. Isto é, funções racionais podem aproximar, dada a precisão configurada pela sua quantidade de coeficientes, outras funções de *kernel* da literatura ou até encontrar funções novas e que não são catalogadas, eximindo o usuário de experimentar várias funções de *kernel* distintas.

Finda-se, portanto, a função de *kernel* modelada por funções racionais como uma possibilidade no contexto de funções de *kernel* em SVMs, que pode ser considerada na medida em que outras funções são utilizadas pelos praticantes da SVM. No estudo de caso, a DE foi útil para ajustar os coeficientes do *kernel* racional, frequentemente considerada uma substituta à busca randômica ou busca em grade na otimização de funções de *kernel* tradicionais.

5.1 LIMITAÇÕES

Apesar de mostrar seu potencial, a metodologia proposta possui algumas limitações. A primeira é que a metodologia de aprendizado automatizado de funções de *kernel* proposta não considerou a introdução de um mecanismo de tratamento dos dados, apenas de transformação. Tratar os dados é essencial pois conjuntos de dados reais frequentemente apresentam dados discrepantes, nulos ou até faltantes.

Outro ponto é que as funções de *kernel* aprendidas, teoricamente, podem não ser funções de *kernel* válidas segundo as condições de Mercer. Como as condições não são restrições da otimização da proposta, não há garantia de que as funções aprendidas de fato correspondem a um produto interno em um espaço de dimensão mais alta, isto é, um *kernel* propriamente dito.

Pode-se pontuar, também, a limitação na escolha de $g(x_i, x_j)$ como a distância euclidiana. A distância euclidiana pode não ser uma métrica universalmente adequada a conjuntos de dados reais. Ainda, a função de aptidão também pode não ser a melhor adequada para casos gerais, tampouco a combinação das métricas envolvidas por serem, de certa forma, métricas não contínuas.

Por fim, o estudo de caso tão somente analisa a capacidade da metodologia proposta no conjunto de dados testado. Expandir o leque de conjuntos de dados nos experimentos se mostra essencial para uma análise transversal da efetividade da metodologia em encontrar funções de *kernel* adequadas em cada contexto.

5.2 TRABALHOS FUTUROS

Um estágio de tratamento e limpeza de dados é um precursor crítico para a implantação de algoritmos de AM, especialmente os automáticos, como a proposta deste trabalho. Somente a etapa de transformação dos dados não é suficiente, dados crus geralmente apresentam inconsistências, erros, valores faltantes e potenciais *outliers* que, se não forem tratados, podem resultar em resultados não confiáveis e enganosos (BATISTA; MONARD, 2003). Por exemplo, há 227 valores iguais a zero nas instâncias da coluna que representa a largura da pele do tríceps do conjunto de dados Pima, o que é fisicamente impossível. Portanto, a construção de um estágio de limpeza de dados se mostra necessária em trabalhos futuros.

O teorema de Mercer fornece as condições necessárias e suficientes para que uma função seja um *kernel* válido, principalmente garantindo que a matriz de Gram associada é simétrica e positiva semidefinida. Isto é fundamental para que a função seja um mapeamento implícito de um produto interno em um espaço de dimensão maior. Além disso, o teorema assegura que o problema de otimização intrínseco do treino das SVMs é convexo, isto é, solucionável com eficiência (VAPNIK, 1998; SCHÖLKOPF; SMOLA, Alexander J, 2002). Sem tal garantia, o processo de treino das SVMs pode não convergir e levar a um

desempenho imprevisível. Há espaço para que pesquisas subsequentes implementem testes das condições de Mercer e garantir que mapeamento implícito existe.

Métricas de distância são outra parte essencial na modelagem e interpretação de fenômenos empíricos encapsulados em dados do mundo real. A distância euclidiana, apesar de difundida, não é invariavelmente alinhada com todos as estruturas encontradas em multifacetados conjuntos de dados. Por exemplo, a distância de Minkowski, considerada como uma generalização da distância euclidiana e da distância de Manhattan, oferece flexibilidade em capturar várias relações espaciais e geométrica entre pontos, tornando-o uma ferramenta valiosa em AM e análise espacial (DE AMORIM; MIRKIN, 2012). Continuações do estudo podem explorar a viabilidade de outras distâncias como substituta da euclidiana, utilizada neste trabalho.

A necessidade da definição de uma função de aptidão representar bem o espaço de aptidão é ressaltada quando se considerada a complexidade e natureza estocástica das EAs, especialmente as DEs. A função de aptidão bem estruturada, portanto, acomoda o ambiente dinâmico e incerto que as EAs operam, ao passo que oferece uma medida de avaliação consistente e confiável que se mantém invariante à natureza do processo evolucionário (BACK; HAMMEL; SCHWEFEL, 1997). Deste modo, uma função de aptidão mais adequada e que combine métricas realmente contínuas sugere um tópico de investigação a serem explorados em estudos posteriores. Além disso, faz-se necessário investigar o indício de sobreajuste e penalizar a aptidão do indivíduo por meio de abordagens de compensação inspirados em *early stopping* ou técnicas similares.

A significância de avaliar uma abordagem de AM em uma variedade de conjuntos de dados não pode ser subjugada, pois é fundamental para determinar a generalização e a robustez da metologia. Ao interagir com diversos conjuntos de dados, é possível discernir o desempenho e a confiança da metodologia em diferentes contextos, consequentemente garantindo que as soluções concebidas são plenamente aplicáveis e não estritamente confinadas a distribuição de dados específicas (HASTIE *et al.*, 2009). Prospectos de pesquisa futura podem investigar e até confirmar a aplicabilidade da metodologia proposta neste trabalho para uma variada gama de conjuntos de dados.

REFERÊNCIAS

AYAT, Nedjem-Eddine; CHERIET, Mohamed; SUEN, Ching Y. Automatic model selection for the optimization of SVM kernels. **Pattern Recognition**, Elsevier, v. 38, n. 10, p. 1733–1745, 2005.

BACK, Thomas; HAMMEL, Ulrich; SCHWEFEL, H-P. Evolutionary computation: Comments on the history and current state. **IEEE transactions on Evolutionary Computation**, IEEE, v. 1, n. 1, p. 3–17, 1997.

BAKER, George A; GRAVES-MORRIS, Peter. Padé approximants second edition. **(No Title)**, Cambridge University Press, 1996.

BATISTA, Gustavo EAPA; MONARD, Maria Carolina. An analysis of four missing data treatment methods for supervised learning. **Applied artificial intelligence**, Taylor & Francis, v. 17, n. 5-6, p. 519–533, 2003.

BAZARAA, Mokhtar S; SHERALI, Hanif D; SHETTY, Chitharanjan M. **Nonlinear programming: theory and algorithms**. [*S.l.*]: John wiley & sons, 2013.

BEAVIS, Brian; DOBBS, Ian. **Optimisation and stability theory for economic analysis**. [*S.l.*]: Cambridge university press, 1990.

BERGSTRA, James; BENGIO, Yoshua. Random search for hyper-parameter optimization. **Journal of machine learning research**, v. 13, n. 2, 2012.

BISHOP, Christopher M; NASRABADI, Nasser M. **Pattern recognition and machine learning**. [*S.l.*]: Springer, 2006. v. 4.

BLANK, J.; DEB, K. pymoo: Multi-Objective Optimization in Python. **IEEE Access**, v. 8, p. 89497–89509, 2020.

BLUM, Christian; ROLI, Andrea. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, Acm New York, NY, USA, v. 35, n. 3, p. 268–308, 2003.

BOTTOU, Léon. Large-scale machine learning with stochastic gradient descent. *In*: SPRINGER. PROCEEDINGS of COMPSTAT'2010: 19th International Conference on

Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers. [*S.l.*: *s.n.*], 2010. P. 177–186.

BOYD, Stephen P; VANDENBERGHE, Lieven. **Convex optimization**. [*S.l.*]: Cambridge university press, 2004.

BREZINSKI, C. An introduction to Padé approximations. **Curves and Surfaces in Geometric design**, v. 1, p. 59–65, 1994.

BULTHEEL, Adhemar; VAN BAREL, Marc *et al.* Orthogonal basis functions in discrete least-squares rational approximation. **Journal of Computational and Applied Mathematics**, Elsevier, v. 164, p. 175–194, 2004.

BURGES, Christopher JC. A tutorial on support vector machines for pattern recognition. **Data mining and knowledge discovery**, Springer, v. 2, n. 2, p. 121–167, 1998.

CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. **Machine learning**, Springer, v. 20, p. 273–297, 1995.

CRISTIANINI, Nello; SHAWE-TAYLOR, John. **An introduction to support vector machines and other kernel-based learning methods**. [*S.l.*]: Cambridge university press, 2000.

CROOM, Fred H. **Principles of topology**. [*S.l.*]: Courier Dover Publications, 2016.

DAS, Swagatam; MAITY, Sayan; QU, Bo-Yang;
SUGANTHAN, Ponnuthurai Nagaratnam. Real-parameter evolutionary multimodal optimization—A survey of the state-of-the-art. **Swarm and Evolutionary Computation**, Elsevier, v. 1, n. 2, p. 71–88, 2011.

DE AMORIM, Renato Cordeiro; MIRKIN, Boris. Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering. **Pattern Recognition**, Elsevier, v. 45, n. 3, p. 1061–1075, 2012.

DE JONG, Kenneth. Evolutionary computation: a unified approach. *In*: PROCEEDINGS of the Genetic and Evolutionary Computation Conference Companion. [*S.l.*: *s.n.*], 2017. P. 373–388.

DIOŞAN, Laura; ROGOZAN, Alexandrina; PECUCHET, Jean-Pierre. Improving classification performance of support vector machine by genetically optimising kernel shape and hyper-parameters. **Applied Intelligence**, Springer, v. 36, p. 280–294, 2012.

EIBEN, Agoston E; SMITH, James E; EIBEN, AE; SMITH, JE. Genetic algorithms. **Introduction to Evolutionary Computing**, Springer, p. 37–69, 2003.

EVERITT, Brian S; SKRONDAL, Anders. The Cambridge dictionary of statistics, 2010.

GEISSER, Seymour. **Predictive inference**. [*S.l.*]: CRC press, 1993. v. 55.

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow**. [*S.l.*]: "O'Reilly Media, Inc.", 2022.

GHAHRAMANI, Zoubin. Unsupervised learning. *In*: SUMMER school on machine learning. [*S.l.*]: Springer, 2003. P. 72–112.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome H; FRIEDMAN, Jerome H. **The elements of statistical learning: data mining, inference, and prediction**. [*S.l.*]: Springer, 2009. v. 2.

HERBRICH, Ralf; GRAEPEL, Thore; OBERMAYER, Klaus *et al.* **Regression models for ordinal data: A machine learning approach**. [*S.l.*]: Citeseer, 1999.

HOWLEY, Tom; MADDEN, Michael G. An evolutionary approach to automatic kernel construction. *In*: SPRINGER. INTERNATIONAL Conference on Artificial Neural Networks. [*S.l.*: *s.n.*], 2006. P. 417–426.

HU, Yong; TAO, Vincent; CROITORU, Arie. Understanding the rational function model: methods and applications. **International archives of photogrammetry and remote sensing**, v. 20, n. 6, p. 119–124, 2004.

KOCH, Patrick; BISCHL, Bernd; FLASCH, Oliver; BARTZ-BEIELSTEIN, Thomas; WEIHS, Claus; KONEN, Wolfgang. Tuning and evolution of support vector kernels. **Evolutionary Intelligence**, Springer, v. 5, p. 153–170, 2012.

KOHAVI, Ron *et al.* A study of cross-validation and bootstrap for accuracy estimation and model selection. *In*: MONTREAL, CANADA, 2. IJCAI. [*S.l.*: *s.n.*], 1995. v. 14, p. 1137–1145.

KOLO, Brian. **Binary and multiclass classification**. [*S.l.*]: Lulu. com, 2011.

LEE, Ga Young; ALZAMIL, Lubna; DOSKENOV, Bakhtiyar; TERMEHCHY, Arash. A survey on data cleaning methods for improved machine learning model performance. **arXiv preprint arXiv:2109.07127**, 2021.

LEFSCHETZ, Solomon. **Introduction to topology**. [*S.l.*]: Princeton University Press, 2015.

MANTOVANI, Rafael G; ROSSI, André LD; VANSCHOREN, Joaquin; BISCHL, Bernd; DE CARVALHO, André CPLF. Effectiveness of random search in SVM hyper-parameter tuning. *In*: IEEE. 2015 international joint conference on neural networks (IJCNN). [*S.l.*: *s.n.*], 2015. P. 1–8.

MITCHELL, Tom M. **Machine learning**. [*S.l.*]: McGraw-hill New York, 1997. v. 1.

NASTESKI, Vladimir. An overview of the supervised machine learning methods. **Horizons. b**, v. 4, p. 51–62, 2017.

NEWMAN, DJ; SHAPIRO, HS. Approximation by generalized rational functions. *In*: SPRINGER. ON Approximation Theory/Über Approximationstheorie: Proceedings of the Conference held in the Mathematical Research Institute at Oberwolfach, Black Forest, August 4–10, 1963/Abhandlungen zur Tagung im Mathematischen Forschungsinstitut Oberwolfach, Schwarzwald, vom 4.–10. August 1963. [*S.l.*: *s.n.*], 1964. P. 245–251.

PAN, V Ya. Methods of computing values of polynomials. **Russian Mathematical Surveys**, IOP Publishing, v. 21, n. 1, p. 105, 1966.

PARKS, Thomas W; BURRUS, C Sidney. **Digital filter design**. [*S.l.*]: Wiley-Interscience, 1987.

PEDREGOSA, F. *et al.* Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PLATT, John. Using analytic QP and sparseness to speed training of support vector machines. **Advances in neural information processing systems**, v. 11, 1998.

PRICE, Kenneth V; STORN, Rainer M; LAMPINEN, Jouni A. The differential evolution algorithm. **Differential evolution: a practical approach to global optimization**, Springer, p. 37–134, 2005.

PROVOST, Foster J; FAWCETT, Tom; KOHAVI, Ron *et al.* The case against accuracy estimation for comparing induction algorithms. *In*: ICML. [*S.l.*: *s.n.*], 1998. v. 98, p. 445–453.

ROMAN, Ibai; SANTANA, Roberto; MENDIBURU, Alexander; LOZANO, Jose A. In-depth analysis of SVM kernel learning and its components. **Neural Computing and Applications**, Springer, v. 33, n. 12, p. 6575–6594, 2021.

ROMANO, Joseph D *et al.* PMLB v1.0: an open source dataset collection for benchmarking machine learning methods. **arXiv preprint arXiv:2012.00058v2**, 2021.

SARLE, Warren S. Neural networks and statistical models. *In*: PROCEEDINGS of the 19th annual SAS users group international conference. [*S.l.*: *s.n.*], 1994. v. 13.

SCHÖLKOPF, Bernhard; SMOLA, Alexander J. **Learning with kernels: support vector machines, regularization, optimization, and beyond**. [*S.l.*]: MIT press, 2002.

SHALEV-SHWARTZ, Shai; SINGER, Yoram; SREBRO, Nathan. Pegasos: Primal estimated sub-gradient solver for svm. *In*: PROCEEDINGS of the 24th international conference on Machine learning. [*S.l.*: *s.n.*], 2007. P. 807–814.

SHAWE-TAYLOR, John; CRISTIANINI, Nello. **Kernel methods for pattern analysis**. [*S.l.*]: Cambridge university press, 2004.

SMITH, Jack W; EVERHART, James E; DICKSON, WC; KNOWLER, William C; JOHANNES, Robert Scott. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *In*: AMERICAN MEDICAL INFORMATICS ASSOCIATION. PROCEEDINGS of the annual symposium on computer application in medical care. [*S.l.*: *s.n.*], 1988. P. 261.

SMOLA, Alex J; SCHÖLKOPF, Bernhard. A tutorial on support vector regression. **Statistics and computing**, Springer, v. 14, p. 199–222, 2004.

SOKOLOVA, Marina; LAPALME, Guy. A systematic analysis of performance measures for classification tasks. **Information processing & management**, Elsevier, v. 45, n. 4, p. 427–437, 2009.

STAHL, Herbert; SCHMELZER, Thomas. An extension of the '1/9'-problem. **Journal of computational and applied mathematics**, Elsevier, v. 233, n. 3, p. 821–834, 2009.

STORN, Rainer; PRICE, Kenneth. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization**, Springer, v. 11, p. 341–359, 1997.

SUTTON, Richard S; BARTO, Andrew G. **Reinforcement learning: An introduction**. [*S.l.*]: MIT press, 2018.

SYARIF, Iwan; PRUGEL-BENNETT, Adam; WILLS, Gary. SVM parameter optimization using grid search and genetic algorithm to improve classification performance. **TELKOMNIKA (Telecommunication Computing Electronics and Control)**, v. 14, n. 4, p. 1502–1509, 2016.

TIAN, Yingjie; JU, Xuchan; QI, Zhiquan; SHI, Yong. Efficient sparse least squares support vector machines for pattern classification. **Computers & Mathematics with Applications**, Elsevier, v. 66, n. 10, p. 1935–1947, 2013.

UCI MACHINE LEARNING. **Pima Indians Diabetes Database**: Predict the onset of diabetes based on diagnostic measures. [*S.l.*], 2016. Disponível em: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database. Acesso em: 25 set. 2023.

URBANOWICZ, Ryan J; MOORE, Jason H. ExSTraCS 2.0: description and evaluation of a scalable learning classifier system. **Evolutionary intelligence**, Springer, v. 8, p. 89–116, 2015.

VAN ROSSUM, Guido; DRAKE, Fred L *et al.* **Python reference manual**. [*S.l.*]: Centrum voor Wiskunde en Informatica Amsterdam, 1995. v. 111.

VAPNIK, Vladimir. The support vector method of function estimation. *In*: NONLINEAR modeling: Advanced black-box techniques. [*S.l.*]: Springer, 1998. P. 55–85.

YU, Xiaobing; WANG, Xuming. A novel hybrid classification framework using SVM and differential evolution. **Soft Computing**, Springer, v. 21, p. 4029–4044, 2017.